UNIVERZITET U NOVOM SADU
PRIRODNO - MATEMATIČKI FAKULTET
DEPARTMAN ZA MATEMATIKU I INFORMATIKU

Milica Matijević

Dinamičko programiranje i primena

Master rad

Mentor:
Prof. dr Sanja Rapajić

Novi Sad, 2021.

Sadržaj

Uvod

Optimizacija je matematička disciplina koja se bavi problemima optimizacije, njihovim osobinama, razvojem, kao i implementacijom algoritama koji se koriste za rešavanje takvih problema. U praksi su česti problemi kod kojih je potrebno odrediti ekstremnu vrednost funkcije cilja nad nekim skupom ograničenja. Na primer, menadžer proizvodnje organizuje proizvodnju u fabrici tako da profit bude maksimalan, pri čemu su na raspolaganju ograničeni resursi (broj radnika, radno vreme, kapital).

Teorijski gledano, ovakvi problemi optimizacije se mogu rešavati simulacijom svih dopustivih stanja. Međutim, iako je ovakav pristup korektan on je neefikasan za one probleme kod kojih postoji veliki broj dopustivih stanja.

Drugi mogući pristup se zasniva na biranju lokalno najboljeg rešenja na svakoj grani grananja. Na ovaj način se smanjuje složenost problema kao i sam prostor pretrage, ali se ne može uvek utrvditi tačnost ovakvih algoritama.

Ovaj rad se bavi dinamičkim programiranjem, čijim tehnikama se jedna klasa problema optimizacije efikasno može rešavati. Za takve probleme se kaže da imaju *optimalnu podstrukturu.* Uz pomoć dinamičkog programiranja se smanjuje složenost algoritma - najčešće od eksponencijalne do polinomne.

U ovom master radu su definisane osnove dinamičkog programiranja i navedeni su primeri primene dinamičkog programiranja. U prvom poglavlju dat je istorijski pregled razvoja dinamičkog programiranja i definisani su njegovi osnovni koncepti. Primer određivanja $n-$ tog Fibonačijevog broja za neko $n \in \mathbb{N}$ jedan je od osnovnih primera primene dinamičkog programiranja. Drugo poglavlje je posvećeno problemima optimizacije na grafovima, kao što su problemi pronalaska najkraćeg puta u grafu i problem trgovačkog putnika. U trećem poglavlju je predstavljen problem množenja niza matrica. Četvrto i peto poglavlje posvećeni su algoritmima za pretragu stringova. Ovi algoritmi imaju primenu u biologiji, za analizu DNK sekvenci. Šesto poglavlje se bavi problemom proste raspodele jednorodnog resursa, pri čemu je količina resursa ograničena. Dat je primer raspodele jednakih mašina na nekoliko pogona, kao i raspodele resursa na nekoliko različitih proizvodnih linija. Takođe je opisana i optimizacija poslovnog procesa fabrike mernih transformatora iz Zaječara. Sedmo poglavlje se bavi problemom raspodele poslova na mašine, pri čemu svaka mašina može da obavlja posao tipa 1 i 2. Takođe se uzima u obzir da će jedan deo mašina posle određenog perioda biti amortizovan (odnosno neće biti za dalju upotrebu). U osmom poglavlju je prikazan problem složene raspodele jednorodnog resursa. Problem ranca koji je predstavljen u radu predstavlja specijalni slučaj problema složene raspodele jednorodnog resursa. Tehnikama dinamičkog programiranja su rešene obe varijante ovog problema - sa i bez ponavljanja. U devetom poglavlju je opisan i rešen specijalni slučaj transportnog problema. U pitanju je takozvani

dinamički transportni problem gde se prevoz tereta obavlja transportnim sredstvom koje snabdeva robom više odredišta iz samo jednog ishodišta. Za rešavanje nekih problema korišćen je programski paket Matlab. Deseto poglavlje je zaključak.

1 Dinamičko programiranje

U periodu posle Drugog svetskog rata primećeno je da se mnoge interesantne i značajne aktivnosti mogu klasifikovati kao višeetapni procesi odlučivanja. Postojeće tehnike za rešavanje ovih problema su bile ograničene, posebno za probleme velikih dimenzija. To je dovelo do pojave novih matematičkih teorija i metoda među kojima je i dinamičko programiranje, koje je predstavljalo novi pristup koji se zasniva na principu optimalnosti.

Dinamičko programiranje (DP), kao i sam termin, uveo je američki matematičar Ričard Belman (Richard Bellman, 1920-1984) 1957. godine. On je proučavao problem proučavajući hijerarhiju potproblema sadržanih u glavnom problemu i rešavanje počinjao od najjednostavnijih. Belman se smatra osnivačem dinamičkog programiranja jer je upravo on dao čvrstu matematičku osnovu za rešavanje ovakvih problema.

Poznato je da su se tehnike koje je Belman opisao koristile i ranije (u periodu od 1930 - 1940. godine) iako sam termin "dinamičko programiranje" tada nije bio definisan. Pjer Mase (Pierre Massé, 1898-1987) francuski ekonomista, inženjer i primenjeni matematičar se služio algoritmima dinamičkog programiranja za optimizaciju rada hidroelektrana za vreme Višijevskog režima (1940-1944). Džon fon Nojman (John von Neumann, 1903-1957) i Oskar Morgenštern (Oskar Morgenstern, 1902-1977) su korisitli algoritme dinamičkog programiranja da odrede pobedničku strategiju u igrama dva igrača sa kartama (na primer dame). Za vreme Drugog svetskog rata, engleski matematičar, logičar i kriptograf Alan Turing (Alan Mathison Turing, 1912-1954) je radio u Britanskoj organizaciji za razbijanje šifara (Government Code and Cypher School) i koristio slične metode prilikom razbijanja nemačkih šifara.

Termin "programiranje" koji se javlja u nazivu se može shvatiti kao "planiranje", jer je dinamičko programiranje upravo i osmišljeno kao optimalno planiranje *višeetapnih procesa upravljanja.* Ovakvi procesi se sastoje od niza etapa, i na svakoj etapi je potrebno doneti odluku. Dakle, kod problema dinamičkog programiranja, potrebno je odrediti onaj niz odluka za koji se postiže najbolja vrednost neke zadate funkcije cilja. To znači da je optimizacija zadate funkcije cilja posledica niza međusobno zavisnih odluka koje grade politiku odlučivanja.

Za razliku od linearnog programiranja, koje ima jasno definisanu standardnu formulaciju opšteg modela, kao i metode kojima se ova klasa problema rešava - kod dinamičkog programiranja to nije slučaj. Za različite probleme razvijaju se različite tehnike za njihovo rešavanje koje zavise od prirode i specifičnosti datog problema.

Ideja dinamičkog programiranja je korišćenje principa domina: sve domine će popadati ukoliko se poruši prva domina u nizu. Odnosno, da bi se rešio neki problem, potrebno je taj problem podeliti na manje probleme (potprobleme). Algoritmi "podeli pa vladaj" (eng. divide and conquer) vrše particiju početnog problema, na više nezavisnih potproblema, a potom nastupa rekurzivno rešavanje potproblema,

kako bi se njihovim spajanjem dobilo rešenje polaznog problema.

DP je do sada uspešno primenjeno na brojne praktične probleme (kako determinističke tako i probabilističke prirode) - i to ne samo one koji su prirodno struktuirani kao višeetapni procesi, nego i na one probleme koji na prvi pogled ne predstavljaju takve zadatke, već predstavljaju probleme matematičkog programiranja ili optimizacije na grafovima (o čemu će biti reč u nastavku). Za sve probleme dinamičkog programiranja je karakteristično da imaju etapnu prirodu, etape karakterišu partikularna rešenja i rešenja na svakoj etapi imaju osobinu aditivnosti.

Kroz rešavanje problema pronalaska $n-$tog Fibonačijevog broja, koji je opšte poznat problem, mogu se ilustrovati osnovni koncepti dinamičkog programiranja.

1.1 Fibonačijev niz

Fibonačijev niz predstavlja niz brojeva koji je primećen u mnogim fizičkim, hemijskim i biološkim pojavama. Ime je dobio po italijanskom matematičaru Leonardu Fibonačiju koji je prilikom proučavanja razmnožavanja zečeva uočio zanimljiv redosled brojeva. Fibonačijevi brojevi su od velikog značaja jer se pojavljuju u prirodi, od suncokreta, uragana pa sve do galaksije.

Ovaj niz je vrsta rekurzivnog niza u kome zbir prethodna dva broja u nizu daju vrednost narednog člana niza. Indeksiranje niza počinje od nule, a prva dva člana niza su 0 i 1. Dakle,

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2},\ n \geqslant 2. \end{aligned}$$

Tako se dobija niz $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$
Rekurzivna definicija Fibonačijevih brojeva daje algoritam pomoću kog se može izračunati $n-$ti elemenat tog niza. Jedan takav program dat je u nastavku.

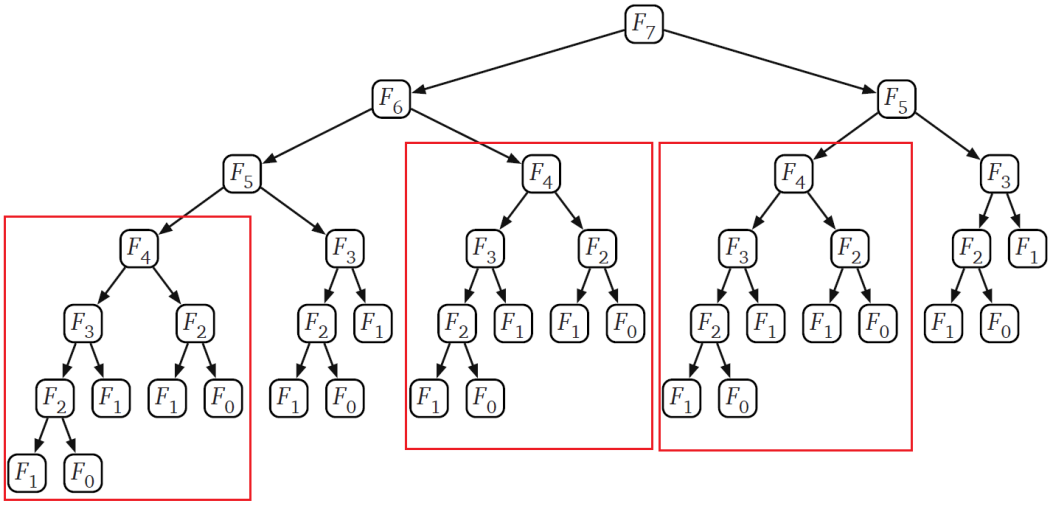
```
% INPUT: n
% OUTPUT: F - n-ti Fibonacijev broj

function F=fib(n)
if n==0
  F=0;
elseif n==1
  F=1;
else
  F=fib(n-1)+fib(n-2);
end
```

Neka je $T(n)$ broj rekurzivnih poziva algoritma fib

$$T(0) = 1$$
$$T(1) = 1$$
$$T(n) = T(n-1) + T(n-2) + 1.$$

Indukcijom se može pokazati da važi $T(n) = 2F_{n+1} - 1$. Dakle, vreme koje je potrebno za rekurzivno rešavanje problema je eksponencijalno. To sledi iz činjenice da prilikom izvršavanja rekurzivne funkcije dolazi do preklapanja rekurzivnih poziva, tj. pojedini delovi kôda se izvršavaju nekoliko puta. Takvi programi su po pravilu izuzetno neefikasni.

Kada se pozove rekurzivno fib(n), poziva se jednom fib$(n-1)$, dva puta fib$(n-2)$, tri puta fib$(n-3)$, pet puta fib$(n-4)$ i generalno F_{k-1} puta se poziva fib$(n-k)$, za svako $0 \leqslant k < n$. To ilustruje Slika 1 za slučaj $n = 7$.



Slika 1: Prikaz izvršavanja rekurzivnog koda. $F_i = \text{fib}(i)$, $i \in \mathbb{N}$. Na primer F_4 se računa 3 puta.

Do efikasnijeg rešenja moguće je doći primenom tehnika dinamičkog programiranja. Dinamičko programiranje se najčešće javlja u dva oblika:

1. Prva tehnika zadržava rekurzivnu definiciju, ali se u nekoj dodatnoj strukturi podataka (najčešće je to niz ili matrica) beleže svi rezultati rekurzivnih poziva, da bi se u narednim pozivima, u kojima su parametri isti, samo očitali iz te strukture.

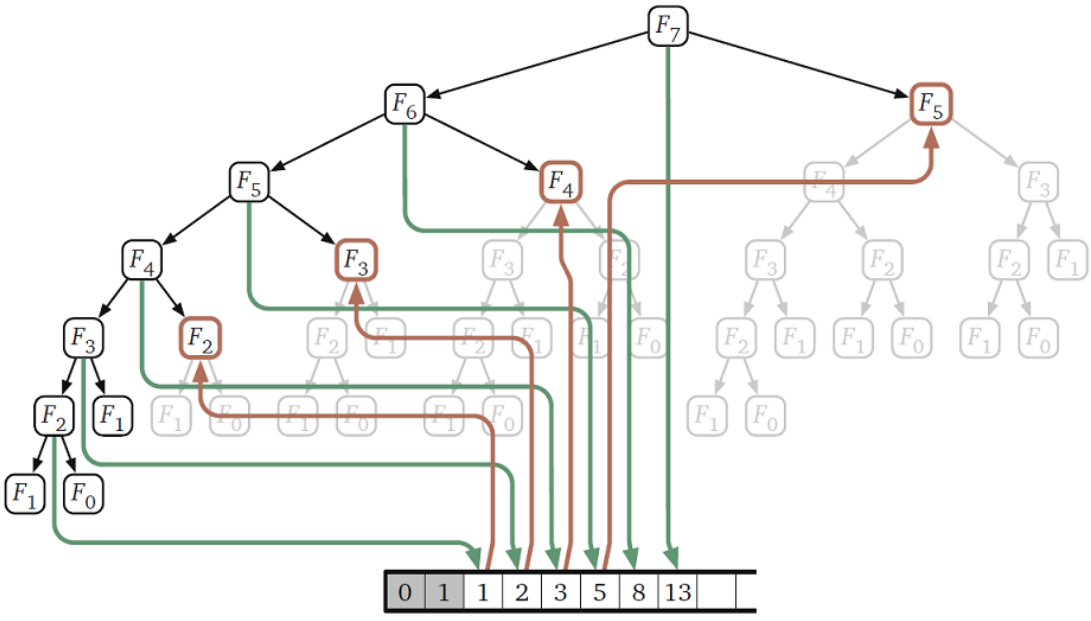
2. Druga tehnika u potpunosti uklanja rekurziju i tu pomoćnu strukturu podataka popunjava sistematično u nekom iscrpnom redosledu.

U nastavku je dat algoritam koji opisuje prvu tehniku. Formiran je niz *memo* koji predstavlja dodatnu strukturu u kojoj se beleže prethodno dobijeni rezultati.

Algoritam 1: Nalaženje $n-$tog Fibonačijevog broja

```
memo = { }
fib(n) :
        if n in memo : return memo[n]
        else if n = 0 : return 0
        else if n = 1 : return 1
        else: f = fib(n − 1) + fib(n − 2)
        memo[n] = f
        return f
```

Primetimo da se na ovaj način prvo računa fib(2), zatim fib(3),..., sve do fib(n), pri čemu je $n \in \mathbb{N}$. Takođe fib(i) se računa tek kada je već izračunat njegov prethodnik fib($i-1$), za sve $i \geqslant 2$ i pri tome se fib(i) računa tačno jednom, za svaki indeks $i \geqslant 2$ (Slika 2).



Slika 2: Zelene strelice na slici ukazuju na to da se izračunate vrednosti upisuju u niz koji je u prethodnom algoritmu nazvan *memo*. Crvene strelice ukazuju na to da se iz niza *memo* čitaju prethodno izračunate vrednosti.

Druga tehnika podrazumeva da se ukloni rekurzija i da se sve vrednosti pomoćne strukture (u ovom slučaju niza) popune u nekom redosledu. S obzirom da vrednosti

niza na višim pozicijama zavise od onih na nižim, niz se popunjava s leva na desno. Prva dva mesta se na početku popunjavaju nulom i jedinicom, a zatim se ostali elementi niza računaju na osnovu prethodne dve vrednosti.

```
% INPUT: n
% OUTPUT: F - n-ti Fibonacijev broj

function F=fibonacci(n)
  fib = zeros(1,n+1);
  fib(2)=1;
  for i=3:n+1
    fib(i) =fib(i-1)+fib(i-2);
  end
  F=fib(n+1);
```

Može se primetiti da u ovom slučaju niz dužine n nije neophodan. Svaka vrednost u nizu zavisi samo od prethodne dve, pa je stoga potrebno čuvati samo prethodne dve vrednosti niza, kao što je ilustrovano u narednom programu.

```
% INPUT: n
% OUTPUT: F - n-ti Fibonacijev broj

function F=fibonacci2(n)

  fib = zeros(1,2);
  fib(2)=1;
  for i=3:n+1
    pom = fib(1)+fib(2);
    fib(1)=fib(2);
    fib(2)=pom;
  end
  F=fib(2);
```

Sledeći koraci su primenjeni prilikom rešavanja ovog problema:

1. Problem se definiše rekurzivno i ta rekurzivna definicija je neefikasna jer se funkcija za iste argumente poziva više puta.

2. U pomoćnom nizu/matrici čuvaju se izračunati rezultati rekurzivnih poziva.

3. Umesto rekurzije u kojoj se već izračunate vrednosti iz pomoćnog niza pozivaju, rekurzija se eliminiše i niz/matrica se popunjava iscrpno nekim redosledom.

4. Na kraju se vrši memorijska optimizacija. Umesto istovremenog pamćenja svih elemenata niza/matrice, pamte se samo one vrednosti koje su potrebne za dalje popunjavanje niza/matrice.

1.2 Višeetapni procesi upravljanja

Višeetapni procesi su oni procesi u kojima je potrebno doneti niz odluka, i te odluke se donose postepeno u vremenu. Osnove višeetapnih procesa upravljanja date su u [3]. Mnogi problemi u tehnici, ekonomiji, biologiji, fizici itd. predstavljaju višeetapne procese i oni se mogu rešavati primenom metoda dinamičkog programiranja. Potrebno je da svaki ovakav proces ima definisanu funkciju cilja koju je potrebno minimizirati ili maksimizirati, kao i ograničenja koja se uzimaju u obzir u toku procesa. Osnovni cilj je dobijanje optimalnog niza (plana) upravljanja, odnosno pronalaženje onog niza odluka za koji se postiže najbolja vrednost zadate funkcije cilja. Višeetapni procesi upravljanja imaju sledeća svojstva:

- Proces se posmatra u konačno mnogo diskretnih trenutaka (etapa) i na svakoj etapi je potrebno doneti neku odluku, odnosno vrši se optimizacija funkcije cilja (na primer maksimizacija dobiti, minimizacija troškova i slično);

- Ponašanje procesa na svakoj etapi je okarakterisano vrednostima odgovarajućih parametara kojima se meri ovo ponašanje i koji definišu trenutno stanje procesa. Početno stanje procesa je unapred poznato;

- Na svakoj etapi vrši se (na deterministički način) izbor vrednosti parametara upravljanja iz zadatog skupa dopustivih vrednosti i taj skup dopustivih vrednosti zavisi od prethodnih stanja procesa;

- Kada se izabere neko upravljanje na trenutnoj etapi, proces menja svoje ponašanje i prelazi iz prethodnog u novo stanje - koje predstavlja novu etapu. Novo stanje zavisi od prethodnog stanja i izabranog upravljanja.

Prethodno opisan višeetapni proces ima konačan broj etapa, na deterministički način se vrši izbor upravljanja i prelazak u novo stanje ne zavisi od trenutne etape, već samo od trenutnog stanja i upravljanja, pa se ovakav proces još naziva i *konačni, deterministički i stacionarni.*

Dinamičko programiranje je tehnika za rešavanje ovakve klase problema. Princip "podeli pa vladaj" se koristi u mnogim algoritmima - da bi se rešio složeniji problem, on se razbija na manje probleme koji se mogu rešavati nezavisno. U nekim slučajevima, kada nije poznato koje manje probleme je potrebno rešavati, reše se svi, a potom se rezultati zapamte, kako bi se upotrebili za rešavanje polaznog problema.

Postoje dva osnovna nedostatka ove tehnike. Prvo, postoje slučajevi kada nije moguće kombinovanje dva rešenja potproblema kako bi se dobilo rešenje složenijeg problema. Drugo, broj manjih potproblema često je izuzetno velik.

Ipak, postoji jasno definisana klasa problema za koje dinamičko programiranje daje efikasno rešenje. To su upravo oni problemi koji zadovoljavaju *Belmanov princip optimalnosti,* o kome će kasnije biti reči.

1.2.1 Osnovni pojmovi

Posmatra se konačno mnogo vremenskih trenutaka (etapa)

$$t = (t_0, t_1, ..., t_n).$$

Ponašanje procesa na svakoj etapi je okarakterisano *vektorom stanja.* Vektor stanja je jedan realni vektor i pri tome vrednost njegovih koordinata zavisi od etape na kojoj se proces posmatra. Dakle, stanje procesa se na svakoj etapi t može zapisati u sledećem obliku

$$r(t) = (r_1(t), r_2(t), ..., r_m(t)),$$

pri čemu je m zadata dimenzija vektora stanja.
Ako se stanje procesa na $i-$toj etapi označi sa $r_i = r(t_i)$, onda se dobijeni niz $r_0, r_1, ..., r_n$ naziva *trajektorija procesa.* Stanje r_0 je početno stanje i ono je unapred dato, a r_n se naziva završno stanje trajektorije.
Na svakoj etapi procesa, potrebno je na deterministički način izvršiti izbor odgovarajućih parametara upravljanja. Upravljanje koje se bira na nekoj etapi ima uticaj na dalju trajektoriju procesa i definisano je pomoću *vektora upravljanja*

$$u(t) = (u_1(t), u_2(t), ..., u_k(t)).$$

Ako se upravljanje na $i-$toj etapi označi sa $u_i = u(t_i)$, onda ovo upravljanje treba da bude dopustivo, odnosno treba da pripada skupu dopustivih upravljanja, koji u opštem slučaju zavisi od prethodnog stanja r_{i-1} i koji se označava sa $U_i(r_{i-1})$, dakle

$$u_i \in U_i(r_{i-1}).$$

Zakon prelaska procesa iz stanja r_{i-1} u stanje r_i, pod uticajem upravljanja u_i definiše se na sledeći način

$$r_i = w(r_{i-1}, u_i), \text{ za } i = 1, 2, ..., n,$$

pri čemu je $w(r, u)$ jedna vektorska funkcija, tj. $w(r, u) = (w_1(r, u), w_2(r, u), ..., w_m(r, u))$, i $w_i(r, u)$ je realna funkcija od $m + k$ promenljivih za svako $i = 1, 2, ..., m$.

Polazeći od nekog unapred zadatog početnog stanja r_0, na svakoj etapi t_i, $i = 1, 2, ..., n$ se iz skupa svih dopustivih upravljanja bira upravljanje u_i, pod čijem uticajem proces prelazi iz stanja r_{i-1} u stanje r_i prema unapred definisanom zakonu

$w(r_{i-1}, u_i)$. Dobijena trajektorija $r_0, r_1, ..., r_n$ odgovara nizu dopustivih upravljanja $u_1, u_2, ..., u_n$ i početnom stanju r_0. U Tabeli 1 je dat detaljniji prikaz ponašanja jednog ovakvog procesa.

Etapa	Prethodno stanje procesa	Upravljanje	Trenutno stanje procesa
t_1	r_0	$u_1 \in U_1(r_0)$	$r_1 = w(r_0, u_1)$
t_2	r_1	$u_2 \in U_2(r_1)$	$r_2 = w(r_1, u_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_i	r_{i-1}	$u_i \in U_i(r_{i-1})$	$r_i = w(r_{i-1}, u_i)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	r_{n-1}	$u_n \in U_n(r_{n-1})$	$r_n = w(r_{n-1}, u_n)$

Tabela 1: Višeetapni proces upravljanja

Zadatak optimizacije u višeetapnim procesima upravljanja

$$\begin{aligned} \min \text{ (max)} \quad & f(r_0, r_1, ..., r_n, u_1, u_2, ..., u_n) \\ & r_i = w(r_{i-1}, u_i),\ i = 1, 2, ..., n \\ & u_i \in U_i(r_{i-1}),\ i = 1, 2, ..., n \\ & r_0 - \text{početno stanje.} \end{aligned}$$

Problem predstavlja traženje minimuma/maksimuma neke realne funkcije f na skupu svih nizova dopustivih upravljanja ovog procesa. Metodologija dinamičkog programiranja može se upotrebiti samo na one probleme gde je funkcija cilja f u *separabilnom obliku,* odnosno f predstavlja zbir ili proizvod n izraza, pri čemu $i-$ti izraz zavisi samo od promenljive u_i. To znači da se zadatak optimizacije može predstaviti u jednom od sledećih oblika:

$$\begin{aligned} \min \text{ (max)} \quad & \sum_{i=1}^{n} f_i(r_{i-1}, u_i) \\ & r_i = w(r_{i-1}, u_i),\ i = 1, ..., n \\ & u_i \in U_i(r_{i-1}),\ i = 1, ..., n \\ & r_0 - \text{početno stanje.} \end{aligned} \qquad \begin{aligned} \min \text{ (max)} \quad & \prod_{i=1}^{n} f_i(r_{i-1}, u_i) \\ & r_i = w(r_{i-1}, u_i),\ i = 1, ..., n \\ & u_i \in U_i(r_{i-1}),\ i = 1, ..., n \\ & r_0 - \text{početno stanje.} \end{aligned}$$

Separabilni oblik funkcije se često prirodno nameće u realnim višeetapnim procesima, pa stoga ne predstavlja veliko ograničenje, jer se najčešće svakoj etapi pridružuje neki "efekat" (npr. prihod, trošak itd), koji zavisi samo od upravljanja u_i i prethodnog stanja r_{i-1}. Potrebno je minimizirati, odnosno maksimizirati ukupan posmatrani "efekat" koji predstavlja zbir ili proizvod "efekata" na svim etapama (na primer ukupan prihod ili trošak).

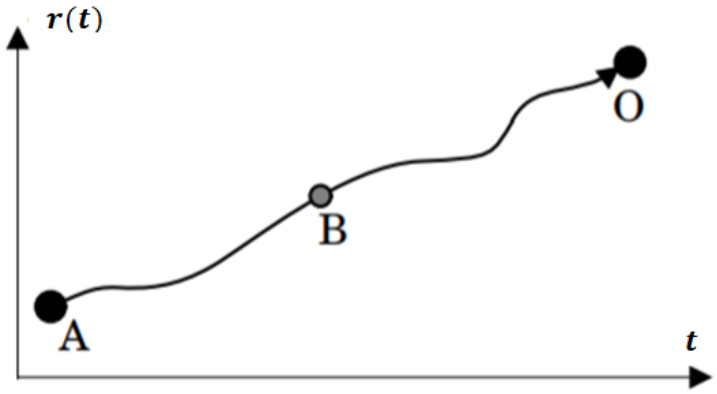
Niz upravljanja $u_1^*, u_2^*, ..., u_n^*$ za koji funkcija cilja dostiže minimum, odnosno maksimum naziva se *optimalni niz upravljanja,* a njemu odgovarajuća trajektorija $r_0^*, r_1^*, ..., r_n^*$ naziva se *optimalna trajektorija procesa.*

1.2.2 Belmanov princip optimalnosti

Ričard Belman je 1957. godine definisao princip optimalnosti, koji predstavlja osnovu dinamičkog programiranja, na sledeći način:
Optimalni niz upravljanja ima svojstvo da, bez obzira na početno stanje i početnu odluku, sve preostale odluke moraju predstavljati optimalni niz upravljanja u odnosu na stanje koje proizilazi iz prve odluke.[4]

Princip optimalnosti je posledica separabilnog oblika funkcije cilja, i u literaturi se definiše na različite načine. U nastavku je data još jedna formulacija principa optimlanosti koja je bliska originalnoj Belmanovoj.

Optimalni niz upravljanja za ceo proces je i po delovima optimalan, odnosno on ima osobinu da je, bez obzira na upravljanja koja su dovela do nekog stanja na nekoj etapi, niz upravljanja na ostalim etapama optimalan, u odnosu na to stanje kao početno. Formalno, ova osobina se može zapisati na sledeći način: Ako je $u_1^*, u_2^*, ..., u_n^*$ optimalni niz upravljanja i $r_0, r_1^*, ..., r_n^*$ optimalna trajektorija procesa, tada za ma koje dve etape i i j, $1 \leq i \leq j \leq n$, važi da je i $u_i^*, u_{i+1}^*, ..., u_j^*$ optimalni niz upravljanja za deo procesa od $i-$te do $j-$te etape, pri čemu je r_{i-1}^* početno, a r_j^* završno stanje procesa.
U slučaju da je $j = n$ dobija se originalni Belmanov princip optimalnosti, definisan 1957. godine.



Slika 3: Ako B pripada optimalnoj trajektoriji procesa čije je početno stanje A, a krajnje O, onda je optimalna trajektorija za deo procesa od B do O ista kao i za ceo proces.

1.2.3 Rekurentne relacije i optimalni niz upravljanja

U nastavku se posmatra sledeći problem optimizacije

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} f_i(r_{i-1}, u_i) \\ & r_i = w(r_{i-1}, u_i),\ i = 1, 2, ..., n \\ & u_i \in U_i(r_{i-1}),\ i = 1, 2, ..., n \\ & r_0 - \text{početno stanje.} \end{aligned} \tag{1.1}$$

Neka je $F_i(r),\ i = 1, 2, ..., n$ maksimalna vrednost funkcije cilja za deo procesa od $i-$te do $n-$te etape u odnosu na proizvoljno početno stanje r na etapi $i-1$. Tada je

$$F_i(r) = \max_{u_i, ..., u_n} \sum_{l=i}^{n} f_l(r_{l-1}, u_l), \tag{1.2}$$

pri čemu je $r = r_{i-1}$ početno stanje za ostatak procesa.
U slučaju da je $i = n$

$$F_n(r) = \max_{u_n \in U_n(r)} f_n(r, u_n). \tag{1.3}$$

Kada je $i < n$ važi

$$F_i(r) = \max_{u_1,\ldots,u_n} \sum_{l=i}^{n} f_l(r_{l-1}, u_l) = \max_{u_i \in U_i(r)} \left\{ f_i(r, u_i) + \max_{u_{i+1},\ldots,u_n} \sum_{l=i+1}^{n} f_l(r_{l-1}, u_l) \right\} =$$
$$= \max_{u_i \in U_i(r)} \{ f_i(r, u_i) + F_{i+1}(r_i) \}, \text{ za } 1 \leqslant i \leqslant n-1, \tag{1.4}$$

jer je funkcija $F_i(r)$ definisana kao maksimum separabilne funkcije i promenljiva u_i se nalazi samo u prvom sabirku.
Kako je $r_i = w(r, u_i)$, kada se to uvrsti u jednačinu (1.4) dobija se

$$F_i(r) = \max_{u_i \in U_i(r)} \{ f_i(r, u_i) + F_{i+1}(w(r, u_i)) \}, \text{ za } 1 \leqslant i \leqslant n-1. \tag{1.5}$$

Na ovaj način se dobija opšti oblik rekurentnih relacija (formule (1.3) i (1.5)) pomoću kojih se može rešiti početni problem. $F_1(r_0)$ predstavlja maksimalnu vrednost funkcije cilja problema (1.1). Dakle, osnovna ideja je rešavanje problema "unazad" (eng. Bottom-up approach). To znači da polazeći od poslednje, $n-$te etape i krećući se unazad, etapa po etapa, na svakoj etapi i se za svako moguće stanje r određuje optimalna vrednost funkcije cilja za taj deo procesa (od $i-$te do $n-$te etape) i to na rekurzivan način, odnosno korišćenjem ranije izračunatih optimalnih vrednosti funkcije cilja na prethodno razmatranim etapama. To znači da nema vraćanja na isti potproblem i svaki potproblem se rešava samo jednom.
Napomena:

- U slučaju da je (1.1) problem minimizacije, u formulama (1.2), (1.3), (1.4) i (1.5) će umesto max biti min.
- U slučaju da je funkcija cilja problema (1.1) u obliku proizvoda, u formulama (1.2), (1.3), (1.4) i (1.5) će svuda umesto operacije sabiranja biti operacija množenja.
- Formule (1.3), (1.4) i (1.5) su dobijene korišćenjem principa "unazad". Kod nekih problema, etape nisu nužno uzastopni vremenski trenuci, što znači da redosled razmatranja etapa prilikom definisanja rekurentnih formula može biti proizvoljan. Na taj način se dobijaju ekvivalentni oblici rekurentnih formula (1.3), (1.4) i (1.5).

Rešavanje problema optimizacije se odvija u dve faze (kao što je prikazano u Tabeli 1):

Prva faza: Izračunavanje optimalne vrednosti funkcije cilja.
Na svakoj etapi i, $i = n, n-1, ..., 1$ računa se maksimalna vrednost funkcije $F_i(r)$, koja je definisana rekurentnim formulama (1.3) i (1.5), kao i upravljanje $u_i^*(r)$ koje pripada dopustivom skupu $U_i(r)$ za koje se ova vrednost dostiže, i to za sva stanja r u koja proces može da dođe na etapi $i-1$, polazeći od unapred datog početnog stanja r_0. S obzirom da je r_0 unapred dato, na prvoj etapi se računa $F_1(r_0)$ i ono predstavlja otpimalnu vrednost funkcije cilja, dok je $u_1^*(r_0)$ optimalno upravljanje na prvoj etapi.

Druga faza: Formiranje optimalnog niza upravljanja.
U prvoj fazi su određene veličine $u_1^*(r_0), u_2^*(r), ..., u_n^*(r)$, dok se optimalni niz upravljanja $u_1^*, u_2^*, ..., u_n^*$ i optimalna trajektorija $r_0, r_1^*, ..., r_n^*$ formiraju sekvencijalno, idući od prve do poslednje etape na rekurzivni način. Na $i-$toj etapi, $i = 1, 2, ..., n$ se vrednosti za optimlano upravljanje i optimalno stanje određuju korišćenjem optimalnog stanja na prethodnoj etapi koje je već prethodno izračunato, i to na sledeći način

$$u_i^* = u_i^*(r_{i-1}^*), \text{ i } r_i^* = w(r_{i-1}^*, u_i^*).$$

	Prva faza		Druga faza	
etapa	$F_i(r),\ \forall r$	$u_i^*(r),\ \forall r$	optimalno upravljanje	optimalno stanje
n	$F_n(r)$	$u_n^*(r)$	$u_n^* = u_n^*(r_{n-1}^*)$	$r_n^* = w(r_{n-1}^*, u_n^*)$
$n-1$	$F_{n-1}(r)$	$u_{n-1}^*(r)$	$u_{n-1}^* = u_{n-1}^*(r_{n-2}^*)$	$r_{n-1}^* = w(r_{n-2}^*, u_{n-1}^*)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2	$F_2(r)$	$u_2^*(r)$	$u_2^* = u_2^*(r_1^*)$	$r_2^* = w(r_1^*, u_2^*)$
1	$F_1(r_0)$	$u_1^*(r_0)$	$u_1^* = u_1^*(r_0)$	$r_1^* = w(r_0^*, u_1^*)$

Prva faza ↓ (levo) Druga faza ↑ (desno)

Tabela 2: Šematski prikaz rešavanja problema (1.1).

Na ovaj način se polazni problem (1.1) gde je potrebno pronaći maksmimum funkcije f koja zavisi od n promenljivih svodi na rešavanje n problema nalaženja ekstrema funkcija koje zavise samo od jedne promenljive - na svakoj etapi se vrši maksimizacija odgovarajuće funkcije po promenljivoj u_i, pa je nalaženje optimalnog rešenja znatno olakšano.
Napomena. Može se desiti da na $i-$toj etapi, za neko $i \in \{1, 2, ..., n\}$ postoji više od jednog upravljanja $u_i^*(r)$ za koje funkcija $F_i(r)$ dostiže maksimum. Tada navedeni problem optimizacije nema jedinstveno rešenje.

Ako su svi dopustivi skupovi $U_i(r)$ konačni, onda je skup mogućih stanja na svakoj etapi konačan, pa se vrednosti funkcija $f_i(r,u)$ mogu zadavati tabelarno za sva moguća stanja r na etapi $i-1$ i svako $u \in U_i(r)$. Tada se određivanje vrednosti $F_i(r)$ i $u_i^*(r)$ svodi na pronalaženje maksimuma, odnosno minimuma i njemu odgovarajućeg upravljanja na konačnom skupu vrednosti koje funkcija može da ima na $U_i(r)$. Ako je skup $U_i(r)$ diskretan za svako $i = 1, 2, .., n$, onda su i odgovarajuće promenljive u_i i r_i diskretne.

Kada su skupovi $U_i(r)$ kontinualni, onda su i promenljive r_i i u_i kontinualne, funkcije $f_i(r,u)$ treba da budu analitički zadate. Onda se određivanje vrednosti $F_i(r)$ i $u_i^*(r)$ svodi na određivanje maksimuma, odnosno minimuma analitički izražene funkcije jedne ili više promenljivih.

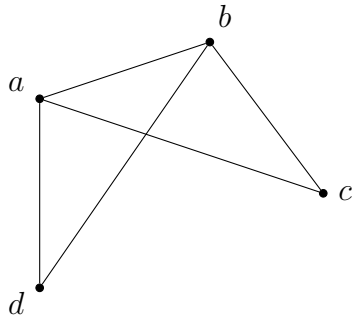
2 Grafovi

Definicija 2.1. [12] Graf G je uređeni par $G = (V, E)$. Elementi skupa V se zovu *čvorovi* (eng. vetrex), a elementi skupa E *grane* (eng. edge) grafa G. Za dati graf G, skup čvorova se označava sa $V(G)$, a skup grana sa $E(G)$.

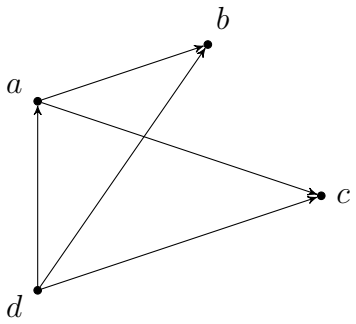
Najčešće se uzima da je u grafu broj čvorova $|V(G)| = n$, a broj grana $|E(G)| = m$. Dvoelementni podskupovi skupa V u oznaci $\{v_i, v_j\}$ ili v_iv_j su elementi skupa E.

- Elementi neuređenog para iz E su krajevi grane i kaže se da su krajevi incidentni sa tom granom, odnosno grana je incidentna sa svojim krajevima.
- Dve grane su susedne ako imaju isti čvor.
- Dva čvora su susedna ako postoji grana sa krajevima u ta dva čvora.
- Grana koja spaja čvor sa samim sobom naziva se petlja.
- Prost graf je onaj graf koji nema ni jednu petlju.
- Neorijentisani graf $G = (V(G), E(G))$ je uređen skup čvorova i grana gde je $E \subset \binom{V}{2} \cup V$.
- Orijentisani graf (digraf) $G = (V(G), E(G))$ je uređen skup čvorova i grana gde je $E \subset V \times V$. Ovaj graf ima orijentaciju, odnosno za granu ab početni čvor je a a krajnji b.

Primer 2.1. *Prost graf*
Dat je skup $V(G) = \{a, b, c, d\}$ i $E(G) = \{ab, ac, ad, bd, bc\}$.



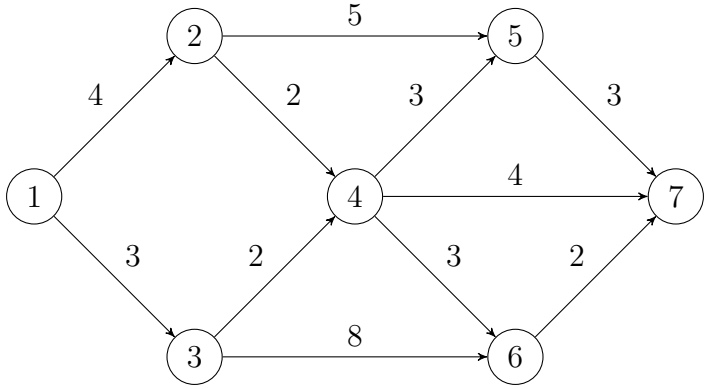
Primer 2.2. *Orijentisan graf*
Dat je skup $V(G) = \{a, b, c, d\}$ i $E(G) = \{ab, ac, da, dc, db\}$.



Čvorovi na primer mogu biti gradovi, a grane putevi između njih. Ako bi čvorovi označavali računare, grane bi mogle predstavljati način komunikacije između njih.

2.1 Najkraći put u grafu

Primer 2.3. Dat je orijentisani graf G sa $n = 7$ čvorova i svakom luku je pridružena dužina luka koja se označava sa d_{ij} (kao na slici). Potrebno je naći najkraći put od čvora 1 do čvora 7, odnosno onaj niz nadovezanih lukova koji počinje u čvoru 1 a završava se u čvoru 7, tako da je zbir dužina njegovih lukova minimalan.



Rešenje: Graf $G = (V(G), E(G))$ je zadat na sledeći način:

$V(G) = \{1, 2, 3, 4, 5, 6, 7\}$ − skup čvorova.

$E(G) = \{(1,2), (1,3), (2,4), (2,5), (3,4), (3,6), (4,5), (4,6), (4,7), (5,7), (6,7)\}-$ skup grana.

$D = \{d_{12} = 4, d_{13} = 3, d_{24} = 2, d_{25} = 5, d_{34} = 2, d_{36} = 8, d_{45} = 3, d_{46} = 3, d_{47} = 4, d_{57} = 3, d_{67} = 2\}$ − skup dužina grana.

Neka je f_i dužina najkraćeg puta od $i-$tog do $n-$tog čvora u grafu G, sa n čvorova. Da bi se odredilo f_i potrebno je za svaki susedni čvor j, naći najkraći put od $i-$tog

do $n-$tog čvora koji sadrži čvor j i ta dužina je jednaka $d_{ij} + f_j$. Tada f_i predstavlja najkraći od svih ovakvih puteva. Dakle,

$$f_i = \min_j \{d_{ij} + f_j\},\ j = 1, 2, ..., n-1$$
$$f_n = 0,$$

pri čemu se minimizacija vrši po svim čvorovima j koji su susedni čvoru i. Nalaženje najkraćeg puta može se protumačiti kao jedan višeetapni proces u kome se na svakoj etapi i bira upravljanje u_i - odnosno bira se jedan čvor puta. Skup dopustivih upravljanja $U_i(r_{i-1})$ kome mora da pripada čvor u_i je skup svih susednih čvorova prethodno izabranom čvoru u_{i-1}, koji karatkeriše stanje na prethodnoj etapi $i-1$. Početno stanje ovog procesa je čvor 1, a završno čvor 7.

Proces rešavanja problema podeljen je u etape i na svakoj etapi se minimizira funkcija jedne realne promenljive. Minimalna vredost će biti pronađena na poslednjoj etapi, odnosno optimalna vrednost funkcije cilja (dužina najkraćeg puta) je f_1. Najkraći put (optimalno rešenje) se traži u obrnutom smeru, odnosno praćenjem rednih brojeva čvorova grafa u kojima se dostižu minimumi na odgovarajućim etapama.

$f_7 = 0$
$f_6 = 2 + f_7 = 2, \quad k_6 = 7$
$f_5 = 3 + f_7 = 3, \quad k_5 = 7$

$$f_4 = \min \begin{cases} 3 + f_5 \\ 3 + f_6 \\ 4 + f_7 \end{cases} = \min \begin{cases} 3 + 3 \\ 3 + 2 \\ 4 + 0 \end{cases} = \min\{6, 5, 4\} = 4, \quad k_4 = 7$$

$$f_3 = \min \begin{cases} 2 + f_4 \\ 8 + f_6 \end{cases} = \min \begin{cases} 2 + 4 \\ 8 + 2 \end{cases} = \min\{6, 10\} = 6, \quad k_3 = 4$$

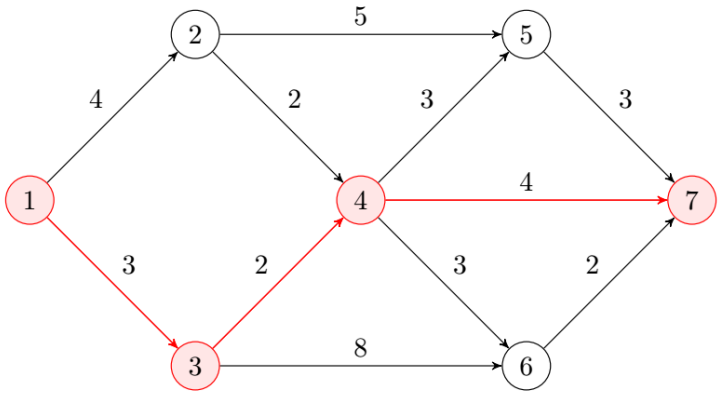
$$f_2 = \min \begin{cases} 2 + f_4 \\ 5 + f_5 \end{cases} = \min \begin{cases} 2 + 4 \\ 5 + 3 \end{cases} = \min\{6, 8\} = 6, \quad k_2 = 4$$

$$f_1 = \min \begin{cases} 4 + f_2 \\ 3 + f_3 \end{cases} = \min \begin{cases} 4 + 6 \\ 3 + 6 \end{cases} = \min\{10, 9\} = 9, \quad k_1 = 3.$$

$t_0 = r_0 = 1$
$t_1, u_1 \in U_1(r_0) = \{2, 3\}, u_1 = r_1 = 3$
$t_2, u_2 \in U_2(r_1) = \{4, 6\}, u_2 = r_2 = 4$
$t_3, u_3 \in U_3(r_2) = \{5, 6, 7\}, u_3 = r_3 = 7$

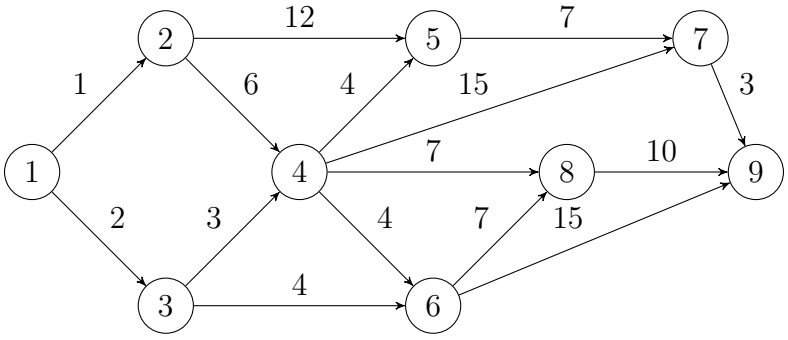
etapa i	prethodno stanje r_{i-1}	upravljanje u_i	skup dopustivih upravljanja $U_i(r_{i-1})$	trenutno stanje $r_i = u_i$
1	1	3	$\{2, 3\}$	3
2	3	4	$\{4, 6\}$	4
3	4	7	$\{5, 6, 7\}$	7

Tabela 3: Višeetapni proces formiranja puta u grafu.



Slika 4: Najkraći put u grafu je: $1 \to 3 \to 4 \to 7$, i njegova dužina je 9.

Primer 2.4. [11] Džon svakoga dana odlazi na posao i odlučio je da odredi najkraći put od kuće do posla. U datom grafu, čvor 1 predstavlja kuću, a čvor 9 posao. Svakoj grani u grafu je pridruženo vreme potrebno za prelaženje tog dela puta izraženo u minutima. Na primer, potrebno je 15 minuta kako bi se prešao put od četvrtog do sedmog čvora.



Rešenje:

$V(G) = \{1,2,3,4,5,6,7,8,9\}$ − skup čvorova.
$E(G) = \{(1,2),(1,3),(2,4),(2,5),(3,4),(3,6),(4,5),(4,6),(4,7),(4,8),(5,7),(6,8),$
$(6,9),(7,9),(8,9)\}$ − skup grana
$T = \{t_{12}=1, t_{13}=2, t_{24}=6, t_{25}=12, t_{34}=3, t_{36}=4, t_{45}=4, t_{46}=4, t_{47}=15,$
$t_{48}=7, t_{57}=7, t_{68}=7, t_{69}=15, t_{79}=3, t_{89}=10\}$ − skup dužina grana.

Neka je f_i vreme koje je potrebno da se pređe put od čvora i do čvora 9. Kao i u prethodnom primeru, $f_1, f_2, ..., f_n$ zadovoljavaju sledeći sistem jednačina

$$f_i = \min_j\{t_{ij} + f_j\},\ j = 1,2,...,n-1$$
$$f_n = 0,$$

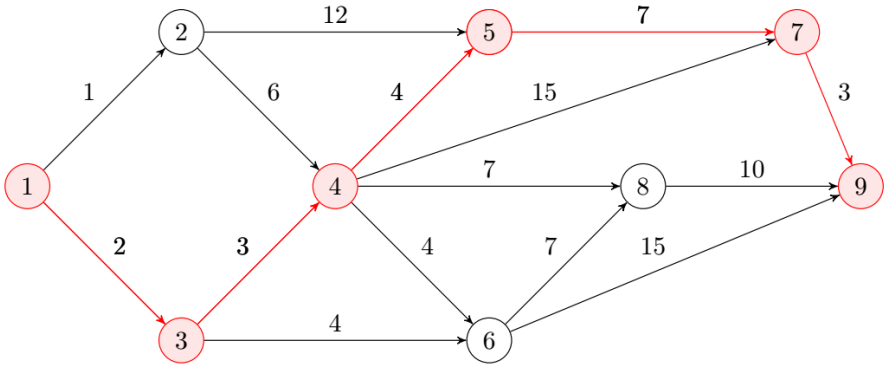
pri čemu je t_{ij} vreme potrebno da se pređe put od čvora i do njemu susednog čvora j.

$f_9 = 0$
$f_8 = 10 + f_9 = 10, \quad k_8 = 9$
$f_7 = 3 + f_9 = 3, \quad k_7 = 9$
$f_6 = \min\begin{cases} 7+f_8 \\ 15+f_9 \end{cases} = \min\begin{cases} 7+10 \\ 15+0 \end{cases} = \min\{17,15\} = 15, \quad k_6 = 9$
$f_5 = 7 + f_7 = 7 + 3 = 10, \quad k_5 = 7$
$f_4 = \min\begin{cases} 4+f_5 \\ 4+f_6 \\ 15+f_7 \\ 7+f_8 \end{cases} = \min\begin{cases} 4+10 \\ 4+15 \\ 15+3 \\ 7+10 \end{cases} = \min\{14,19,18,17\} = 14, \quad k_4 = 5$
$f_3 = \min\begin{cases} 3+f_4 \\ 4+f_6 \end{cases} = \min\begin{cases} 3+14 \\ 4+15 \end{cases} = \min\{17,19\} = 17, \quad k_3 = 4$
$f_2 = \min\begin{cases} 6+f_4 \\ 12+f_5 \end{cases} = \min\begin{cases} 6+14 \\ 12+10 \end{cases} = \min\{20,22\} = 20, \quad k_2 = 4$
$f_1 = \min\begin{cases} 1+f_2 \\ 2+f_3 \end{cases} = \min\begin{cases} 1+20 \\ 2+17 \end{cases} = \min\{21,19\} = 19, \quad k_1 = 3.$

etapa i	prethodno stanje r_{i-1}	upravljanje u_i	skup dopustivih upravljanja $U_i(r_{i-1})$	trenutno stanje $r_i = u_i$
1	1	3	$\{2, 3\}$	3
2	3	4	$\{4, 6\}$	4
3	4	5	$\{5, 6, 7, 8\}$	5
4	5	7	$\{7\}$	7
5	7	9	$\{9\}$	9

Tabela 4: Višeetapni proces formiranja puta u grafu.

$t_0 = r_0 = 1$
$t_1, u_1 \in U_1(r_0) = \{2, 3\}, u_1 = r_1 = 3$
$t_2, u_2 \in U_2(r_1) = \{4, 6\}, u_2 = r_2 = 4$
$t_3, u_3 \in U_3(r_2) = \{5, 6, 7, 8\}, u_3 = r_3 = 5$
$t_4, u_4 \in U_4(r_3) = \{7\}, u_4 = r_4 = 7$
$t_5, u_5 \in U_5(r_4) = \{9\}, u_5 = r_5 = 9$



Slika 5: Najkraći put u grafu je $1 \to 3 \to 4 \to 5 \to 7 \to 9$ i vreme koje je potrebno da se taj put pređe iznosi 19 minuta.

2.2 Problem trgovačkog putnika

Švajcarski matemetičar i fizičar Leonard Ojler (Leonhard Euler, 1707-1783) je prvi proučavao problem sličan problemu trgovačkog putnika, takozvani problem skakačevog obilaska šahovske ploče (Knight's Tour Problem): Da li je moguće skakanjem obići sva polja šahovske table tako da se svako polje obiđe tačno jedanput? Rešenje za ovaj problem je objavio 1759. godine.

Poznato je i da su Vilijam Hamilton (William Rowan Hamilton, 1805-1865) i Tomas Kirkman (Thomas Kirkman, 1806-1895) proučavali probleme koji se svode na problem trgovačkog putnika početkom XX veka. Hamilton je 1856. godine predstavio sledeći problem: Trgovački putnik treba da obiđe određen broj gradova i da se vrati na mesto polaska, tako da u toku putovanja obiđe svaki grad tačno jedanput.

Ipak opšta forma ovog problema pojavljuje tek 30−ih godina XX veka, dok je pojam "trgovački putnik" prvi put iskorišćen 1932. godine.

Ovo je jedan od najstarijih problema kombinatorne optimizacije i jedan od najintenzivnije proučavanih matematičkih problema. U teoriji, kao i u praksi postoji veliki broj problema koji se svode na problem trgovačkog putnika.

Trgovački putnik ima spisak gradova koje treba da poseti i takođe zna kolika je njihova međusobna udaljenost. U obavezi je da poseti svaki grad tačno jednom i da se vrati u grad polaska. Postavlja se pitanje kojim redosledom treba obilaziti gradove, tako da se putuje najoptimalnijom mogućom rutom, što znači da dužina puta treba da bude minimalna.

Primer 2.5. [1] Trgovački putnik mora da poseti svaki od četiri grada (Njujork, Majami, Dalas i Čikago) i između svakog para gradova postoji put dužine date u Tabeli 5. Polazeći od Njujorka, naći put minimalne dužine, tako da putnik poseti svaki grad tačno jednom i vrati se u Njujork.

	Gradovi			
	Njujork	Majami	Dalas	Čikago
1 Njujork	-	1334	1559	809
2 Majami	1334	-	1343	1397
3 Dalas	1559	1343	-	921
4 Čikago	809	1397	921	-

Tabela 5: Dužina puta između gradova izražena u miljama.

Rešenje:

Definicija 2.2. Hamiltonov put je put koji prolazi kroz sve čvorove grafa tačno jednom. Put koji se završava u istom čvoru a kroz sve ostale čvorove prolazi tačno jednom naziva se zatvoren Hamiltonov put (Hamiltonov put formira konturu koja sadrži sve

čvorove grafa pa se umesto pojma Hamiltonov put češće koristi pojam Hamiltonove konture). Graf koji ima Hamiltonovu konturu se naziva Hamiltonovim grafom.

Dakle, potrebno je odrediti najkraći Hamiltonov put (konturu) u potpunom težinskom grafu.

Problem se rašava korišćenjem metoda DP, odnosno rešavanjem problema "unazad." Može se primetiti da kada putniku ostane jedan grad da poseti, njegov problem je trivijalan - jednostavno ide sa trenutne lokacije u Njujork.

Neka su etape indeksirane brojem gradova koje je putnik već posetio. Na svakoj etapi putnik donosi odluku koji grad sledeće treba da poseti i pri tom je poznata trenutna lokacija, kao i spisak gradova koje je prethodno posetio. Ako je prethodno posetio $t-1$ grad, i skup S predstavlja skup posećenih gradova, pri čemu je grad i poslednji posećen, onda se $f_t(i,S)$ definiše kao minimalno rastojanje koje putnik treba da pređe na etapi t . Neka je c_{ij} rastojanje između gradova i i j.

Etapa 4: Na ovoj etapi mora važiti $S=\{2,3,4\}$, i jedina moguća stanja su $(2,\{2,3,4\}),(3,\{2,3,4\}),(4,\{2,3,4\})$. Iz trenutne lokacije, putnik ide u Njujork. Odatle sledi

$$\begin{aligned} f_4(2,\{2,3,4\}) &= 1334, \ (\text{grad } 2 \to \text{grad } 1)\\ f_4(3,\{2,3,4\}) &= 1559, \ (\text{grad } 3 \to \text{grad } 1)\\ f_4(4,\{2,3,4\}) &= 809, \quad (\text{grad } 4 \to \text{grad } 1) \end{aligned}$$

Etapa 3: Ako je putnik u gradu i i putuje u grad j on prelazi dužinu c_{ij}. Potom prelazi u etapu 4, na toj etapi važi da je poslednje posetio grad j, i do tada je posetio gradove $S\cup\{j\}$, pa je dužina ostatka puta koji mora da pređe jednaka $f_4(j,S\cup\{j\})$. Odavde se dobija sledeća formula

$$f_3(i,S)=\min_{\substack{j\notin S\\ j\neq 1}}\{c_{ij}+f_4(j,S\cup\{j\})\}. \tag{2.1}$$

Putnik je prethodno posetio gradove $\{2,3\},\{2,4\}$ ili $\{4,3\}$ i treba na ovoj etapi da poseti neki od gradova koje do sada već nije posetio, a da to ne bude grad 1. Na osnovu formule (2.1) dobija se $f_3(\cdot)$ za sva moguća stanja na trećoj etapi.

$$\begin{aligned} f_3(2,\{2,3\}) &= c_{24}+f_4(4,\{2,3,4\}) = 1397+809 = 2206, && (\text{grad } 2 \to \text{grad } 4)\\ f_3(3,\{2,3\}) &= c_{34}+f_4(4,\{2,3,4\}) = 921+809 = 1730, && (\text{grad } 3 \to \text{grad } 4)\\ f_3(2,\{2,4\}) &= c_{23}+f_4(3,\{2,3,4\}) = 1343+1559 = 2902, && (\text{grad } 2 \to \text{grad } 3)\\ f_3(4,\{2,4\}) &= c_{43}+f_4(3,\{2,3,4\}) = 921+1559 = 2480, && (\text{grad } 4 \to \text{grad } 3)\\ f_3(3,\{3,4\}) &= c_{32}+f_4(2,\{2,3,4\}) = 1343+1334 = 2677, && (\text{grad } 3 \to \text{grad } 2)\\ f_3(4,\{3,4\}) &= c_{42}+f_4(2,\{2,3,4\}) = 1397+1334 = 2731, && (\text{grad } 4 \to \text{grad } 2) \end{aligned}$$

Važi

$$f_t(i,S)=\min_{\substack{j\notin S\\ j\neq 1}}\{c_{ij}+f_{t+1}(j,S\cup\{j\})\},\ t=1,2,3, \tag{2.2}$$

jer ako je putnik sada u gradu i, posetio je gradove iz skupa S i sledeći grad koji posećuje je grad j, onda on mora da pređe rastojanje c_{ij}. Ostatak njegovog puta počinje u gradu j, i on je posetio gradove $S\cup\{j\}$. To znači da je dužina ostatka njegovog puta jednaka $f_{t+1}(j,S\cup\{j\})$, pa jasno važi (2.2).

Etapa 2: Poznato je da je do sada posećen samo jedan grad, pa su moguća stanja na ovoj etapi: $(2,\{2\}),(3,\{3\}),(4,\{4\})$. Na osnovu formule (2.2) dobija se

$$f_2(2,\{2\})=\min\begin{cases}c_{23}+f_3(3,\{2,3\})=1343+1730=3073\\ (\text{grad } 2\to\text{grad } 3)\\ c_{24}+f_3(4,\{2,4\})=1397+2480=3877\\ (\text{grad } 2\to\text{grad } 4)\end{cases}=3073.$$

$$f_2(3,\{3\})=\min\begin{cases}c_{32}+f_3(2,\{2,3\})=1343+2206=3549\\ (\text{grad } 3\to\text{grad } 2)\\ c_{34}+f_3(4,\{3,4\})=921+2731=3652\\ (\text{grad } 3\to\text{grad } 4)\end{cases}=3549.$$

$$f_2(4,\{4\})=\min\begin{cases}c_{42}+f_3(2,\{2,4\})=1397+2902=4299\\ (\text{grad } 4\to\text{grad } 2)\\ c_{43}+f_3(4,\{3,4\})=921+2677=3598\\ (\text{grad } 4\to\text{grad } 3)\end{cases}=3598.$$

Etapa 1: Na ovoj etapi, putnik još uvek nije posetio nijedan grad, i nalazi se u Njujorku (koji je njegova polazna i krajnja tačka). Na osnovu formule (2.2) dobija se

$$f_1(1,\{\cdot\})=\min\begin{cases}c_{12}+f_2\,(2,\{2\})=1334+3073=4407\\ (\text{grad } 1\to\text{grad } 2)\\ c_{13}+f_2\,(3,\{3\})=1559+3549=5108\\ (\text{grad } 1\to\text{grad } 3)\\ c_{14}+f_2\,(4,\{4\})=809+3598=4407\\ (\text{grad } 1\to\text{grad } 4)\end{cases}=4407.$$

Od grada 1 (Njujork) putnik prvo može da ide u grad 2 (Majami) ili 4 (Čikago). Pretpostavimo da je izabrao Čikago. Sledeći grad koji posećuje je onaj za koji se minimizira vrednost $f_2(4, \{4\})$ i to je grad 3 (Dalas). Potom odlazi u grad za koji se dostiže $f_3(3, \{3, 4\})$ i to je grad 2 (Majami). Na kraju, naravno mora da poseti grad 1 odnosno Njujork.

Na ovaj način se dobija optimalan put: Njujork → Čikago → Dalas → Majami → Njujork i dužina tog puta iznosi $f_1(1, \{\cdot\}) = 4407$ milja.

Ukoliko se pretpostavi da je putnik prvo odabrao Majami, analogno se dobija da je optimalni put: Njujork → Majami → Dalas → Čikago → Njujork. Ovaj optimalni put je suprotan prethodno dobijenom putu.

Napomena 1. U simetričnim mrežama se uvek dobija paran broj najkraćih Hamiltonovih kontura. Razlog je jasan: jednu konturu je moguće obići u oba smera.

Napomena 2. Problem se može proširiti dodavanjem različitih ograničenja. Na primer, postoji i vremenski zavisan problem trgovačkog putnika. On uzima u obzir i vreme koje je potrebno da se put pređe.

Prethodno opisan problem 2.5 predstavlja simetrični problem trgovačkog putnika, jer je dužina puta između gradova i i j ista kao dužina puta između gradova j i i. Ukoliko u grafu dužina puta između gradova i i j zavisi od smera obilaska, onda je reč o asimetričnom problemu.

2.2.1 Rešenje problema trgovačkog putnika pomoću programskog paketa Matlab

```
% INPUT:  s - vektor koji odgovara spisku posecenih
%             gradova: i-ta koordinata je 0 ako grad
%             nije posecen, u suprotnom je 1
%         lok - trenutna lokacija trgovackog putnika
% OUTPUT: d_opt - duzina optimalnog puta

function d_opt = tsp(s,lok)

% Matrica D sadrzi podatke o rastojanju izmedju gradova
D=[0 1334 1559 809; 1334 0 1343 1397; 1559 1343 0 921; 809
    1397 921 0];
[m,n]=size(D);
% Proverava da li su dimenzije vektora i matrice jednake
if(length(s)~=m ||  m~=n)
error('Pogresan unos!')
else
poseceni_svi = ones(length(D),1); % vektor jedinica
```

```
d_opt=sum(sum(D)); % na pocetku se inicijalizuje d_opt
d=0;
if all(s==poseceni_svi) % Ako su svi gradovi poseceni
  d_opt=D(lok,1);    % putnik se vraca u grad 1
else
for grad=1:length(D)
  if s(grad)==0 % grad nije posecen
    s1=s;
    s1(grad)=1;
    d=D(lok,grad) + tsp (s1,grad);
    d_opt=min(d_opt, d);
  endif
endfor
endif
end
```

Za $D = \begin{bmatrix} 0 & 1334 & 1559 & 809 \\ 1334 & 0 & 1397 & 1397 \\ 1559 & 1343 & 0 & 921 \\ 809 & 1397 & 921 & 0 \end{bmatrix}$, $s = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ i lok $= 1$, izvršavanjem kôda dobija se isti rezultat

$$\boxed{d_{\text{opt}} = 4407}.$$

3 Problem množenja niza matrica

3.1 Matrice

Definicija 3.1. [10] Neka su $m, n \in \mathbb{N}$. Matrica formata $m \times n$ nad poljem F je svako preslikavanje $\{1, 2, ..., m\} \times \{1, 2, ..., n\} \to F$. Slike a_{ij} parova $(i, j),\ i = 1, ..., m,\ j = 1, ..., n$ se nazivaju elementima matrice. Matricu formata $m \times n$ prikazujemo u obliku pravougaone tablice koja ima m vrsta i n kolona

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Navedena matrica se skraćeno označava sa $[a_{ij}]_{m\times n}$ ili $[a_{ij}]$.

Matrice nad poljem realnih brojeva nazivamo realnim matricama.

Niz $(a_{i1}, ..., a_{in})$ nazivamo $i-$ta vrsta matrice, a niz $(a_{1j}, ..., a_{mj})$ $j-$ta kolona matrice.

Matrica formata $n \times n$ naziva se kvadratna matrica reda n.

Kvadratna matrica je gornja (donja) trougaona ako je $a_{ij} = 0$ za svako $i > j (i < j)$.

Kvadratna matrica je dijagonalna ako je $a_{ij} = 0$ za svako $i \neq j$. Dijagonalna matrica čiji su svi elementi na dijagonali jednaki naziva se skalarna matrica.

Dijagonalnu matricu A čija je glavna dijagonala $(a_{11}, ..., a_{nn})$ označava se sa $A = \text{diag}(a_{11}, ..., a_{nn})$.

Dijagonalna matrica reda n čiji su svi dijagonalni elementi jednaki 1 naziva se jedinična matrica reda n i označava se sa E_n ili samo sa E.

Matrica formata $m \times n$ čiji su svi elementi jednaki 0 naziva se nula matrica i označava se sa $O_{m\times n}$ ili samo sa O.

Definicija 3.2. [10] Neka su $A = [a_{ij}]_{m\times n}$ i $B = [b_{ij}]_{n\times p}$. Proizvod matrica A i B je matrica $C = [c_{ij}]_{m\times p}$ takva da je

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj},\ \ i = 1, 2, ..., m,\ j = 1, 2, ..., p.$$

Teorema 3.1. [10] *Asocijativnost množenja matrica.*
Za svako $A = [a_{ij}]_{m\times n}$, $B = [b_{ij}]_{n\times p}$ i $C = [c_{ij}]_{p\times q}$ važi

$$(AB)C = A(BC).$$

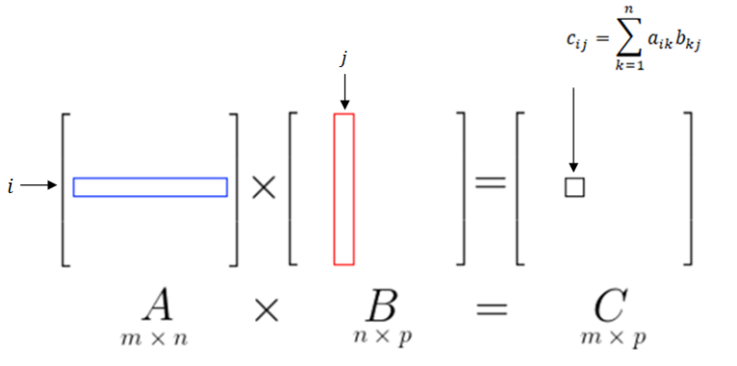
Napomena: Množenje matrica nije komutativno ($AB \neq BA$).

3.2 Formulacija problema

Neka je dat niz matrica. Odrediti najefikasniji način na koji se date matrice mogu pomnožiti.

Množenje matrica je asocijativno (na osnovu Teoreme 3.1), što znači da se proizvod može izračunati na više različitih načina i pritom je krajnji rezultat uvek isti, međutim redosled množenja (odnosno raspored zagrada) utiče na brzinu izvršenja tog množenja.

Prilikom računanja proizvoda dve matrice dimenzija $m \times n$ i $n \times p$ potrebno je izvršiti mnp operacija.



Slika 6: Množenje matrica.

Primer 3.2. [7] Neka su date matrice A, B i C dimenzija 10×100, 100×5 i 5×50, respektivno. Množenje matrica je asocijativno, pa važi $(AB)C = A(BC)$.

$$AB \to 10 \cdot 100 \cdot 5 = 5000 \text{ operacija} \Rightarrow AB \text{ je formata } 10 \times 5$$
$$(AB)C \to 10 \cdot 5 \cdot 50 = 2500 \text{ operacija} \Rightarrow (AB)C \text{ je formata } 10 \times 50.$$
$$\text{Ukupno } 7500 \text{ operacija.}$$

$$BC \to 100 \cdot 5 \cdot 50 = 25000 \text{ operacija} \Rightarrow BC \text{ je formata } 100 \times 50$$
$$A(BC) \to 10 \cdot 100 \cdot 50 = 50000 \text{ operacija} \Rightarrow A(BC) \text{ je formata } 10 \times 50.$$
$$\text{Ukupno } 75000 \text{ operacija.}$$

Ovaj primer pokazuje prednosti računanja optimalnog redosleda. Jasno je da drugačiji redosled množenja dovodi do velikih razlika u vremenu potrebnom za izvršenje množenja. U ovom primeru prvi način računanja proizvoda matrica je čak 10 puta brži.

Prvi način za rešavanje problema je direktno (pretraživanjem svih mogućnosti) - *brute force.* Neka je dat niz od n matrica. Tada postoji $n - 1$ mesto u nizu na kome

se niz može podeliti. Nakon podele dobijaju se dva podniza - jedan dužine k a drugi dužine $n-k$. Označimo sa L broj načina na koji se matrice iz podniza dužine k mogu pomnožiti i sa R broj načina na koji se mogu pomnožiti matrice iz preostalog niza. Kako su ova dva načina nezavisna jedan od drugog, onda je ukupan broj jednak $L \cdot R$.

Neka je $P(n)$ broj načina na koji se može pomnožiti n matrica. Na osnovu prethodnog zaključka sledi

$$P(n) = \begin{cases} 1, & n = 1, \\ \sum_{k=1}^{n-1} P(k)P(n-k), & n > 1. \end{cases}$$

Ovakva rekurzija je povezana sa Katalanovim brojevima [1], odnosno važi

$$P(n) = C(n-1), \text{ gde je } C(n)\ n-\text{ti Katalanov broj.}$$

Dakle, proveravanje svih mogućnosti nije praktično ukoliko je n veliko (jer je $C(n-1)$ veliki broj za velike vrednosti broja n.)

Bolji način - dinamičko programiranje.

Neka je $A = A_1 \cdot A_2 \cdots A_k \cdot A_{k+1} \cdot A_{k+2} \cdots A_n$, i neka su dimenzije matrica A_i, $i = 1, ..., n,\ p_0 \times p_1,\ p_1 \times p_2, ..., p_{n-1} \times p_n$ redom.
Problem se rešava u nekoliko koraka.

1. Rastaviti problem na više manjih potproblema.

   $$A = \boxed{A_1 \cdot A_2 \cdots A_k} \cdot \boxed{A_{k+1} \cdot A_{k+2} \cdots A_n},\ 1 \leqslant k < n.$$

   Proizvod se podeli na dva dela $A_1 \cdot A_2 \cdots A_k$ i $A_{k+1} \cdot A_{k+2} \cdots A_n$, za $1 \leqslant k < n$. Potom se rešavaju dva dobijena potproblema.

2. Definisati rekurentnu formulu.

   Neka je $M[i, j]$ minimalni "trošak" prilikom računanja proizvoda matrica od $i-$te do $j-$te matrice.

   Rekurzivna formula je

   $$M[i, j] = M[i, k] + M[k + 1, j] + p_{i-1}p_k p_j, \text{ za } 1 \leqslant k < n.$$
   $$M[i, i] = 0, \text{ za } i = 1, 2, ..., n.$$
   $$M[1, n] = ?$$

[1] Katalanovi brojevi predstavljaju niz prirodnih brojeva koji su značajni u kombinatorici i zadovoljavaju rekurziju $C_0 = 1,\ C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}$, za $n \geqslant 0$. Mogu se definisati i preko binomnih koeficijenata $C_n = \frac{1}{n+1}\binom{2n}{n}$.

Dakle, dobija se sledeća formula

$$M[i,j] = \begin{cases} 0, & i = j \\ \min\limits_{1 \leqslant k < j} \{M[i,k] + M[k+1,j] + p_{i-1}p_k p_j\}, & i \neq j. \end{cases} \tag{3.1}$$

Primer 3.3. [14] Neka su date matrice A_1, A_2, A_3, A_4, A_5, čije su dimenzije redom $4 \times 10,\ 10 \times 3,\ 3 \times 12,\ 12 \times 20,\ 20 \times 7$. Odrediti optimalni redosled množenja matrica.

Rešenje:

i \ j	1	2	3	4	5
1	0	120	264	1080	1344
2		0	360	1320	1350
3			0	720	1140
4				0	1680
5					0

U datoj tabeli se u preseku $i-$te vrste i $j-$te kolone upisuju vrednosti $M[i,j]$ koje se računaju na osnovu (3.1).

$$M[1,2] = \min_{1 \leqslant k < 2} \{M[1,k] + M[k+1,2] + p_0 p_k p_2\} = M[1,1] + M[1+1,2] + p_0 p_1 p_2 =$$
$$= 0 + 0 + 4 \cdot 10 \cdot 3 = 120.$$

$$M[2,3] = \min_{2 \leqslant k < 3} \{M[2,k] + M[k+1,3] + p_1 p_k p_3\} = M[2,2] + M[2+1,3] + p_1 p_2 p_3 =$$
$$= 0 + 0 + 10 \cdot 3 \cdot 12 = 360.$$

$$M[3,4] = \min_{3 \leqslant k < 4} \{M[3,k] + M[k+1,4] + p_2 p_k p_4\} = M[3,3] + M[3+1,4] + p_2 p_3 p_4 =$$
$$= 0 + 0 + 3 \cdot 12 \cdot 20 = 720.$$

$$M[4,5] = \min_{4 \leqslant k < 5} \{M[4,k] + M[k+1,5] + p_3 p_k p_5\} = M[4,4] + M[4+1,5] + p_3 p_4 p_5 =$$
$$= 0 + 0 + 12 \cdot 20 \cdot 7 = 1680.$$

$$M[1,3] = \min_{1\leqslant k<3}\{M[1,k]+M[k+1,3]+p_0p_kp_3\} = \min\begin{cases} M[1,1]+M[2,3]+p_0p_1p_3 \\ M[1,2]+M[3,3]+p_0p_2p_3 \end{cases}$$
$$= \min\begin{cases} 0+360+4\cdot 10\cdot 12, & k=1 \\ 120+0+4\cdot 3\cdot 12, & k=2 \end{cases} = \min\begin{cases} 840, & k=1 \\ 264, & k=2 \end{cases} = 264,\ \boxed{k=2}$$

$$M[2,4] = \min_{2\leqslant k<4}\{M[2,k]+M[k+1,4]+p_1p_kp_4\} = \min\begin{cases} M[2,2]+M[3,4]+p_1p_2p_4 \\ M[2,3]+M[4,4]+p_1p_3p_4 \end{cases}$$
$$= \min\begin{cases} 0+720+10\cdot 3\cdot 20, & k=2 \\ 360+0+10\cdot 12\cdot 20, & k=3 \end{cases} = \min\begin{cases} 1320, & k=2 \\ 2760, & k=3 \end{cases} = 1320,\ \boxed{k=2}$$

$$M[3,5] = \min_{3\leqslant k<5}\{M[3,k]+M[k+1,5]+p_2p_kp_5\} = \min\begin{cases} M[3,3]+M[4,5]+p_2p_3p_5 \\ M[3,4]+M[5,5]+p_2p_4p_5 \end{cases}$$
$$= \min\begin{cases} 0+1680+3\cdot 12\cdot 7, & k=3 \\ 720+0+3\cdot 20\cdot 7, & k=4 \end{cases} = \min\begin{cases} 1932, & k=3 \\ 1140, & k=4 \end{cases} = 1140,\ \boxed{k=4}$$

$$M[1,4] = \min_{1\leqslant k<4}\{M[1,k]+M[k+1,4]+p_0p_kp_4\} = \min\begin{cases} M[1,1]+M[2,4]+p_0p_1p_4 \\ M[1,2]+M[3,4]+p_0p_2p_4 \\ M[1,3]+M[4,4]+p_0p_3p_4 \end{cases}$$
$$= \min\begin{cases} 0+1320+4\cdot 10\cdot 20, & k=1 \\ 120+720+4\cdot 3\cdot 20, & k=2 \\ 264+0+4\cdot 12\cdot 20, & k=3 \end{cases} = \min\begin{cases} 2120, & k=1 \\ 1080, & k=2 \\ 1224. & k=3 \end{cases} = 1080,\ \boxed{k=2}$$

$$M[2,5] = \min_{2\leqslant k<5}\{M[2,k]+M[k+1,5]+p_1p_kp_5\} = \min\begin{cases} M[2,2]+M[3,5]+p_1p_2p_5 \\ M[2,3]+M[4,5]+p_1p_3p_5 \\ M[2,4]+M[5,5]+p_1p_4p_5 \end{cases}$$
$$= \min\begin{cases} 0+1140+10\cdot 3\cdot 7, & k=2 \\ 360+1680+10\cdot 12\cdot 7, & k=3 \\ 1320+0+10\cdot 20\cdot 7, & k=4 \end{cases} = \min\begin{cases} 1350, & k=2 \\ 2880, & k=3 \\ 2720, & k=4 \end{cases} = 1350,\ \boxed{k=2}$$

$$M[1,5] = \min_{1\leqslant k<5}\{M[1,k]+M[k+1,5]+p_0p_kp_5\} = \min\begin{cases} M[1,1]+M[2,5]+p_0p_1p_5 \\ M[1,2]+M[3,5]+p_0p_2p_5 \\ M[1,3]+M[4,5]+p_0p_3p_5 \\ M[1,4]+M[5,5]+p_0p_4p_5 \end{cases}$$

$$= \min\begin{cases} 0+1350+4\cdot 10\cdot 7, & k=1 \\ 120+1140+4\cdot 3\cdot 7, & k=2 \\ 264+1680+4\cdot 12\cdot 7, & k=3 \\ 1080+0+4\cdot 20\cdot 7, & k=4 \end{cases} = \min\begin{cases} 1630, & k=1 \\ 1344, & k=2 \\ 2280, & k=3 \\ 1640, & k=4 \end{cases} = 1344, \boxed{k=2}$$

$M[1,5] = 1344$, odnosno za optimalno množenje matrica potrebno je izvršiti 1344 operacija množenja. Potrebno je još odrediti kojim redosledom se množe matrice, odnosno gde treba staviti zagrade.

$$k=2 : M[1,5] = M[1,2]+M[3,5]+p_0p_2p_5 \Longrightarrow A = (A_1A_2)(A_3A_4A_5).$$
$$k=4 : M[3,5] = M[3,4]+M[5,5]+p_2p_3p_5 \Longrightarrow A = (A_1A_2)((A_3A_4)A_5).$$

Rešenje: $A = (A_1A_2)((A_3A_4)A_5)$.

3.3 Rešavanje problema množenja niza matrica pomoću programskog paketa Matlab

```
% INPUT:      p - vektor dimenzija matrica
% OUTPUT:     r - optimalno resenje (najmalnji broj
%                 operacija koje je potrebno izvrsiti
%             s - redosled mnozenja matrica

function [r,s] = optim(p)
    n = length(p)-1; % Broj zagrada koje postavljamo
    u = zeros(n,n);
    v = ones(n,n)*inf;
    u(:,1) = -1;
    v(:,1) = 0;
    for j = 2:n
        for i = 1:n-j+1
            for k = 1:j-1
```

```
                    c = v(i,k)+v(i+k,j-k)+p(i)*p(i+k)*p(i+j);
                    if c<v(i,j)
                        u(i,j) = k;
                        v(i,j) = c;
                    end
                end
            end
        end
        r = v(1,n);
        s = aux(u,1,n);
end

% Odredjivanje optimalnog redosleda mnozenja matrica
function s = aux(u,i,j)
        k = u(i,j);
        if k<0
            s = sprintf("%d",i);
        else
            s = sprintf("(%s*%s)",aux(u,i,k),aux(u,i+k,j-k));
        end
end
```

Primer 3.3. Prilikom izvršavanja kôda za $p = \begin{bmatrix} 4 & 10 & 3 & 12 & 20 & 7 \end{bmatrix}$ dobija se sledeći rezultat
r = 1344
s = ((1*2)*((3*4)*5)).
Dakle, dobija se sledeće optimalno množenje matrica $(A_1A_2)(((A_3A_4)A_5))$ i potrebno je izvršiti 1344 operacija množenja.

Primer 3.4. [15] Odrediti optimalni redosled množenja matrica ako su dimenzije matrica date u narednoj tabeli

matrica	A_1	A_2	A_3	A_4	A_5	A_6
dimenzije	30×35	35×15	15×5	5×10	10×20	20×25

Rešenje: Za $p = \begin{bmatrix} 30 & 35 & 15 & 5 & 10 & 20 & 25 \end{bmatrix}$ se izvršavanjem kôda dobija rezultat
r = 15125
s = ((1*(2*3))*((4*5)*6)).
Može se zaključiti da je za optimalno množenje matrica potrebno izvršiti 15125 operacija množenja i pri tom je redosled množenja sledeći $(A_1(A_2A_3))((A_4A_5)A_6)$.

4 Upoređivanje nizova i edit distanca

Neformalno, string je niz simbola iz nekog alfabeta, pri čemu je alfabet najčešće neki konačan skup.

Definicija 4.1. [17] Ako je Σ alfabet a n prirodan broj, Σ^n označava skup svih uređenih $n-$torki $(x_1, x_2, ..., x_n)$, gde je $x_i \in \Sigma$, za sve $i \in \{1, 2, ..., n\}$.

Ova $n-$torka se može kraće zapisati sa x, i njena dužina je $|x|$. Koristi se i zapis $x_1 x_2 \cdots x_n$.

Definicija 4.2. [17] Ako je Σ alfabet, onda je

$$\Sigma^+ = \bigcup_{n=1}^{\infty} \Sigma^n$$

skup svih nepraznih reči nad alfabetom Σ.

Ovom skupu se može dodati i prazna reč koja se označava sa e.

Definicija 4.3. [17] Skup svih reči nad alfabetom Σ je skup

$$\Sigma^* = \Sigma^+ \cup \{e\}.$$

Podstring počev od pozicije l, do pozicije r ne uključujući r, nekog stringa x, gde važi $0 \leq l \leq r \leq |x|$ je string $x_l x_{l+1} \cdots x_{r-1}$.

Podstringovi stringa x kod kojih je $l = 0$ se nazivaju *prefiksi,* dok se oni kod kojih je $r = |x|$ nazivaju *sufiksi.* Ukoliko je $l = r$ u pitanju je prazan string.

Definicija 4.4. [2] Levenštajnovo rastojanje ili edit distanca je broj minimalnih operacija neophodnih za transformaciju jednog stringa u drugi, pri čemu su dozvoljene sledeće operacije (tzv. edit operacije) čija je cena 1:

- umetanje znaka
- brisanje znaka
- zamena jednog znaka drugim.

Nazvana je po ruskom naučniku (Vladimir Levenshtein) koji se 1965. bavio ovim rastojanjem.

Primer 4.1. [2] Neka su data dva stringa: SNOWY i SUNNY. Odrediti rastojanje između njih.

Rešenje:
Levenštajnovo rastojanje (edit distanca) iznosi 3

$$SNOWY \xrightarrow[\text{slova } U]{\text{umetanje}} SUNOWY \xrightarrow[O \to N]{\text{zamena}} SUNNWY \xrightarrow[\text{slova } W]{\text{brisanje}} SUNNY.$$

Napomena: Može se primetiti da rešenje ne mora biti jedinstveno. Drugo moguće rešenje je

$$SNOWY \xrightarrow[N \to U]{\text{zamena}} SUOWY \xrightarrow[O \to N]{\text{zamena}} SUNWY \xrightarrow[W \to N]{\text{zamena}} SUNNY.$$

Neke primene edit distance su:

- Softveri za ispravljanje pravopisnih grešaka - spelčekeri (eng. spell checker). Kada spelčeker uoči moguću grešku u spelovanju (odnosno reč koju je korisnik uneo se ne nalazi u rečniku), smatra se da reč nije korektno uneta i izlistavaju se sve reči koje su "bliske" unetoj reči. Da bi se odredilo koliko su neke reči bliske koristi se upravo Levenštajnovo rastojanje.

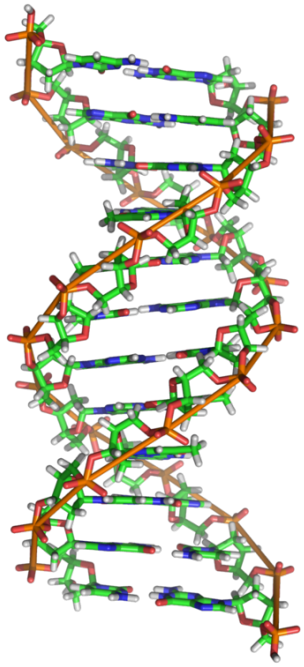
- Prepoznavanje govora.

- U molekularnoj biologiji se koristi za izučavanje DNK sekvenci. Dezoksiribonukleinska kiselina (DNK) sadrži uputstva za razvoj i pravilno funkcionisanje svih živih organizama. Zajedno sa RNK i proteinima, DNK je jedan od tri glavna tipa makromolekula koji su esencijalni za sve poznate forme života. DNK segment koji prenosi ova važna uputstva se naziva gen. Svaki se lanac DNK sastoji od nukleotida (gradivnih jedinica) kojih ima 4: adenin (A), citozin (C), guanin (G) i timin (T). Nukleotidi obrazuju dva polinukleotidna lanca koji su spiralno uvijeni. Lanci su međusobno vezani vodoničnim vezama koje se uspostavljaju između sledećih tzv. azotnih parova: A-T; C-G. Raspored ovih baza duž DNK lanca (npr. ATCGATTA) je DNK sekvenca.

  Genom je kompletan set DNK u jednom organizmu i sastavljen je od čak 3 milijarde baznih parova. Poznato je da se za svaka dva ljudska organizma DNK sekvence razlikuju za ne više od 0.1%, što znači da postoji 3 miliona pozicija na kojima se DNK sekvence razlikuju. Te razlike su od ogromne naučne i medicinske važnosti, jer se pomoću njih može predvideti podložnost ljudi određenim bolestima.

  DNK sekvenca može da se posmatra kao string sačinjen od slova A,T,G,C, što znači da se može koristiti edit distanca za merenje varijacija između DNK molekula.

  Jedan od načina da se stekne uvid u funkcionisanje nekog novootkrivenog gena jeste da se pronađu poznati geni koji su bliski novootkrivenom.



Slika 7: Struktura DNK

4.1 Određivanje edit distance pomoću DP

Neka su data dva niza (stringa) $x_1x_2\cdots x_m$ i $y_1y_2\cdots y_n$. Odrediti edit distancu između njih. Kada je reč o dinamičkom programiranju prvo pitanje koje se nameće je kako odrediti potprobleme.

Sa $E(i,j)$ je označena edit distanca između prefiksa prve reči $x_1x_2\cdots x_i$ i nekog prefiksa druge reči $y_1y_2\cdots y_j$, za $i \in \{1,2,...,m\}, j \in \{1,2,...,n\}$. Krajnji cilj je određivanje $E(m,n)$.
$E(i,0) = i$ jer je drugi string prazan string, pa je potrebno i puta izvršiti brisanje slova u prvom stringu (dužine i).
Analogno važi $E(0,j) = j$.
U nastavku se posmatra $x_1x_2\cdots x_i$ i $y_1y_2\cdots y_j$, za $i \in \{0,1,2,...,m\}, j \in \{0,1,2,...,n\}$.
Postoje dve mogućnosti:

1. Ako je $x_i = y_j$, što znači da je elemenat na $i-$tom mestu u prvom stringu jednak $j-$tom elementu u drugom i tada je

   $$E(i,j) = E(i-1,j-1)$$

   (traži se edit distanca između $x_1x_2\cdots x_{i-1}$ i $y_1y_2\cdots y_{j-1}$).

2. Ako je $x_i \neq y_j$ moguće je izvršiti jednu od tri prethodno navedenih edit operacija:

   (a) **umetanje znaka:** Na string $x_1x_2\cdots x_i$ se dodaje y_j, i dobija se string dužine $i+1$. Tada novodobijeni string i $y_1y_2\cdots y_j$ na poslednjem mestu imaju isto slovo, što znači da je

   $$E(i,j) = \underbrace{E(i,j-1)}_{\substack{\text{edit distanca}\\ \text{za } x_1\cdots x_i \text{ i}\\ y_1\cdots y_{j-1}}} + \underbrace{1}_{\substack{\text{umetanje}\\ \text{znaka}}}$$

   (b) **brisanje znaka:** Stringu $x_1x_2\cdots x_i$ se briše poslednje slovo i dobija se string $x_1x_2\cdots x_{i-1}$ dužine $i-1$. Tada je

   $$E(i,j) = \underbrace{E(i-1,j)}_{\substack{\text{edit distanca}\\ \text{za } x_1\cdots x_{i-1} \text{ i}\\ y_1\cdots y_j}} + \underbrace{1}_{\substack{\text{brisanje}\\ \text{znaka}}}$$

   (c) **zamena jednog znaka drugim:** U prvom stringu se poslednje slovo x_i zameni poslednjim slovom drugog stringa y_j. Sada ovako dobijeni stringovi imaju isto poslednje slovo, što znači da je

   $$E(i,j) = \underbrace{E(i-1,j-1)}_{\substack{\text{edit distanca}\\ \text{za } x_1\cdots x_{i-1} \text{ i}\\ y_1\cdots y_{j-1}}} + \underbrace{1}_{\substack{\text{zamena}\\ \text{znaka}}}.$$

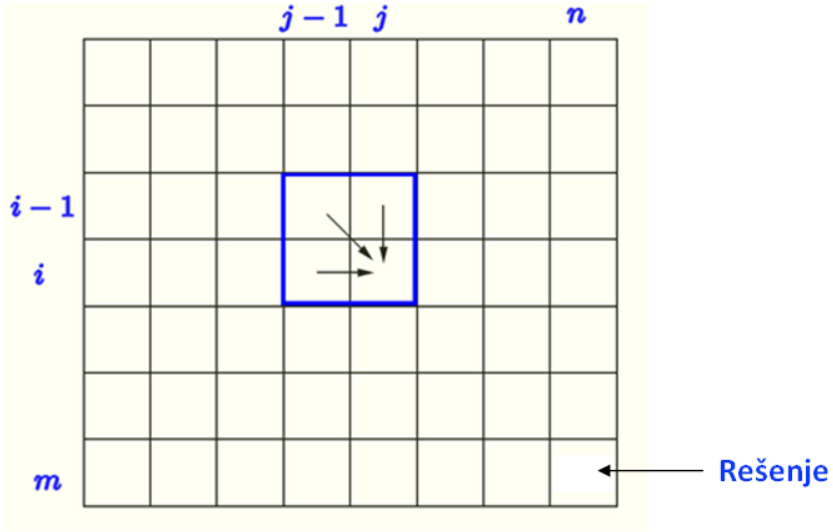
   U slučaju da je $x_i \neq y_j$ važi

   $$E(i,j) = \min \begin{cases} E(i,j-1)+1 \\ E(i-1,j)+1 \\ E(i-1,j-1)+1. \end{cases}$$

Najmanji broj edit operacija iznosi

$$E(i,j) = \min \begin{cases} E(i,j-1)+1 \\ E(i-1,j)+1 \\ E(i-1,j-1)+c(i,j) \end{cases}, \quad c(i,j) = \begin{cases} 1, & x_i \neq y_j \\ 0, & x_i = y_j. \end{cases} \tag{4.1}$$

Da bi se odredilo $E(m,n)$ popunjava se tabela dimenzije $m+1 \times n+1$, koristeći (4.1). Rešenje problema, odnosno optimalna dužina edit distance se čita iz donjeg desnog ugla tabele.



Primer 4.2. [2] Data su dva stringa $EXPONENTIAL$ i $POLYNOMIAL$. Odrediti edit distancu između njih.

Rešenje:
Ovaj problem se rešava razbijanjem na manje probleme. Posmatraju se prefiksi prvog i drugog stringa i za svaki od njih se određuje njihova edit distanca. Krajni cilj je odrediti edit distancu polazne reči, odnosno $E(11, 10)$. Jedan potproblem je dat u nastavku.

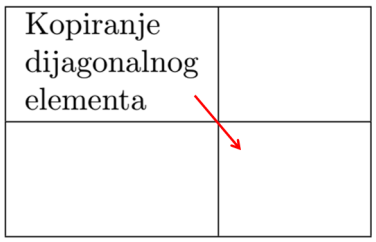
E	X	P	O	N	E	N	T	I	A	L
P	O	L	Y	N	O	M	I	A	L	

$E(7, 5)$ predstavlja edit distancu prefiksa $EXPONEN$ i $POLYN$.
Prilikom rešavanja problema popunjava se tabela u kojoj se u preseku $i-$te vrste i $j-$te kolone nalazi $E(i, j)$, i to na sledeći način:

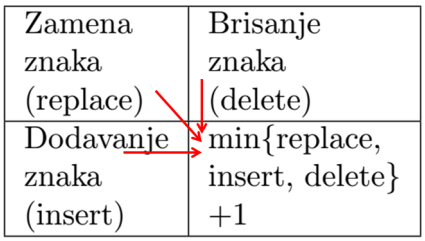
1. Prvo se popune prva vrsta i prva kolona, jer je poznato da je $E(i, 0) = i$ i $E(0, j) = j$. Oznaka $-$ u tabeli predstavlja prazan string, odnosno string dužine 0.

2. Ostali elementi tabele se popunjavaju pomoću rekurzivne formule (4.1), i to na sledeći način:

(a) Prvo se proverava da li su poslednja slova dva prefiksa jednaka. Ukoliko jesu, odgovarajuće polje se popunjava tako što se kopira njemu odgovarajuće dijagonalno polje.



(b) U suprotnom svako polje date tabele se popunjava na sledeći način



U nastavku se popunjava tabela formata 11×10.

$$E(i,0) = i, \text{ za sve } i = 1, 2, ..., m.$$
$$E(0,j) = j, \text{ za sve } j = 1, 2, ..., n.$$
$$E(1,1) = \min \begin{cases} E(1,0)+1 \\ E(0,1)+1 \\ E(0,0)+1 \end{cases} = \min \begin{cases} 1+1 \\ 1+1 \\ 0+1 \end{cases} = 1$$
$$E(1,2) = \min \begin{cases} E(1,1)+1 \\ E(0,2)+1 \\ E(0,1)+1 \end{cases} = \min \begin{cases} 1+1 \\ 2+1 \\ 1+1 \end{cases} = 2$$
$$E(1,3) = \min \begin{cases} E(1,2)+1 \\ E(0,3)+1 \\ E(0,2)+1 \end{cases} = \min \begin{cases} 2+1 \\ 3+1 \\ 2+1 \end{cases} = 3$$

$$E(1,4) = \min \begin{cases} E(1,3)+1 \\ E(0,4)+1 \\ E(0,3)+1 \end{cases} = \min \begin{cases} 3+1 \\ 4+1 \\ 3+1 \end{cases} = 4$$

$$E(1,5) = \min \begin{cases} E(1,4)+1 \\ E(0,5)+1 \\ E(0,4)+1 \end{cases} = \min \begin{cases} 4+1 \\ 5+1 \\ 4+1 \end{cases} = 5$$

$$E(1,6) = \min \begin{cases} E(1,5)+1 \\ E(0,6)+1 \\ E(0,5)+1 \end{cases} = \min \begin{cases} 5+1 \\ 6+1 \\ 5+1 \end{cases} = 6$$

$$E(1,7) = \min \begin{cases} E(1,6)+1 \\ E(0,7)+1 \\ E(0,6)+1 \end{cases} = \min \begin{cases} 6+1 \\ 7+1 \\ 6+1 \end{cases} = 7$$

$$E(1,8) = \min \begin{cases} E(1,7)+1 \\ E(0,8)+1 \\ E(0,7)+1 \end{cases} = \min \begin{cases} 7+1 \\ 8+1 \\ 7+1 \end{cases} = 8$$

$$E(1,9) = \min \begin{cases} E(1,8)+1 \\ E(0,9)+1 \\ E(0,8)+1 \end{cases} = \min \begin{cases} 8+1 \\ 9+1 \\ 8+1 \end{cases} = 9$$

$$E(1,10) = \min \begin{cases} E(1,9)+1 \\ E(0,10)+1 \\ E(0,9)+1 \end{cases} = \min \begin{cases} 9+1 \\ 10+1 \\ 9+1 \end{cases} = 10$$

$$\vdots$$

$$E(11,1) = \min \begin{cases} E(11,0)+1 \\ E(10,1)+1 \\ E(10,0)+1 \end{cases} = \min \begin{cases} 11+1 \\ 9+1 \\ 10+1 \end{cases} = 10$$

$$E(11,2) = \min \begin{cases} E(11,1)+1 \\ E(10,2)+1 \\ E(10,1)+1 \end{cases} = \min \begin{cases} 10+1 \\ 8+1 \\ 9+1 \end{cases} = 10$$

$$E(11,3) = E(10,2) = 8$$

$$E(11,4) = \min \begin{cases} E(11,3)+1 \\ E(10,4)+1 \\ E(10,3)+1 \end{cases} = \min \begin{cases} 8+1 \\ 8+1 \\ 8+1 \end{cases} = 9$$

$$E(11,5) = \min \begin{cases} E(11,4)+1 \\ E(10,5)+1 \\ E(10,4)+1 \end{cases} = \min \begin{cases} 9+1 \\ 7+1 \\ 8+1 \end{cases} = 8$$

$$E(11,6) = \min \begin{cases} E(11,5)+1 \\ E(10,6)+1 \\ E(10,5)+1 \end{cases} = \min \begin{cases} 8+1 \\ 7+1 \\ 7+1 \end{cases} = 8$$

$$E(11,7) = \min \begin{cases} E(11,6)+1 \\ E(10,7)+1 \\ E(10,6)+1 \end{cases} = \min \begin{cases} 8+1 \\ 7+1 \\ 7+1 \end{cases} = 8$$

$$E(11,8) = \min \begin{cases} E(11,7)+1 \\ E(10,8)+1 \\ E(10,7)+1 \end{cases} = \min \begin{cases} 8+1 \\ 7+1 \\ 7+1 \end{cases} = 8$$

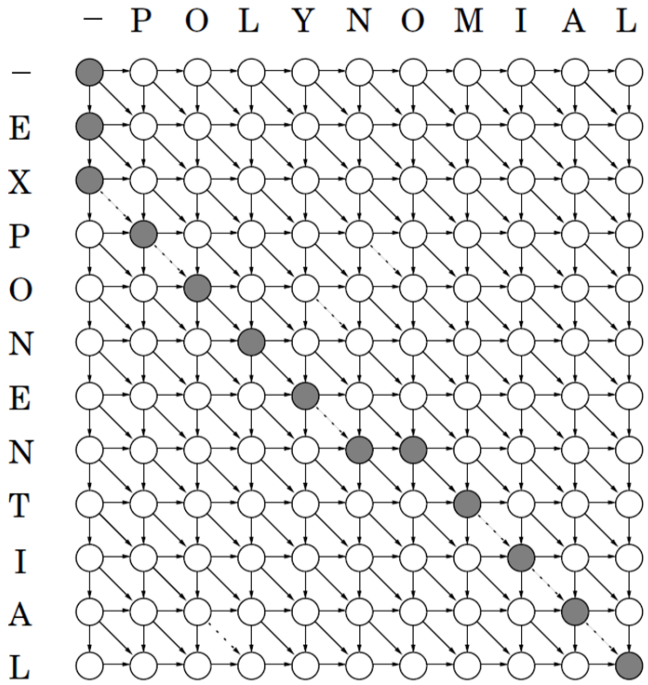
$$E(11,9) = \min \begin{cases} E(11,8)+1 \\ E(10,9)+1 \\ E(10,8)+1 \end{cases} = \min \begin{cases} 8+1 \\ 6+1 \\ 7+1 \end{cases} = 7$$

$$E(11,10) = E(10,9) = 6$$

	–	P	O	L	Y	N	O	M	I	A	L
–	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	7	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

$E(11, 10) = 6$ što znači da je potrebno izvršiti 6 edit operacija kako bi se string EXPONENTIAL transformisao POLYNOMIAL. Potrebno je još odrediti koje su to operacije u pitanju. Elementi tabele (i, j) se mogu posmatrati kao čvorovi, a grane su oblika $(i-1, j) \to (i, j), (i, j-1) \to (i, j)$ i $(i-1, j-1) \to (i, j)$. Dužina svake grane je 1, osim u slučaju kada je $\{(i-1, j-1) \to (i, j) : x_i = y_j\}$, tada je dužina jednaka 0 (u tabeli je u nastavku prikazano isprekidanom linijom). Dakle, traži se najkraći put između čvorova $(0, 0)$ i (m, n). Kada se pronađe jedan najkraći put od $(0, 0)$ do (m, n), do optimalnog niza edit operacija se dolazi kada se krene od (m, n) (u donjem desnom uglu), i svako pomeranje gore predstavlja brisanje znaka, pomeranje gore dijagonalno predstavlja zamenu znaka, a pomeranje levo je dodavanje znaka.

Jedno rešenje je dato u nastavku (ranije je ustanovljeno da ukoliko rešenje postoji ono ne mora biti jedinstveno).



$$EXPONENTIAL \xrightarrow[T \to M]{\text{zamena}} EXPONENMIAL \xrightarrow[\text{slova O}]{\text{dodavanje}} EXPONENOMIAL$$
$$\xrightarrow[E \to Y]{\text{zamena}} EXPONYNOMIAL \xrightarrow[N \to L]{\text{zamena}} EXPOLYNOMIAL$$
$$\xrightarrow[\text{slova X}]{\text{brisanje}} EPOLYNOMIAL \xrightarrow[\text{slova E}]{\text{brianje}} POLYNOMIAL.$$

4.2 Rešenje problema pomoću programskog paketa Matlab

```
% INPUT: s1-string 1
%        s2-string 2
% OUTPUT: d-edit distanca

function d=edit_distance(s1,s2)

m=numel(s1);  % duzina stringa s1
n=numel(s2);  % duzina stringa s2

d=zeros(m+1,n+1);  % nula matrica
  % inicijalizacija matrice
  for i=1:m+1
    d(i,1)=i-1;
  end
  for j=1:n+1
    d(1,j)=j-1;
  end

  for j=2:n+1
    for i=2:m+1
      if s1(i-1) == s2(j-1)
        d(i,j)=d(i-1,j-1);
      else
        d(i,j)=min([ ...
          d(i-1,j) + 1, ...  % brisanje znaka
          d(i,j-1) + 1, ...  % dodavanje znaka
          d(i-1,j-1) + 1 ... % zamena dva znaka
          ]);
      end
    end
  end
  d=d(m+1,n+1);
```

Primer 4.2 Za $s1 =$ 'EXPONENTIAL' i $s2 =$ 'POLYNOMIAL' se izvršavanjem kôda dobija $d = 6$.

5 Najduži zajednički podniz

5.1 Motivacija

Date su DNK neke dve vrste. Koliko su slične?

DNK:	**DNK:**
AGCCCTAAGGGCTACCTAGCTT	GACAGCCTACAAGCGTTAGCTTG
AGCCCTAAGGGCTACCTAGCTT	GACAGCCTACAAGCGTTAGCTTG

Kada se uporedi DNK ove dve vrste, pri čemu se DNK može posmatrati kao string (odnosno kao niz slova A,T,G i C), određivanjem njihovog najdužeg zajedničkog podniza - **AGCCTAAGCTTAGCTT**, jasno je da su ove dve vrste poprilično slične.

Postoje i druge primene najdužeg zajedničkog podniza, poput softverskog inženjerstva. Dva niza mogu da potiču iz dve verzije izvornog kôda za isti program pa je nekada potrebno utvrditi koje su promene izvršene od jedne do druge verzije.

Najjednostavnije rešenje problema nalaženja najdužeg zajedničkog podniza bi podrazumevalo traženje svih mogućih podnizova dva data stringa, a zatim pronalaženje najdužeg među njima. Ako je X dužine m, a Y dužine n, postoji 2^m podnizova niza X, i za svaki od njih je potrebno $O(n)$ vremena da se proveri da li je podniz niza Y. To dovodi do toga da bi ovakvim rešavanjem problema vreme bilo eksponencijalno, odnosno $O(2^m n)$, što čini ovaj pristup izuzetno neefikasnim. Zbog toga je potrebno primeniti tehnike dinamičkog programiranja kako bi se dobio efikasniji algoritam za rešavanje ovog problema.

5.2 Formulacija problema

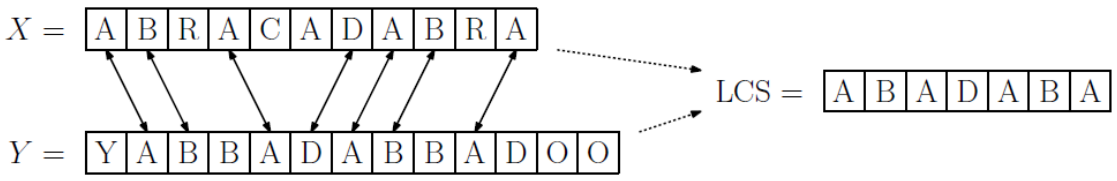
Problem najdužeg zajedničkog podniza (eng. *Longest Common Subsequence Problem* - LCS): Neka su data dva niza $X = \langle x_1, x_2, ..., x_m \rangle$ i $Y = \langle y_1, y_2, ..., y_n \rangle$. Odrediti najduži zajednički podniz ova dva niza i njegovu dužinu.

Definicija 5.1. [16] Neka su data dva niza $X = \langle x_1, x_2, ..., x_m \rangle$ i $Z = \langle z_1, z_2, ..., z_k \rangle$. Kaže se da je Z podniz niza X ako postoji striktno rastući niz indeksa $\langle i_1, i_2, ..., i_k \rangle$ $(1 \leqslant i_1 < i_2 < ... < i_k \leqslant n)$ tako da je $Z = \langle x_{i_1}, x_{i_2}, ..., x_{i_k} \rangle$.

Na primer, ako je $X = \langle ABRACADABRA \rangle$ onda je niz $Z = \langle AADAA \rangle$ jedan podniz niza X.

Definicija 5.2. [16] Za dva niza X i Y najduži zajednički podniz nizova X i Y je najduži niz Z koji je podniz oba niza, i X i Y.

Na primer, neka je $X = \langle ABRACADABRA \rangle$ i i $Y = \langle YABBADABBADOO \rangle$. Tada je naduži zajednički podniz $Z = \langle ABADABA \rangle$ (Pogledati Sliku 8).



Slika 8: Pimer najdužeg zajedničkog podniza nizova X i Y.

Napomena: Može se primetiti da najduži zajednički podniz ne mora biti jedinstven. Na primer: LCS za nizove $\langle ABC \rangle$ i $\langle BAC \rangle$ su $\langle AC \rangle$ i $\langle BC \rangle$.

5.3 LCS algoritam

U nastavku će biti pokazano da LCS problem ima optimalnu podstrukturu. Neka je dat niz $X = \langle x_1, x_2, ..., x_m \rangle$ tada se $i-$ti prefiks niza X, za svako $i = 1, 2, ..., m$, označava sa $X_i = \langle x_1, x_2, ..., x_i \rangle$.

Teorema 5.1. [15] Neka su $X = \langle x_1, x_2, ..., x_m \rangle$ i $Y = \langle y_1, y_2, ..., y_n \rangle$ dva niza i neka je $Z = \langle z_1, z_2, ..., z_k \rangle$ najduži zajednički podniz za X i Y. Tada važe sledeća tvrđenja:

1. Ako je $x_m = y_n$ onda $z_k = x_m = y_n$ i Z_{k-1} je najduži zajednički podniz za X_{m-1} i Y_{n-1}.

2. Ako je $x_m \neq y_n$ onda $z_k \neq x_m$ implicira da je Z najduži zajednički podniz nizova X_{m-1} i Y.

3. Ako je $x_m \neq y_n$ onda $z_k \neq y_n$ implicira da je Z najduži zajednički podniz nizova X i Y_{n-1}.

Dokaz:

1. Ako je $z_k \neq x_m$ onda se dodavanjem $x_m = y_n$ na niz Z dobija zajednički podniz nizova X i Y koji je dužine $k+1$ što je kontradikcija sa pretpostavkom da je Z najduži zajednički podniz. Dakle, mora važiti $x_m = y_n = z_k$.
   Dalje, prefiks Z_{k-1} je dužine $k-1$ i on je zajednički podniz nizova X_{m-1} i Y_{n-1}. Treba pokazati da je on i najduži. Pretopstavimo suprotno. Neka je W podniz nizova X_{m-1} i Y_{n-1} čija je dužina veća od $k-1$. Dodavanjem $x_m = y_n$ na niz W se dobija zajednički podniz za X i Y koji je dužine veće od k, što je kontradikcija.

2. Ako je $z_k \neq x_m$ onda je Z zajednički podniz za X_{m-1} i Y. Treba pokazati da je Z ujedno i najduži takav niz. Pretpostavimo suprotno, neka postoji W, zajednički podniz nizova X_{m-1} i Y dužine veće od k. Tada je W ujedno i zajednički podniz za X i Y, što je kontradikcija sa pretpostavkom da je Z najduži zajednički podniz za X i Y.

3. Analogno kao 2.

■

Neka su X i Y dva niza dužine m i n respektivno. $L(i,j)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa $X_i = \langle x_1, x_2, ..., x_i \rangle$ i $Y_j = \langle y_1, y_2, ..., y_j \rangle$. U nastavku se posmatraju dva slučaja:

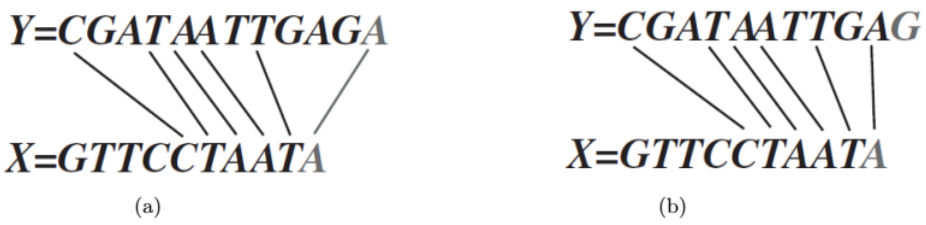
1. Slučaj: $x_i = y_j$.
Na osnovu Teoreme 5.1 sledi naredna jednakost

$$L(i,j) = L(i-1, j-1) + 1. \tag{5.1}$$

2. Slučaj: $x_i \neq y_j$. Najduži zajednički podniz se može završavati sa x_i, y_j, ili ni jednim ni drugim. Ovim razmatranjem se dobija jednakost

$$L(i,j) = \max \{L(i-1, j),\ L(i, j-1)\}. \tag{5.2}$$

$L(i,0) = L(0,j) = 0$ za $i = 1, 2, ..., m,\ j = 1, 2, ..., n$ (ukoliko je jedan string prazan onda je i LCS takođe prazan string).



Slika 9: Dva slučaja za $L(i,j)$: $(a)\ x_i = y_j$; $(b) x_i \neq y_j$.

Prilikom rešavanja problema formira se tabela L dimenzija $(m+1) \times (n+1)$. Prvi korak je inicijalizacija: $L(i,0) = L(0,j) = 0$, za $i = 1, 2, ..., m,\ j = 1, 2, ..., n$. Ostatak tabele se popunjava na osnovu rekurzivnih relacija (5.1) i (5.2). Ovakav algoritam pamti samo vrednosti $L(i,j)$, a ne i poklapanja stringova. Dakle, LCS algoritam

daje dužinu najdužeg podniza dva stringa (to je vrednost $L(m,n)$ u tabeli u donjem desnom uglu) ali ne daje sam podniz.

Najduži zajednički podniz se dobija korišćenjem principa "unazad." Jedan način je da se krene od nazad, tj. od $L(m,n)$ i da se rekonstruiše najduži podniz. Za svaku poziciju $L(i,j)$ se prvo određuje da li je $x_i = y_j$. Ukoliko je ovo ispunjeno, za sledeći element u podnizu se uzima x_i, i vrši se pomeranje na poziciju $L(i-1,j-1)$. U suprotnom se vrši pomeranje na veću od sledeće dve pozicije u tabeli: $L(i-1,j)$ ili $L(i,j-1)$. Proces se zaustavlja kada se dođe do slučaja da je $i=0$ ili $j=0$.
Napomena: Vreme potrebno za određivanje najdužeg zajedničkog podniza stringa X (dužine m) i Y (dužine n) iznosi $O(mn)$.

Primer 5.2. [16] Odrediti najduži zajednički podniz za stringove $X = BACDB$ i $Y = BDCB$.

Rešenje:
Prva faza rašavanja problema: Određivanje dužine najdužeg zajedničkog podniza.
Potrebno je formirati tabelu formata 6×5, koja se popunjava na osnovu (5.1) i (5.2).

		-	B	D	C	B
	L	0	1	2	3	4
-	0	0	0	0	0	0
B	1	0	1	1	1	1
A	2	0	1	1	1	1
C	3	0	1	1	2	2
D	4	0	1	2	2	2
B	5	0	1	2	2	3

$L(i,0) = L(0,j) = 0$ za sve $i = 1,...,5,\ j = 1,...,4.$
$L(1,1) = L(0,0) + 1 = 0 + 1 = 1$
$L(1,2) = \max\{L(1,1), L(0,2)\} = \max\{1,0\} = 1$
$L(1,3) = \max\{L(1,2), L(0,3)\} = \max\{1,0\} = 1$
$L(1,4) = L(0,3) + 1 = 0 + 1 = 1$
$L(2,1) = \max\{L(2,0), L(1,1)\} = \max\{0,1\} = 1$
$L(2,2) = \max\{L(2,1), L(1,2)\} = \max\{1,1\} = 1$
$L(2,3) = \max\{L(2,2), L(1,3)\} = \max\{1,1\} = 1$
$L(2,4) = \max\{L(2,3), L(1,4)\} = \max\{1,1\} = 1$
$L(3,1) = \max\{L(3,0), L(2,1)\} = \max\{0,1\} = 1$

$L(3,2) = \max\{L(3,1), L(2,2)\} = \max\{1,1\} = 1$
$L(3,3) = L(2,2) + 1 = 1 + 1 = 2$
$L(3,4) = \max\{L(3,3), L(2,4)\} = \max\{2,1\} = 2$
$L(4,1) = \max\{L(4,0), L(3,1)\} = \max\{0,1\} = 1$
$L(4,2) = L(3,1) + 1 = 1 + 1 = 2$
$L(4,3) = \max\{L(4,2), L(3,3)\} = \max\{2,2\} = 2$
$L(4,4) = \max\{L(4,3), L(3,4)\} = \max\{2,2\} = 2$
$L(5,1) = L(4,0) + 1 = 0 + 1 = 1$
$L(5,2) = \max\{L(5,1), L(4,2)\} = \max\{1,2\} = 2$
$L(5,3) = \max\{L(5,2), L(4,3)\} = \max\{2,2\} = 2$
$L(5,4) = L(4,3) + 1 = 2 + 1 = 3$

Dužina najdužeg zajedničkog podniza iznosi $L(5,4) = 3$.

Druga faza rešavanja problema: Određivanje najdužeg zajedničkog podniza.

		-	B	D	C	B
	L	0	1	2	3	4
-	0	0	0	0	0	0
B	1	0	**1**	1	1	1
A	2	0	1	1	1	1
C	3	0	1	1	**2**	2
D	4	0	1	2	2	2
B	5	0	1	2	2	**3**

- **Korak 1:** Posmatra se polje $L(5,4)$ u tabeli. Kako je $x_5 = y_4 = B$, LCS se zarvšava slovom B. Potom se vrši pomeranje na polje $L(4,3)$ u tabeli.
- **Korak 2:** Vrednost $L(4,3)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa BACD i BDC. Kako je $D = x_4 \neq y_3 = C$ pomeranje se vrši na veću od sledeće dve pozicije: $L(4,2) = 2$ i $L(3,3) = 2$. Kako su ove vrednosti jednake, proizvoljno se bira jedna od njih - na primer $L(3,3)$.
- **Korak 3:** Vrednost $L(3,3)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa BCA i BDC. Kako je $x_3 = y_3 = C$, LCS ovih prefiksa se završava slovom C. Potom je potrebno pomeriti se na polje $L(2,2)$ u tabeli.

- **Korak 4:** Vrednost $L(2,2)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa BA i BD. S obzirom da je $A = x_2 \neq y_2 = D$ pomeranje se vrši na veću od sledeće dve pozicije u tabeli: $L(2,1) = 1$ ili $L(1,2) = 1$. Pošto su ove dve vrednosti jednake, proizvoljno se bira jedna od njih. Na primer, neka je to $L(1,2)$.

- **Korak 5:** Vrednost $L(1,2)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa B i BD. Kako je $B = x_1 \neq y_2 = D$ pomeranje se vrši na veću od sledeće dve pozicije u tabeli: $L(1,1) = 1$ ili $L(0,2) = 0$. Dakle, potrebno je pomeriti se na polje $L(1,1)$ u tabeli.

- **Korak 6:** Vrednost $L(1,1)$ predstavlja dužinu najdužeg zajedničkog podniza prefiksa B i B. Kako su oni jednaki, B je deo najdužeg zajedničkog podniza polaznih stingova.

Dužina najdužeg zajedničkog niza iznosi 3 i jedan takav niz je BCB.

Primer 5.3. [21] Odrediti najduži zajednički podniz za stringove $X = GTTCCTAATA$ i $Y = CGATAATTGAGA$.
Rešenje:

Tabela se popunjava analogno, kao u prethodnom primeru.

L	-	C	G	A	T	A	A	T	T	G	A	G	A
-	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	1	1	1	1	1	1	1	1	1	1	1
T	0	0	1	1	2	2	2	2	2	2	2	2	2
T	0	0	1	1	2	2	2	3	3	3	3	3	3
C	0	1	1	1	2	2	2	3	3	3	3	3	3
C	0	**1**	1	1	2	2	2	3	3	3	3	3	3
T	0	1	1	1	**2**	2	2	3	4	4	4	4	4
A	0	1	1	2	2	**3**	3	3	4	4	5	5	5
A	0	1	1	2	2	3	**4**	4	4	4	5	5	6
T	0	1	1	2	3	3	4	5	**5**	5	5	5	6
A	0	1	1	2	3	4	4	5	5	5	6	6	**6**

Dužina najdužeg zajedničkog podniza iznosi 6 i jedan takav podniz je $CTAATA$.

5.4 Rešenje problema najdužeg zajedničkog podniza pomoću programskog paketa Matlab

```
% INPUT: X - prvi string
%        Y - drugi string
% OUTPUT: d - duzina najduzeg zajednickog podniza
%         LongestString - najduzi zajednicki podniz

function [d, LongestString] = LCS(X,Y)

n =numel(X);  % duzina prvog stringa
m =numel(Y);  % duzina drugog stringa

% Inicijalizacija matrice L
L=zeros(n+1,m+1);

% b je matrica pridruzena matrici L
b = zeros(n+1,m+1);
b(:,1)=1; % Pomeranje gore
b(1,:)=2; % Pomeranje levo

for i = 2:n+1
    for j = 2:m+1
        if (X(i-1) == Y(j-1))
            L(i,j) = L(i-1,j-1) + 1;
            b(i,j) = 3;
        else
            L(i,j) = L(i-1,j-1);
        end
        if(L(i-1,j) >= L(i,j))
            L(i,j) = L(i-1,j);
            b(i,j) = 1; % Pomeranje gore
        end
        if(L(i,j-1) >= L(i,j))
            L(i,j) = L(i,j-1);
            b(i,j) = 2;% Pomeranje levo
        end
    end
end
```

```
L(:,1) = []; % brisanje prve kolone matrice L
L(1,:) = []; % brisanje prve vrste matrice L
b(:,1) = []; % brisanje prve kolone matrice b
b(1,:) = []; % brisanje prve vrste matrice b
d = L(n,m);
if(d == 0)
    LongestString = '';  % Prazan string
else
    % Odredjivanje najduzeg zajednickog podniza
    i = n;
    j = m;
    p = d;
    LongestString = {};
    while(i>0 && j>0)
        if(b(i,j) == 3)
            LongestString{p} = X(i);
            p = p-1;
            i = i-1;
            j = j-1;
        elseif(b(i,j) == 1)
            i = i-1;
        elseif(b(i,j) == 2)
            j = j-1;
        end
    end
    LongestString = char(LongestString)';
end
end
```

Primer 5.1. Izvršavanjem programa za $X =' BACDB'$ i $Y =' BDCB'$ se dobija rešenje: $d = 3$ i $LongestString = BCB$.
Primer 5.2. Za $X =' GTTCCTAATA'$ i $Y =' CGATAATTGAGA'$ se dobija rešenje: $d = 6$ i $LongestString = CTAAATA$.

6 Prosta raspodela jednorodnog resursa

Neka je na raspolaganju jedna vrsta resursa u ograničenoj količini od S jedinica i posmatrajmo n aktivnosti $A_1, A_2, A_3, ..., A_n$ (različiti proizvodni procesi prerade posmatranog resursa, različite linije, odnosno mašine, radna mesta i slično) u čiju realizaciju je potrebno uložiti određenu količinu resursa. Ukoliko se uloži x jedinica ovog resursa u aktivnost A_i ostvaruje se dobit od $c_i(x)$, pri čemu je funkcija $c_i(x)$ neka unapred zadata nenegativna realna funkcija za sve $i = 1, 2, ..., n$.

Problem proste raspodele jednorodnog resursa dat u [3], se uobičajeno modelira u sledećem obliku

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i(x_i) \\ & \sum_{i=1}^{n} x_i = S \\ & x_i \geq 0,\ i = 1, 2, ..., n. \end{aligned} \tag{6.1}$$

Za svaku aktivnost A_i potrebno je odrediti količinu resursa x_i koju u nju treba uložiti, tako da ukupna ostvarena dobit bude maksimalna i da ukupna raspoloživa količina resursa bude u potpunosti raspoređena.

U slučaju da su kapaciteti procesa (odnosno aktivnosti) ograničeni, javljaju se sledeća ograničenja

$$0 \leq x_i \leq Q_i, \quad \text{pri čemu je } Q_i \text{ kapacitet } i-\text{tog procesa.}$$

Rešavanje problema (6.1) primenom standardnih metoda matematičkog programiranja često ume da bude izuzetno neefikasno jer funkcije c_i mogu da budu nelinearne, a osim toga, nekada je resurs takav da su mu jedinice nedeljive, odnosno promenljive x_i treba da zadovoljavaju i uslov celobrojnosti. Stoga je nekada jednostavnije prethodno opisani proces posmatrati kao jedan višeetapni proces upravljanja.

Etapa i predstavlja realizaciju aktivnosti $A_i,\ i = 1, 2, ..., n$, upravljanje na toj etapi je jednako količini resursa x_i, dok je stanje r_i određeno ukupnom preostalom količinom resursa koja je neraspoređena posle ulaganja u prvih i aktivnosti - $A_1, A_2, ..., A_i$. Upravljanje $x_i \in [0, r_{i-1}]$, a trenutno i prethodno stanje su u relaciji $r_i = r_{i-1} - x_i$. Početno stanje $r_0 = S$ (jer je raspoloživa količina resursa jednaka S), dok je završno stanje $r_n = 0$ (jer je potrebno da ukupna količina resursa bude u potpunosti raspoređena).

Ako se ovaj problem intrepretira pomoću Belmanovog principa optimalnosti onda se on može definisati na sledeći način: Ako se n-tom procesu (aktivnosti) dodeli određena količina ograničenog resursa S, koja će se označiti sa x_n, onda količinu koja preostaje (S–x_n) treba raspodeliti na ostale procese (aktivnosti), pri čemu ostvarena dobit treba da bude maksimalna.

Problem (6.1) se svodi na sledeći oblik

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i(x_i) \\ & r_i = r_{i-1} - x_i, \ i = 1, 2, ..., n. \\ & x_i \in [0, r_{i-1}] \\ & r_0 = S, \ r_n = 0. \end{aligned} \tag{6.2}$$

Neka je $F_i(r)$ maksimalna dobit prilikom ulaganja resursa količine r u prvih i aktivnosti $A_1, A_2, ..., A_i$. U tom slučaju važi

$$F_i(r) = \max_{x_1, x_2, ..., x_i} \sum_{l=1}^{i} c_l(x_l),$$

pri čemu je $\sum_{l=1}^{i} x_l = r$ i $x_l \geq 0$, za sve $l = 1, 2, ..., i$.
U slučaju da je $i = 1$ maksimizacija se vrši samo po jednoj promenljivoj $x_1 = r$ i tada važi

$$F_1(r) = c_1(r). \tag{6.3}$$

Za $i > 1$ važi

$$F_i(r) = \max_{x_1, x_2, ..., x_i} \sum_{l=1}^{i} c_l(x_l) = \max_{x_i \in [0,r]} \left\{ c_i(x_i) + \max_{x_1, x_2, ..., x_{i-1}} \sum_{l=1}^{i-1} c_l(x_l) \right\},$$

pri čemu se prilikom ulaganja x_i količine resursa u aktivnost A_i, maksimizacija ukupne dobiti od aktivnosti $A_1, A_2, ..., A_{i-1}$ vrši po onim $x_1, x_2, ..., x_{i-1}$ za koje važi $\sum_{l=1}^{i-1} x_l = r - x_i$. Na osnovu toga sledi

$$F_i(r) = \max_{x_i \in [0,r]} \left\{ c_i(x_i) + F_{i-1}(r - x_i) \right\}, \quad \text{za } i = 2, 3, ..., n. \tag{6.4}$$

Ovako definisano $F_n(S)$ predstavlja optimalnu vrednost funkcije cilja problema (6.2). Problem se rešava u dve faze.

- U prvoj fazi se na svakoj etapi idući od prve do poslednje određuje maksimalna dobit $F_i(r)$ na osnovu formula (6.3) i (6.4), kao i ulaganje resursa $x_i^*(r) \in [0, r]$ za koje se ta maksimalna dobit dostiže. Pri tom se ove vrednosti računaju za svako $r \in [0, S]$, gde je r moguće ulaganje resursa u prvih i aktivnosti $(A_1, A_2, ..., A_i)$. Vrednost $F_n(S)$ predstavlja maksimalnu dobit, a $x_n^*(S)$ jednaka je optimalnom

ulaganju resursa u aktivnost A_n.
Napomena: Ako su svi x_i celobrojni, maksimizacija se vrši po $x_i \in \{0, 1, ..., r\}$ i F_i se izračunava za svako $r \in \{0, 1, ..., S\}$.

- Kako su u prvoj fazi određene vrednosti $x_1^*(r), x_2^*(r), ..., x_n^*(S)$, sada se određuje niz optimalnih ulaganja $x_1^*, x_2^*, ..., x_n^*$, i to na rekurzivni način polazeći od poslednje do prve etape na sledeći način

$$x_n^* = x_n^*(S), \quad x_i^* = x_i^*(S - x_n^* - x_{n-1}^* - \cdots - x_{i+1}^*), \quad \text{za } i = n-1, ..., 1.$$

Primer 6.1. [3] U okviru jedne proizvodne organizacije je potrebno raspodeliti $S = 4$ jednake mašine na $n = 3$ pogona, pri čemu je dobit $c_i(x)$ koja je ostvarena pri ulaganju x mašina u $i-$ti pogon za $i = 1, 2, 3$ data u tabeli:

$c_i(x)$ \ i	0	1	2	3	4
$c_1(x)$	0	5	9	11	14
$c_2(x)$	0	3	5	8	12
$c_3(x)$	0	6	8	14	15

Odrediti raspodelu mašina po pogonima kojom se obezbeđuje maksimalna ukupna dobit.
Rešenje: Neka su x_1, x_2 i x_3 broj mašina koje treba uložiti u prvi, drugi i treći pogon respektivno. Problem se može zapisati u sledećem obliku.

$$\begin{aligned} \max \quad & c_1(x_1) + c_2(x_2) + c_3(x_3) \\ & x_1 + x_2 + x_3 = 4 \\ & x_i \geq 0, \quad x_i \text{ celobrojno}, \quad i = 1, 2, 3. \end{aligned}$$

Na etapi $i = 1$ i za $r = 0, 1, 2, 3$ važi

$$F_1(r) = c_1(r) \ \text{ i } \ x_1^*(r) = r.$$

To znači da važi

$$\begin{aligned} & F_1(0) = 0 \ \text{ i } \ x_1^*(0) = 0 \\ & F_1(1) = 5 \ \text{ i } \ x_1^*(1) = 1 \\ & F_1(2) = 9 \ \text{ i } \ x_1^*(2) = 2 \\ & F_1(3) = 11 \ \text{ i } \ x_1^*(3) = 3 \\ & F_1(4) = 14 \ \text{ i } \ x_1^*(4) = 4. \end{aligned}$$

Na etapi $i = 2$ je

$$F_2(r) = \max_{0 \le x_2 \le 4} \{c_2(x_2) + F_1(r - x_2)\}$$

pa za $r = 0, 1, 2, 3, 4$ važi

$$\begin{aligned}
F_2(0) &= \max\{c_2(0) + F_1(0)\} = 0 \ \text{ i } \ x_2^*(0) = 0 \\
F_2(1) &= \max\{c_2(0) + F_1(1), c_2(1) + F_1(0)\} = \max\{0 + 5, 3 + 0\} = 5 \ \text{ i } \ x_2^*(1) = 0 \\
F_2(2) &= \max\{c_2(0) + F_1(2), c_2(1) + F_1(1), c_2(2) + F_1(0)\} = \\
&= \max\{0 + 9, 3 + 5, 5 + 0\} = 9 \ \text{ i } \ x_2^*(2) = 0 \\
F_2(3) &= \max\{c_2(0) + F_1(3), c_2(1) + F_1(2), c_2(2) + F_1(1), c_2(3) + F_1(0)\} = \\
&= \max\{0 + 11, 3 + 9, 5 + 5, 8 + 0\} = 12 \ \text{ i } \ x_2^*(3) = 1 \\
F_2(4) &= \max\{c_2(0) + F_1(4), c_2(1) + F_1(3), c_2(2) + F_1(2), c_2(3) + F_1(1), c_2(4) + F_1(0)\} = \\
&= \max\{0 + 14, 3 + 11, 5 + 9, 8 + 5, 12 + 0\} = \max\{14, 14, 14, 13, 12\} = 14 \\
&\text{i } \ x_2^*(4) = 0 \text{ ili } 1 \text{ ili } 2.
\end{aligned}$$

Na etapi $i = 3$ važi

$$\begin{aligned}
F_3(4) &= \max_{0 \le x_3 \le 4} \{c_3(x_3) + F_3(4 - x_3)\} = \\
&= \max\{c_3(0) + F_2(4), c_3(1) + F_2(3), c_3(2) + F_2(2), c_3(3) + F_2(1).c_3(4) + F_2(0)\} = \\
&= \max\{0 + 14, 6 + 12, 8 + 9, 14 + 5, 15 + 0\} = \max\{14, 18, 17, 19, 15\} = 19 \\
&\text{i } \ x_3^*(4) = 3.
\end{aligned}$$

Maksimalna dobit prilikom ulaganja 4 mašine u 3 pogona je jednaka 19 novčanih jedinica. Optimalni niz upravljanja x_1^*, x_2^*, x_3^* se formira na sledeći način:

$$x_3^* = x_3^*(4) = 3, \quad x_2^* = x_2^*(4 - x_3^*) = x_2^*(1) = 0, \quad x_1^* = x_1^*(4 - x_2^* - x_3^*) = x_1^*(1) = 1.$$

Dakle, optimalna strategija podrazumeva ulaganje 1 mašine u prvi pogon i 3 mašine u treći pogon.

Primer 6.2. [6] Za potrebe rada proizvodnog sistema, potrebno je table iverice (jednorodni resurs) raspodeliti na 3 moguće proizvodne linije, pri čemu je kapacitet proizvodnih linija ograničen i važi $0 \le x_i \le 3$, za $i = 1, 2, 3$. Količina iverice je takođe ograničena i iznosi $S = 6$ jedinica. Ostvarena dobit na proizvodnim linijama zavisi od količine (broja) prerađene iverice i iznosi:

za liniju 1: $c_1(x) = 4x^2$

za liniju 2: $c_2(x) = 7x^2$

za liniju 3: $c_3(x) = 3x^2$

Ograničenu količinu iverice treba distribuirati po proizvodnim linijama tako da ostvarena dobit od njene prerade bude maksimalna.

Rešenje: Ako x_1, x_2 i x_3 predstavljaju količinu (broj tabli) iverice dodeljenu prvoj, drugoj i trećoj proizvodnoj liniji respektivno, problem se može zapisati u sledećem obliku

$$\begin{aligned} \max \quad & 4x_1^2 + 7x_2^2 + 3x_3^2 \\ & x_1 + x_2 + x_3 = 6 \\ & 0 \le x_i \le 3, \ \text{ i } \ x_i \text{ je celobrojno, za } i = 1, 2, 3. \end{aligned}$$

Na etapi $i = 1$ i za $r = 0, 1, 2, 3, 4, 5, 6$ važi

$$F_1(r) = \max_{0 \le r \le 3} c_1(r).$$

To znači da važi

$$\begin{aligned} F_1(0) &= 0 \quad \text{i} \ \ x_1^*(0) = 0 \\ F_1(1) &= 4 \quad \text{i} \ \ x_1^*(1) = 1 \\ F_1(2) &= 16 \ \text{i} \ \ x_1^*(2) = 2 \\ F_1(3) &= 36 \ \text{i} \ \ x_1^*(3) = 3 \\ F_1(4) &= 36 \ \text{i} \ \ x_1^*(4) = 3 \\ F_1(5) &= 36 \ \text{i} \ \ x_1^*(3) = 3 \\ F_1(6) &= 36 \ \text{i} \ \ x_1^*(4) = 3. \end{aligned}$$

Na etapi $i = 2$ za $r = 0, 1, 2, 3, 4, 5, 6$ važi

$$F_2(r) = \max_{0 \le x_2 \le 3} \{c_2(x_2) + F_1(r - x_2)\} = \max_{0 \le x_2 \le 3} \{7x_2^2 + F_1(r - x_2)\}.$$

$$F_2(0) = \max_{x_2 = 0} \{7x_2^2 + F_1(0 - x_2)\} = 7 \cdot 0^2 + F_1(0 - 0) = 0, \ \text{za } x_2^*(0) = 0$$

$$F_2(1) = \max_{0 \le x_2 \le 1} \{7x_2^2 + F_1(1 - x_2)\} = \max \begin{cases} 7 \cdot 0^2 + F_1(1 - 0) \\ 7 \cdot 1^2 + F_1(1 - 1) \end{cases} = \max \begin{cases} 0 + 4 \\ 7 + 0 \end{cases} = 7,$$

$$\text{za } x_2^*(1) = 1$$

$$F_2(2) = \max_{0 \le x_2 \le 2} \{7x_2^2 + F_1(2 - x_2)\} = \max \begin{cases} 7 \cdot 0^2 + F_1(2 - 0) \\ 7 \cdot 1^2 + F_1(2 - 1) \\ 7 \cdot 2^2 + F_1(2 - 2) \end{cases} = \max \begin{cases} 0 + 16 \\ 7 + 4 \\ 28 + 0 \end{cases} = 28,$$

$$\text{za } x_2^*(2) = 2$$

$$F_2(3) = \max_{0\le x_2\le 3}\{7x_2^2 + F_1(3-x_2)\} = \max\begin{cases} 7\cdot 0^2 + F_1(3-0) \\ 7\cdot 1^2 + F_1(3-1) \\ 7\cdot 2^2 + F_1(3-2) \\ 7\cdot 3^2 + F_1(3-3) \end{cases} = \max\begin{cases} 0+36 \\ 7+16 \\ 28+4 \\ 63+0 \end{cases} = 63,$$

za $x_2^*(3) = 3$

$$F_2(4) = \max_{0\le x_2\le 3}\{7x_2^2 + F_1(4-x_2)\} = \max\begin{cases} 7\cdot 0^2 + F_1(4-0) \\ 7\cdot 1^2 + F_1(4-1) \\ 7\cdot 2^2 + F_1(4-2) \\ 7\cdot 3^2 + F_1(4-3) \end{cases} = \max\begin{cases} 0+36 \\ 7+36 \\ 28+16 \\ 63+4 \end{cases} = 67,$$

za $x_2^*(4) = 3$

$$F_2(5) = \max_{0\le x_2\le 3}\{7x_2^2 + F_1(5-x_2)\} = \max\begin{cases} 7\cdot 0^2 + F_1(5-0) \\ 7\cdot 1^2 + F_1(5-1) \\ 7\cdot 2^2 + F_1(5-2) \\ 7\cdot 3^2 + F_1(5-3) \end{cases} = \max\begin{cases} 0+36 \\ 7+36 \\ 28+36 \\ 63+16 \end{cases} = 79,$$

za $x_2^*(5) = 3$

$$F_2(6) = \max_{0\le x_2\le 3}\{7x_2^2 + F_1(6-x_2)\} = \max\begin{cases} 7\cdot 0^2 + F_1(6-0) \\ 7\cdot 1^2 + F_1(6-1) \\ 7\cdot 2^2 + F_1(6-2) \\ 7\cdot 3^2 + F_1(6-3) \end{cases} = \max\begin{cases} 0+36 \\ 7+36 \\ 28+36 \\ 63+36 \end{cases} = 99,$$

za $x_2^*(6) = 3$

Na etapi $i = 3$ važi

$$F_3(r) = \max_{0\le x_3\le r}\{c_3(x_3) + F_2(r-x_3)\} = \max_{0\le x_3\le r}\{3x_3^2 + F_2(r-x_3)\}.$$

Na osnovu toga sledi

$$F_3(6) = \max_{0\le x_3\le 3}\{3x_3^2 + F_2(6-x_3)\} = \max\begin{cases} 3\cdot 0^2 + F_2(6-0) \\ 3\cdot 1^2 + F_2(6-1) \\ 3\cdot 2^2 + F_2(6-2) \\ 3\cdot 3^2 + F_2(6-3) \end{cases} = \max\begin{cases} 0+F_2(6) \\ 3+F_2(5) \\ 12+F_2(4) \\ 27+F_2(3) \end{cases} =$$

$$= \max \begin{cases} 0+99 \\ 3+79 \\ 12+67 \\ 27+63 \end{cases} = 99, \text{ za } x_3^*(6) = 0.$$

Optimalni niz upravljanja se formira na sledeći način

$$x_3^* = x_3^*(6) = 0, \quad x_2^* = x_2^*(6 - x_3^*) = x_2^*(6) = 3, \quad x_1^* = x_1^*(6 - x_2^* - x_3^*) = x_1^*(3) = 3.$$

To znači da ograničenu količinu iverice treba raspodeliti na sledeći način: na prvu liniju tri table, na drugu tri, a na treću nijedna. Prilikom ovakve raspodele resursa (iverice) ostvaruje se maksimalna dobit koja je jednaka $F_3(6) = 99$ (novčanih jedinica).

Primer 6.3. [1] Investitor ima na raspolaganju \$6000 i ulaže novac u 3 aktive. Ako je d_j dolara (u hiljadama) uloženo u $j-$tu aktivu onda je NPV - Net Present Value (NSV - Neto Sadašnja Vrednost) za sve aktive data u nastavku

$$\begin{aligned} r_1(d_1) &= 7d_1 + 2 \\ r_2(d_2) &= 3d_2 + 7 \\ r_3(d_3) &= 4d_3 + 5 \\ r_1(0) &= r_2(0) = r_3(0) = 0. \end{aligned}$$

U svaku aktivu je dozvoljeno uložiti celobrojni umnožak od \$1000. Investitior želi da uloži \$6000 u ove tri aktive tako da neto sadašnja vrednost bude maksimalna.

Neto sadašnja vrednost (Net present value) je jedan od ključnih kriterijuma za ocenu ulaganja sredstava. Predstavlja razliku između sadašnje vrednosti novčanih priliva i odliva projekta ili potencijalne investicije. Većina investitora koristi analizu sadašnje vrednosti ili metodu diskontovanog novčanog toka prilikom donošenja odluka o ulaganju. To ima smisla jer je potrebno videti ishod svojih izbora kada se uzmu u obzir troškovi kamata i drugi vremenski faktori. Što je veći iznos neto sadašnje vrednosti firma (investitor) će ostvariti više prihoda. Računanje neto sadašnje vrednosti osim za razlikovanje dobrih investicija od loših može se koristiti i za upoređivanje dve investicije kako bi se odredilo koja od njih je isplativija. Što je neto sadašnja vrednost veća, investicija je isplativija. Neto sadašnja vrednost je dobar, ali ne nužno i tačan pokazatelj za donošenje investicionih odluka.

Rešenje: Problem se može zapisati na sledeći način

$$\begin{aligned} \max \quad & r_1(d_1) + r_2(d_2) + r_3(d_3) \\ & d_1 + d_2 + d_3 = 6 \\ & d_i \geq 0, \ i = 1, 2, 3. \end{aligned}$$

Prinos svake investicije nije proporcionalan iznosu novca uloženog u nju. Na primer: $16 = r_1(2) \neq 2r_1(1) = 18$. Dakle, nije moguće rešavati problem tehnikama linearnog programiranja.

Neka je $f_t(d_t)$ maksimalna neto sadašnja vrednost koje se dobija investiranjem d_t dolara u investicije $t, t+1, ..., 3$ i neka je $x_t(d_t)$ količina novca koju je potrebno uložiti u aktivu t da bi se dobila vrednost $f_t(d_t)$.

Problem se rešava korišećenjem "principa unazad". U prvoj etapi se računaju vrednosti $f_3(0), f_3(1), ..., f_3(6)$, u drugoj $f_2(0), f_2(1), ..., f_2(6)$ a u trećoj $f_1(6)$.

Etapa 1: Primetimo da se $f_3(d_3)$ postiže ulaganjem svog raspoloživog novca u aktivu 3.

$$\begin{aligned}
&f_3(0) = 0, \quad && x_3(0) = 0\\
&f_3(1) = 9, \quad && x_3(1) = 1\\
&f_3(2) = 13, \quad && x_3(2) = 2\\
&f_3(3) = 17, \quad && x_3(3) = 3\\
&f_3(4) = 21, \quad && x_3(4) = 4\\
&f_3(5) = 25, \quad && x_3(5) = 5\\
&f_3(6) = 29, \quad && x_3(6) = 6.
\end{aligned}$$

Etapa 2: Maksimizacija neto sadašnje vrednosti prilikom ulaganja u aktive 2 i 3.

$$f_2(d_2) = \max_{x_2}\{r_2(x_2) + f_3(d_2 - x_2)\}, \quad \text{pri čemu je } x_2 \in \{0, 1, ..., 6\}.$$

Rezultati su dati u narednoj tabeli.

d_2	x_2	$r_2(x_2)$	$f_3(d_2 - x_2)$	NPV za investicije 2 i 3	$f_2(d_2)$ $x_2(d_2)$
0	0	0	0	0^*	$f_2(0) = 0$
					$x_2(0) = 0$
1	0	0	9	9	$f_2(1) = 10$
1	1	10	0	10^*	$x_2(1) = 1$
2	0	0	13	13	$f_2(2) = 19$
2	1	10	9	19^*	$x_2(2) = 1$
2	2	13	0	13	
3	0	0	17	17	$f_2(3) = 23$
3	1	10	13	23^*	$x_2(3) = 1$
3	2	13	9	22	
3	3	16	0	16	
4	0	0	21	21	$f_2(4) = 27$
4	1	10	17	27^*	$x_2(4) = 1$
4	2	13	13	26	
4	3	16	9	25	
4	4	19	0	19	
5	0	0	25	25	$f_2(5) = 31$
5	1	10	21	31^*	$x_2(5) = 1$
5	2	13	17	30	
5	3	16	13	29	
5	4	19	9	28	
5	5	22	0	22	
6	0	0	29	29	$f_2(6) = 35$
6	1	10	25	35^*	$x_2(6) = 1$
6	2	13	21	34	
6	3	16	17	33	
6	4	19	13	32	
6	5	22	9	31	
6	6	25	0	25	

Etapa 3: Na osnovu prethodno dobijenih rezultata se računa

$$f_1(6) = \max_{x_1}\{r_1(x_1) + f_2(6 - x_1)\}.$$

Računanje ove vrednosti je dato u narednoj tabeli

d_1	x_1	$r_1(x_1)$	$f_1(6-x_1)$	NPV za investicije $1-3$	$f_1(6)$ $x_1(6)$
6	0	0	35	35	$f_1(6) = 49$
6	1	9	31	40	$x_1(6) = 4$
6	2	16	27	43	
6	3	23	23	46	
6	4	30	19	49^*	
6	5	37	10	47	
6	6	44	0	44	

Kako je $x_1(6) = 4$, investitor će uložiti \$4000 u prvu aktivu. Dalje je preostalo \$2000 koje treba investirati u aktive 2 i 3. S obzirom da važi $x_2(2) = 1$, potrebno je investirati \$1000 u aktivu 2. Preostalih \$1000 će investitor uložiti u aktivu 3. Na ovaj način investitor može postići maksimalnu neto sadašnju vrednost koja iznosi $f_1(6) = 49\,000$.

6.1 Optimizacija poslovnog procesa FMT iz Zaječara

Fabrika mernih transformatora (FMT) iz Zaječara proizvodi elektro opremu. Glavni zadatak menadžmenta je raspodela sredstava za proizvodnju artikala tako da fabrika ostvari maksimalni profit.

Ovo istraživanje se sprovodi na 3 proizvoda:

- T_1 - Trafo niskog napona STEM-081 do 0,72 kV,
- T_2 - Trafo srednjeg napona STEM-N 3821 do 35 kV,
- T_3 - Transformator JNT SM 24/12 od 24 kV.

Menadžment firme ima na raspolaganju 80 000 evra. U tabeli u nastavku je dat očekivani profit prilikom investiranja $0, 20\ 000,\ 40\ 000, 60\ 000$ i 80 000 evra u proizvode T_1, T_2, T_3. [2]

[2]Podaci su preuzeti iz rada: N. Petković, M. Božinović: *The aplicaton of the dynamic programming method in investment optimization,* 2016.

Iznos investicije (u evrima)	T_1	T_2	T_3
20 000	9 632	7 428	6 084
40 000	16 000	11 800	9 272
60 000	38 772	31 422	26 946
80 000	41 600	43 536	39 040

Neka je $f_i(x_i)$ očekivani profit prilikom ulaganja iznosa x_i u proizvod $T_i, i = 1, 2, 3$. Matematički model posmatranog problema je

$$\begin{aligned} \max \quad & \sum_{i=1}^{3} f_i(x_i) = f_1(x_1) + f_2(x_2) + f_3(x_3) \\ & \sum_{i=1}^{3} x_i = 80000 \\ & x_i \geq 0, i = 1, 2, 3. \end{aligned}$$

Dalje se problem rešava primenom Belmanovog principa otpimalnosti, tako što se problem podeli na nekoliko etapa. Na svakoj etapi se računa vrednost $F_k(r_k)$ koja predstavlja maksimalni profit prilikom ulaganja iznosa r_k u prvih k proizvoda, za $k = 1, 2, 3$, pri čemu $r_k \in \{0, 20000, 40000, 60000, 80000\}$.

- **Etapa 1:** U prvoj fazi se računaju vrednosti $F_1(r_1)$, pri čemu $r_1 \in \{0, 20000, 40000, 60000, 80000\}$. Daljim izračunavanjem se dobijaju sledeći rezultati

$$\begin{aligned} F_1(0) &= 0 \\ F_1(20000) &= 9632 \\ F_1(40000) &= 16000 \\ F_1(60000) &= 38772 \\ F_1(80000) &= 41600. \end{aligned}$$

- **Etapa 2:** U drugoj fazi se određuje optimalna raspodela ulaganja u proizvode T_1 i T_2, odnosno određuje se $F_2(r_2)$ na osnovu sledeće formule

$$F_2(r_2) = \max_{x_2 \leq r_2} \{f_2(x_2) + F_1(r_2 - x_2)\}, r_2 \in \{0, 20000, 40000, 60000, 80000\}.$$
$$F_2(0) = f_2(0) + F_1(0) = 0 + 0 = 0, \quad x_2 = 0$$

$$F_2(20000) = \max_{x_2 \le 20000}\{f_2(x_2) + F_1(20000 - x_2)\} = \max\begin{cases} f_2(0) + F_1(20000) \\ f_2(20000) + F_1(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 9632 \\ 7428 + 0 \end{cases} = 9632, \quad x_2 = 0$$

$$F_2(40000) = \max_{x_2 \le 40000}\{f_2(x_2) + F_1(40000 - x_2)\} = \max\begin{cases} f_2(0) + F_1(40000) \\ f_2(20000) + F_1(20000) \\ f_2(40000) + F_1(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 16000 \\ 7428 + 9632 \\ 11800 + 0 \end{cases} = \max\{16000, 17060, 11800\} = 17060,$$

$$x_1 = x_2 = 20000$$

$$F_2(60000) = \max_{x_2 \le 60000}\{f_2(x_2) + F_1(60000 - x_2)\} = \max\begin{cases} f_2(0) + F_1(60000) \\ f_2(20000) + F_1(40000) \\ f_2(40000) + F_1(20000) \\ f_2(60000) + F_1(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 38772 \\ 7428 + 16000 \\ 11800 + 9632 \\ 31422 + 0 \end{cases} = \max\{38772, 23428, 21432, 31422\} = 38772,$$

$$x_2 = 0$$

$$F_2(80000) = \max_{x_2 \le 80000}\{f_2(x_2) + F_1(80000 - x_2)\} = \max\begin{cases} f_2(0) + F_1(80000) \\ f_2(20000) + F_1(60000) \\ f_2(40000) + F_1(40000) \\ f_2(60000) + F_1(20000) \\ f_2(80000) + F_1(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 416000 \\ 7428 + 38772 \\ 11800 + 16000 \\ 31422 + 9632 \\ 43536 + 0 \end{cases} = \max\{41600, 46200, 27800, 41054, 43536\} =$$

$$= 46200, \quad x_1 = 60000, \; x_2 = 20000$$

- **Etapa 3:** U ovoj fazi se raspoređuju sredstva na sva tri proizvoda i važi

$$F_3(r_3) = \max_{x_3 \le r_3}\{f_3(x_3) + F_2(r_3 - x_3)\},\ r_3 \in \{0, 20000, 40000, 60000, 80000\}.$$

$$F_3(0) = f_3(0) + F_2(0) = 0 + 0 = 0,\ x_3 = 0$$

$$F_3(20000) = \max_{x_3 \le 20000}\{f_3(x_3) + F_2(20000 - x_3)\} = \max\begin{cases} f_3(0) + F_2(20000) \\ f_3(20000) + F_2(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 9632 \\ 6084 + 0 \end{cases} = 9632,\ x_3 = 0$$

$$F_3(40000) = \max_{x_3 \le 40000}\{f_3(x_3) + F_2(40000 - x_3)\} = \max\begin{cases} f_3(0) + F_2(40000) \\ f_3(20000) + F_2(20000) \\ f_3(40000) + F_2(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 17060 \\ 6084 + 9632 \\ 9272 + 0 \end{cases} = \max\{17060, 15716, 9272\} = 17060,\ x_3 = 0$$

$$F_3(60000) = \max_{x_3 \le 60000}\{f_3(x_3) + F_2(60000 - x_3)\} = \max\begin{cases} f_3(0) + F_2(60000) \\ f_3(20000) + F_2(40000) \\ f_3(40000) + F_2(20000) \\ f_3(60000) + F_2(0) \end{cases} =$$

$$= \max\begin{cases} 0 + 38772 \\ 6084 + 17060 \\ 9272 + 9632 \\ 26946 + 0 \end{cases} = \max\{38772, 23144, 18904, 26946\} = 38772,$$

$$x_3 = 0$$

$$F_3(80000) = \max_{x_3 \le 80000}\{f_3(x_3) + F_2(80000 - x_3)\} = \max\begin{cases} f_3(0) + F_2(80000) \\ f_3(20000) + F_2(60000) \\ f_3(40000) + F_2(40000) \\ f_3(60000) + F_2(20000) \\ f_3(80000) + F_2(0) \end{cases} =$$

$$= \max \begin{cases} 0 + 46200 \\ 6084 + 38772 \\ 9272 + 17060 \\ 26946 + 9632 \\ 39040 + 0 \end{cases} = \max\{46200, 44856, 26332, 36578, 39040\} = 46200,$$

$x_3 = 0,\ x_2 = 20000,\ x_1 = 60000.$

Rezultati prethodno sprovedene analize su izloženi u sledećoj tabeli.

Iznos investicije (u evrima)	x_1	F_1	x_2	F_2	x_3	F_3
0	0	0	0	0	0	0
20 000	20 000	9 632	0	9 632	0	9 632
40 000	40 000	16 000	20 000	17 060	0	17 060
60 000	60 000	38 772	0	35 772	0	38 772
80 000	80 000	41 600	20 000	46 200	0	46 200

Dakle, iznos od 80 000 treba raspodeliti na sledeći način: 60 000 treba investirati u proizvodnju T_1 i 20 000 u proizvodnju T_2. Investiranje u proizvod T_3 nije isplativo. Maksimalni očekivani profit iznosi 46 200 evra.

7 Raspodela poslova na mašine

Neka je na raspolaganju određen broj mašina koje mogu da obavljaju poslove tipa 1 i 2. Poznato je da ukoliko x mašina u toku jednog razmatranog vremenskog perioda obavlja posao tipa 1, to će kompaniji doneti dobit $g(x)$ na kraju tog perioda. Takođe je poznato da će jedan deo mašina biti amortizovan, odnosno neće biti za dalju upotrebu, što znači da se u narednom periodu može računati na $a(x)$ mašina od ukupno x mašina koje su obavljale posao 1. Slično važi za mašine koje obavljaju posao tipa 2. Ako y mašina u toku posmatranog perioda obavlja posao tipa 2, to će kompaniji doneti dobit $h(y)$, a ukupno će $b(y)$ mašina biti raspoloživo za rad u narednom periodu. Dodatna pretpostavka je da u svakom trenutku svaka mašina obavlja jednu od ova dva posla. Drugim rečima ni jedna mašina ne pauzira. Zadatak se svodi na određivanje optimalne raspodele mašina tako da dobit posle N perioda bude maksimalna.
U toku prvog perioda važi:

- Ako broj mašina koje obavljaju posao tipa 1 iznosi x_1, onda je broj mašina koje obavljaju posao tipa 2 jednak $y_1 = n_1 - x_1$, pri čemu je n_1 ukupan broj raspoloživih mašina na samom početku.
- Ukupna dobit iznosi $g(x_1) + h(y_1)$.
- Na kraju prvog perioda je broj upotrebljivih mašina $a(x_1) + b(y_1)$.

Za drugi period važi:

- Na početku drugog perioda je broj raspoloživih mašina jednak $a(x_1) + b(y_1)$.
- Od svih raspoloživih mašina se za prvi posao koristi x_2, a za drugi $y_2 = n_2 - x_2$ mašina.
- Ostvarena dobit u ovom periodu iznosi $g(x_2) + h(y_2)$.
- Na kraju perioda broj upotrebljivih mašina iznosi $n_3 = a(x_2) + b(y_2)$.

Slično važi za sve periode do $N-$tog, pa se stoga dobija sledeći matematički model problema

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} (g(x_i) + h(y_i)) \\ & x_i + y_i = n_i \\ & n_{i+1} = a(x_i) + b(y_i)\ i = 1, 2, ..., N-1. \\ & 0 \leq x_i \leq n_i,\ i = 1, 2, ..., N. \end{aligned} \tag{7.1}$$

U slučaju kada su ograničenja i funkcija cilja linearni, problem se svodi na problem linearnog programiranja koji se rešava tehnikama linearnog programiranja. Ako je problem nelinearan on se rešava pomoću dinamičkog programiranja.

Neka je $f_i(n)$ maksimalna dobit posle $i-$tog perioda, pri čemu je n broj raspoloživih mašina na početku prvog perioda, tada važi:

Ukoliko do kraja planskog perioda prestaje samo jedan period (ili se razmatra problem koji ima samo jedan period), tada važi

$$f_1(n) = \max_{0 \le x_1 \le n} \{g(x_1) + h(n - x_1)\}.$$

Ukoliko je do kraja planskog perioda preostalo k perioda, tada važi

$$f_k(n) = \max_{0 \le x_l \le n} \{g(x_k) + h(n - x_k) + f_{k-1}(a(x_k) + b(n - x_k))\}, \quad \text{za} \quad k \ge 1.$$

Potpuno analogno kao u prethodnom zadatku se računaju vrednosti $x_1^*, ..., x_n^*$.

Primer 7.1. [6] Preduzeće raspolaže sa 100 visokoproduktivnih strugova. U naredne četiri godine ($N = 4$) preduzeće treba da izrađuje vratila tipa A i tipa B, koja se mogu izrađivati na tim strugovima. U zavisnosti od raspodele strugova, u $i-$tom periodu preduzeće ostvaruje dobit

$$g(x_i) + h(y_i) = 0,6x_i + 0,4y_i.$$

Amortizacija zavisi od vrste vratila koje se izrađuje na strugovima i iznosi 60 % za strugove koji izrađuju vratila tipa A i 30 % za strugove koji izrađuju vratila tipa B. To znači da broj raspoloživih strugova za naredni period iznosi

$$n_{i+1} = 0,4x_i + 0,7y_i.$$

Odrediti broj strugova koji će izrađivati vratila tipa A i B u svakom periodu, tako da posle $N = 4$ perioda (godine) ostvarena dobit bude maksimalna.

Rešenje: Matematički model posmatranog problema je

$$\begin{aligned} \max \quad & \sum_{i=1}^{4} (0,6x_i + 0,4y_i) \\ & x_i + y_i = n_i \\ & n_{i+1} = 0,4x_i + 0,7y_i \\ & 0 \le x_i \le n_i, \ n_1 = n. \end{aligned}$$

Prvo se posmatra četvrti period. Maksimalna dobit u ovom periodu iznosi

$$f_1(n) = \max_{0 \le x_4 \le n_4} \{0,6x_4 + 0,4y_4\}.$$

Iz ograničenja sledi $x_4 + y_4 = n_4 \Rightarrow y_4 = n_4 - x_4$.

$$f_1(n) = \max_{0 \le x_4 \le n_4} \{0,6x_4 + 0,4(n_4 - x_4)\} = \max_{0 \le x_4 \le n_4} \{0,2x_4 + 0,4n_4\}.$$

Posmatrana funkcija je linearna, pa je potrebno proveriti dva slučaja: $x_4 = 0$ i $x_4 = n_4$.

Za $x_4 = 0$ sledi

$$f_1(n) = 0,4n_4.$$

Za $x_4 = n_4$ sledi

$$f_1(n) = 0,6n_4.$$

S obzirom da je potrebno odrediti maksimum funkcije, jasno je da se maksimum dostiže za $x_4 = n_4$, i pri tome dobit preduzeća iznosi $f_1(n) = 0,6n_4$. U narednom koraku se posmatra dobit za treći i četvrti period i ona se određuje pomoću formule

$$f_2(n) = \max_{0 \le x_3 \le n_3} \{0,6x_3 + 0,4y_3 + f_1(0,4x_3 + 0,7y_3)\}.$$

Iz ograničenja sledi $x_3 + y_3 = n_3 \Rightarrow y_3 = n_3 - x_3$. Odatle sledi

$$\begin{aligned} f_2(n) &= \max_{0 \le x_3 \le n_3} \{0,6x_3 + 0,4(n_3 - x_3) + 0,6(0,4x_3 + 0,7(n_3 - x_3))\} = \\ &= \max_{0 \le x_3 \le n_3} \{0,02x_3 + 0,82n_3\}. \end{aligned}$$

Kao i u prethodnom koraku razmatraju se dva slučaja i to kada je $x_3 = 0$ i kada je $x_3 = n_3$.

Za $x_3 = 0$ sledi

$$f_2(n) = 0,82n_3.$$

Za $x_3 = n_3$ sledi

$$f_2(n) = 0,84n_3.$$

Maksimum funkcije se dostiže za $x_3 = n_3$, i pri tome dobit preduzeća iznosi $f_2(n) = 0,84n_3$. U trećem koraku se posmatra maksimalna dobit za četvrti, treći i drugi period

$$f_3(n) = \max_{0 \le x_2 \le n_2} \{0,6x_2 + 0,4y_2 + f_2(0,4x_2 + 0,7y_2)\}.$$

Iz ograničenja sledi $x_2 + y_2 = n_2 \Rightarrow y_2 = n_2 - x_2$. Kada se uvrste ova ograničenja dobija se

$$f_3(n) = \max_{0 \le x_2 \le n_2} \{0,6x_2 + 0,4(n_2 - x_2) + 0,84(0,4x_2 + 0,7(n_2 - x_2))\} =$$
$$= \max_{0 \le x_2 \le n_2} \{-0,052x_2 + 0,988n_2\}.$$

Kao i u prethodnom koraku razmatraju se dva slučaja i to kada je $x_2 = 0$ i kada je $x_2 = n_2$.

Za $x_2 = 0$ sledi

$$f_3(n) = 0,988n_2.$$

Za $x_2 = n_2$ sledi

$$f_3(n) = 0,936n_2.$$

Maksimum funkcije se dostiže za $x_2 = 0$, i pri tome dobit preduzeća iznosi $f_3(n) = 0,988n_2$. Kada se uzmu u obzir sva 4 perioda dobija se sledeča formula

$$f_4(n) = \max_{0 \le x_1 \le n_1} \{0,6x_1 + 0,4y_1 + f_3(0,4x_1 + 0,7y_1)\}.$$

Iz ograničenja sledi $x_1 + y_1 = n_1 \Rightarrow y_1 = n_1 - x_1$. Kada se uvrste ova ograničenja dobija se

$$f_4(n) = \max_{0 \le x_1 \le n_1} \{0,6x_1 + 0,4(n_1 - x_1) + 0,988(0,4x_1 + 0,7(n_1 - x_1))\} =$$
$$= \max_{0 \le x_1 \le n_1} \{-0,0964x_1 + 1,0916n_1\}.$$

Dalje se razmatraju dva slučaja i to kada je $x_1 = 0$ i kada je $x_1 = n_1$.

Za $x_1 = 0$ sledi

$$f_4(n) = 1,0916n_1.$$

Za $x_1 = n_1$ sledi

$$f_4(n) = 0,9952n_1.$$

Dakle, maksimum funkcije se dostiže za $x_1 = 0$ i dobit preduzeća iznosi $f_4(n) = 1,0916n$. Rešenje problema, odnosno maksimalna dobit preduzeća iznosi

$$f_4(100) = 1,0916 \cdot 100 = 109,16 \text{ novčanih jedinica.}$$

U nastavku se određuje optimalna strategija firme koja će obezbediti maksimalnu dobit.

Prvi period: $x_1^* = 0, \;\; y_1^* = 100.$
U ovom periodu se proizvode samo vratila tipa B.
Na kraju prvog perioda, zbog amortizacije, broj strugova koji se koriste u drugom periodu iznosi

$$n_2 = 0,4x_1 + 0,7y_1 = 0,4 \cdot 0 + 0,7 \cdot 100 = 70.$$

Drugi period: $n_2 = 70 \;\; x_2^* = 0, \;\; y_2^* = 70.$
U ovom periodu se proizvode samo vratila tipa B.
Na kraju drugog perioda, zbog amortizacije, broj strugova koji se koriste u trećem periodu iznosi

$$n_3 = 0,4x_2 + 0,7y_2 = 0,4 \cdot 0 + 0,7 \cdot 70 = 49.$$

Treći period: $n_3 = 49 \;\; x_3^* = 49, \;\; y_3^* = 0.$
U ovom periodu se proizvode samo vratila tipa A.
Na kraju trećeg perioda, zbog amortizacije, broj strugova koji se koriste u četvrtom periodu iznosi

$$n_4 = 0,4x_4 + 0,7y_4 = 0,4 \cdot 49 + 0,7 \cdot 0 = 19,6 \approx 20.$$

Četvrti period: $n_4 = 20 \;\; x_4^* = 20, \;\; y_4^* = 0.$
U ovom periodu se proizvode samo vratila tipa A.
Na kraju trećeg perioda, zbog amortizacije, broj strugova koji je preostao za eventualni peti period iznosi

$$n_5 = 0,4x_4 + 0,7y_4 = 0,4 \cdot 20 + 0,7 \cdot 0 = 8.$$

8 Složena raspodela jednorodnog resursa

Posmatra se proizvodni proces u okviru kog je potrebno proizvesti n vrsta proizvoda $P_1, P_2, ..., P_n$ koristeći jedan tip resursa čija količina je ograničena i iznosi S jedinica. Prilikom proizvodnje x jedinica $i-$tog proizvoda P_i potrebno je utrošiti količinu $a_i(x)$ ovog resursa, pri čemu se ostvaruje dobit $c_i(x)$, gde su $a_i(x)$ i $c_i(x)$ neke unapred zadate, realne, nenegativne funkcije za sve $i = 1, 2, ..., n$. Problem složene raspodele jednorodnog resursa, prikazan u [3], se uobičajeno modelira u sledećem obliku

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i(x_i) \\ & \sum_{i=1}^{n} a_i(x_i) \leq S \\ & x_i \geq 0,\ i = 1, 2, ..., n. \end{aligned} \tag{8.1}$$

S obzirom da su funkcije $a_i(x)$ i $c_i(x)$ u praksi često nelinearne i promenljive x_i često zadovoljavaju uslov celobrojnosti, to znatno otežava rešavanje problema (8.1). Stoga je, kao i kod problema proste raspodele jednorodnog resursa, nekada jednostavnije ovaj proces posmatrati kao jedan višeetapni proces upravljanja i to na sledeći način:

- $i-$ta etapa procesa predstavlja proizvodnju proizvoda $P_i, i = 1, 2, ..., n$,
- proizvedena količina $i-$tog proizvoda x_i predstavlja upravljanje na $i-$toj etapi,
- stanje r_i je ukupna preostala količina resursa posle ulaganja u proizvodnju prvih i proizvoda $(P_1, P_2, ..., P_i)$,
- skup dopustivih upravljanja na etapi i sadrži sve nenegativne brojeve x za koje važi $a_i(x) \leq r_{i-1}$,
- zakon prelaska iz jednog stanja u naredno je definisan na sledeći način

$$r_i = r_{i-1} - a_i(x_i),$$

- početno stanje $r_0 = S$.

Problem (8.1) se može zapisati u ekvivalentnom obliku

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i(x_i) \\ & r_i = r_{i-1} - a_i(x_i),\ i = 1, 2, ..., n \\ & x_i \in \{x \in \mathbb{R}^+ | a_i(x) \leqslant r_{i-1}\} \\ & r_0 = S, \end{aligned} \tag{8.2}$$

gde je $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geq 0\}$.

Neka je $F_i(r)$ maksimalna dobit prilikom proizvodnje proizvoda $P_1, P_2, ..., P_i$, pri čemu ukupna utrošena količina resursa ne prelazi r, odnosno važi

$$F_i(r) = \max_{x_1, x_2, ... x_i} \sum_{l=1}^{i} c_l(x_l),$$

i pri tome se maksimizacija vrši po svim $x_1, x_2, ...x_i \in \mathbb{R}^+$ za koje važi $\sum_{l=1}^{i} a_l(x_l) \leq r$.

U nastavku se posmatraju dva slučaja:

Za $i = 1$ važi

$$F_1(r) = \max_{x_1} c_1(x_1), \tag{8.3}$$

gde se maksimizacija vrši po svim $x_1 \in \mathbb{R}^+$ za koje je $a_1(x_1) \leq r$.

Za $i > 1$ važi

$$F_i(r) = \max_{x_i} \{c_i(x_i) + F_{i-1}(r - a_i(x_i))\}, \tag{8.4}$$

pri čemu se maksimizacija vrši po svim $x_i \in \mathbb{R}^+$ za koje važi $a_i(x_i) \leq r$.

Problem (8.2) se rešava u dve faze:

- **Prva faza:** Na osnovu formula (8.3) i (8.4) se dobijaju maksimalne dobiti $F_1(r), F_2(r), ..., F_n(S)$ i njima odgovarajuće količine proizvoda $x_1^*(r), x_2^*(r), ...,$ $x_n^*(S)$ u kojima se te vrednosti dostižu.
- **Druga faza:** Formiranje optimalnog niza $x_1^*, x_2^*, ..., x_n^*$ i to na sledeći način:

$x_n^* = x_n^*(S)$, $x_i^* = x_i^*(S - a_n(x_n^*) - a_{n-1}(x_{n-1}^*) - \cdots - a_{i+1}(x_{i+1}^*))$, za $i = n-1, ..., 1$.

8.1 Problem ranca

Problem ranca (eng. Knapsack problem) spada u probleme složene raspodele jednorodnog resursa. To je jedan od najpoznatijih problema dinamičkog programiranja i jedan od najviše istraživanih problema u kombinatornoj optimizaciji. Ovaj problem ima nekoliko varijanti. Ipak, za sve njih je karakteristično da postoji n predmeta i svaki predmet ima pridruženu vrednost v_i i težinu w_i, za sve $1 \leqslant i \leqslant n$ (pri čemu su sve ove vrednosti celobrojne). Cilj je sakupiti određen broj predmeta, tako da vrednost ranca bude maksimalna, ali da težina ne pređe zadati kapacitet W. Dakle, potrebno je odrediti $T \subseteq \{1, 2, ..., n\}$ tako da

$$\max \quad \sum_{i \in T} v_i$$
$$\sum_{i \in T} w_i \leqslant W$$
$$x_i \in \{0, 1\}.$$

Ovakav problem se još naziva i $0-1$ problem (eng. $0-1$ Knapsack problem) jer za svaki predmet postoje dve mogućnosti: ili se bira ili ne.

Ranac koji je optimalno popunjen ne mora biti popunjen do kraja. Preciznije, zbir težina uzetih predmeta ne mora biti jednak kapacitetu ranca, već je bitno da taj zbir ne prelazi zadati kapacitet ranca, a da zbir vrednosti tih predmeta bude maksimalan.

Jedno rešenje podrazumeva proveravanje svih mogućnosti - svih 2^n mogućih podskupova T. Takav pristup problemu zahteva eksponencijalno vreme, pa je zbog toga bolje primeniti tehnike dinamičkog programiranja.

Ideja dinamičkog programiranja:

1. Podela problema na potprobleme.
2. Rešavanje potproblema, pri čemu se svaki problem rešava tačno jednom i ta rešenja se čuvaju u tabeli.
3. Kombinovanje rešenja potproblema kako bi se dobilo rešenje polaznog problema.

Postoje dve verzije ovog problema:

- sa ponavljanjem - moguće je uzeti neograničen broj predmeta svake vrste.
- bez ponavljanja - na raspolaganju je tačno jedan primerak svakog predmeta.

8.1.1 Problem ranca sa ponavljanjem

Provalnik sa rancem kapaciteta W ušao je u prostoriju u kojoj se čuvaju vredni predmeti. U prostoriji ima ukupno n različitih predmeta, pri čemu je svaki tip predmeta raspoloživ u velikoj količini (više nego što može da stane u ranac). Za svaki predmet poznata je njegova vrednosti v_i i težina w_i, $i = 1, 2, ..., n$. Sve navedene vrednosti su celobrojne. Provalnik želi da napuni ranac najvrednijim sadržajem. Potrebno je odrediti predmete koje treba staviti u ranac, kao i njihovu zbirnu vrednost.

Neka je $K(w)$ maksimalna vrednost ranca kapaciteta w, $w \leqslant W$. Ako optimalno rešenje za $K(w)$ uključuje $i-$ti predmet onda se uklanjanjem ove stavke iz ranca dobija optimalno rešenje za $K(w - w_i)$. Odatle sledi da je

$$K(w) = K(w - w_i) + v_i, \text{ za neko } 1 \leqslant i \leqslant n.$$

Kako nije poznato o kom predmetu i se radi, potrebno je ispitati sve mogućnosti.

$$K(w) = \max_{\substack{1 \leqslant i \leqslant n \\ w_i \leqslant W}} K(w - w_i) + v_i$$
$$K(0) = 0.$$

Primer 8.1. Neka je kapacitet ranca $10kg$ i u tabeli su date vrednosti i težine 4 predmeta. Odrediti optimalno popunjavanje ranca pri čemu je dozvoljeno ponavljanje predmeta.

i	1	2	3	4
v_i [\$]	30	14	16	9
w_i $[kg]$	6	3	4	2

Rešenje:

$K(0) = 0$

$K(1) = 0$

$K(2) = \max\{K(2-2) + v_4\} = K(0) + 9 = 9$

$K(3) = \max\{K(3-3) + v_2, K(3-2) + v_4\} = \max\{0 + 14, 0 + 9\} = 14$

$K(4) = \max\{K(4-3) + v_2, K(4-4) + v_3, K(4-2) + v_4\} =$
$= \max\{K(1) + 14, K(0) + 16, K(2) + 9\} = \max\{0 + 14, 0 + 16, 9 + 9\} = 18$

$K(5) = \max\{K(5-3) + v_2, K(5-4) + v_3, K(5-2) + v_4\} =$
$= \max\{K(2) + 14, K(1) + 16, K(3) + 9\} = \max\{9 + 14, 0 + 16, 14 + 9\} = 23$

$K(6) = \max\{K(6-6) + v_1, K(6-3) + v_2, K(6-4) + v_3, K(6-2) + v_4\} =$
$= \max\{K(0) + 30, K(3) + 14, K(2) + 16, K(4) + 9\} =$
$= \max\{0 + 30, 14 + 14, 9 + 16, 18 + 9\} = \max\{30, 28, 25, 27\} = 30$

$K(7) = \max\{K(7-6) + v_1, K(7-3) + v_2, K(7-4) + v_3, K(7-2) + v_4\} =$
$= \max\{K(1) + 30, K(4) + 14, K(3) + 16, K(5) + 9\} =$
$= \max\{0 + 30, 18 + 14, 14 + 16, 23 + 9\} = \max\{30, 32, 30, 32\} = 32$

$K(8) = \max\{K(8-6) + v_1, K(8-3) + v_2, K(8-4) + v_3, K(8-2) + v_4\} =$
$= \max\{K(2) + 30, K(5) + 14, K(4) + 16, K(6) + 9\} =$
$= \max\{9 + 30, 23 + 14, 18 + 16, 30 + 9\} = \max\{39, 37, 34, 39\} = 39$

$K(9) = \max\{K(9-6) + v_1, K(9-3) + v_2, K(9-4) + v_3, K(9-2) + v_4\} =$
$= \max\{K(3) + 30, K(6) + 14, K(5) + 16, K(7) + 9\} =$
$= \max\{14 + 30, 30 + 14, 23 + 16, 32 + 9\} = \max\{44, 44, 39, 41\} = 44$

$$K(10) = \max\{K(10-6)+v_1, K(10-3)+v_2, K(10-4)+v_3, K(10-2)+v_4\} =$$
$$= \max\{K(4)+30, K(7)+14, K(6)+16, K(8)+9\} =$$
$$= \max\{18+30, 32+14, 30+16, 39+9\} = \max\{48, 46, 46, 48\} = 48.$$

Optimalna vrednost ranca iznosi \$48. Da bi se odredilo koji predmeti se nalaze u rancu, kreće se od $K(10)$.

- **Korak 1:** Maksimum se dostiže za prvi (v_1) i četvrti (v_4) predmet. Proizvoljno se može uzeti prvi predmet. Njegova težina iznosi $6kg$, pa se u nastavku posmatra $K(10-6) = K(4)$.

- **Korak 2:** Maksimum za posmatranu vrednost $K(4)$ se dostiže za četvrti predmet (v_4). Njegova težina iznosi $2kg$, pa se u nastavku posmatra $K(4-2) = K(2)$.

- **Korak 3:** Maksimum za posmatranu vrednost se dostiže za četvrti predmet (v_4). Ovim postupkom smo stigli do kraja, jer je ukupna težina izabranih predmeta $6kg + 2kg + 2kg = 10kg$.

8.1.2 Problem ranca bez ponavljanja

Ukoliko se nametne uslov da je svaki predmet dostupan u jednom primerku, odnosno svaki predmet je moguće izabrati najviše jednom, dobija se problem ranca bez ponavljanja.

Neka je poznato optimalno popunjavanje ranca kapaciteta W. Ako u tom popunjavanju ne učestvuje $n-$ti predmet, onda je to popunjavanje optimalno i za ranac kapaciteta W i prvih $n-1$ predmeta. Sa druge strane, ako u popunjavanju učestvuje $n-$ti predmet, onda ostali predmeti iz ranca čine optimalno popunjavanje za ranac kapaciteta $W - w_i$ i prvih $n-1$ predmeta. Na osnovu ovoga se može zaključiti da problem ima optimalnu podstrukturu i da rešenje zavisi od dva parametra: prvi je kapacitet ranca, a drugi broj parametara.

Neka je $K(i, w)$ maksimalna vrednost koja se postiže rancem kapaciteta w i stavkama $1, 2, ..., i$. Ovim vrednostima se popunjava tabela formata $(n+1) \times (W+1)$. Tada je konačno rešenje $K(n, W)$.

Algoritam DP

Korak 1: Inicijalizacija.

$$K(0, w) = 0, \text{ za sve } 0 \leqslant w \leqslant W,$$
$$K(w, i) = -\infty \text{ za sve } w < 0.$$

Korak 2: Definisanje pravila po kom se popunjava tabela.
Postoje dve mogućnosti za svaki predmet i, $i = 1, 2, ..., n$.

1. Optimalno popunjavaje ranca sadrži predmet i (to je moguće jedino ako je $w_i \leqslant w$)
$$K(i, w) = K(i-1, w-w_i) + v_i.$$
2. Optimalno popunjavaje ranca ne sadrži $i-$ti predmet
$$K(i, w) = K(i-1, w).$$

Spajanjem prethodna dva slučaja se dobija

$$K(i,w) = \max\{K(i-1, w-w_i)+v_i, K(i-1,w)\}, \text{ za } 1 \leqslant i \leqslant n \text{ i } 0 \leqslant w \leqslant W. \quad (8.1)$$

$K(i,w)$	$w=0$	1	2	3	$\cdots$	W
$i=0$	0	0	0	0	0	0
1						
2						
$\vdots$						
n						

Slika 10: Na osnovu formule (8.1) u tabeli se popunjavaju vrste redom, jedna za drugom.

Dakle, kada je tabela popunjena, vrednost $K(n, W)$ predstavlja maksimalnu vrednost ranca koja je dobijena odabirom nekih od n predmeta, tako da njihova ukupna težina ne prelazi zadati kapacitet W. Sledeći algoritam daje spisak predmeta koji predstavljaju optimalno popunjavanje ranca.

- Ako je $K(n, W) = K(n-1, W)$ tada $n-$ti predmet nije izabran i potrebno je pomeriti se na polje $K(n-1, W)$ u tabeli.
- Ako je $K(n, W) \neq K(n-1, W)$ onda je $n-$ti predmet izabran i u sledećem koraku se posmatra polje $K(n-1, W-w_n)$ u tabeli.

Ovaj postupak se ponavlja sve dok se ne dođe do nulte vrste u tabeli.

Primer 8.2. [22] Neka su data 3 predmeta čije su težine $3, 2$ i $1kg$ a vrednosti $\$5, \$3$ i $\$4$ respektivno. Ako je kapacitet ranca $W = 5kg$, rešiti problem ranca.

Rešenje: Formira se tabela formata 4×6, koja se popunjava na način koji je opisan u prethodnom algoritmu.

		$K(i,w)$	0	1	2	3	4	5
v_i	w_i	0	0	0	0	0	0	0
5	3	1	0	0	0	5	5	5
3	2	2	0	0	3	5	5	8
4	1	3	0	4	4	7	9	9

$$
\begin{aligned}
K(i,0) &= 0, \text{ za sve } 0 \leqslant i \leqslant 3.\\
K(0,w) &= 0, \text{ za sve } 0 \leqslant w \leqslant 5.\\
K(1,1) &= 0\\
K(1,2) &= 0\\
K(1,3) &= \max\{K(0,0)+v_1, K(0,3)\} = \max\{0+5,0\} = 5\\
K(1,4) &= \max\{K(0,1)+v_1, K(0,4)\} = \max\{0+5,0\} = 5\\
K(1,5) &= \max\{K(0,2)+v_1, K(0,5)\} = \max\{0+5,0\} = 5\\
K(2,1) &= \max\{0\} = 0\\
K(2,2) &= \max\{K(1,0)+v_2, K(2,1)\} = \max\{0+3,0\} = 3\\
K(2,3) &= \max\{K(1,0)+v_1, K(1,1)+v_2, K(1,3)\} = \max\{0+5,0+3,5\} = 5\\
K(2,4) &= \max\{K(1,1)+v_1, K(1,2)+v_2, K(1,4)\} = \max\{0+5,0+3,5\} = 5\\
K(2,5) &= \max\{K(1,2)+v_1, K(1,3)+v_2, K(1,5)\} = \max\{0+5,5+3,5\} = 8\\
K(3,1) &= \max\{K(2,0)+v_3, K(2,1)\} = \max\{0+4,0\} = 4\\
K(3,2) &= \max\{K(2,0)+v_2, K(2,1)+v_3\} = \max\{0+3,0+4\} = 4\\
K(3,3) &= \max\{K(2,0)+v_1, K(2,1)+v_2, K(2,2)+v_3\} =\\
&= \max\{0+5,0+3,3+4\} = 7\\
K(3,4) &= \max\{K(2,1)+v_1, K(2,2)+v_2, K(2,3)+v_3\} =\\
&= \max\{0+5,3+3,5+4\} = 9\\
K(3,5) &= \max\{K(2,2)+v_1, K(2,3)+v_2, K(2,4)+v_3\} =\\
&= \max\{3+5,5+3,5+4\} = 9
\end{aligned}
$$

U nastavku se određuje optimalno popunjavanje ranca.

- **Korak 1:** $K(3,5) \neq K(2,5) \rightarrow$ predmet broj 3 je izabran.
  Prelazi se na polje $K(2,4)$ u tabeli.
- **Korak 2:** $K(2,4) = K(1,4) \rightarrow$ predmet broj 2 nije izabran.
  Prelazi se na polje $K(1,4)$ u tabeli.
- **Korak 3:** $K(1,4) \neq K(0,4) \rightarrow$ predmet broj 1 je izabran.
  $\Rightarrow$ kraj algoritma.

		$K(i,w)$	0	1	2	3	4	5
v_i	w_i	0	0	0	0	0	0	0
5	3	1	0	0	0	5	5	5
3	2	2	0	0	3	5	5	8
4	1	3	0	4	4	7	9	9

Maksimalna vrednost ranca je \$9 i on sadrži predmete 1 i 3.

Primer 8.3. [23] Neka kapacitet ranca iznosi $10kg$, i neka su data 4 predmeta čije su težine i vrednosti date u tabeli. Odrediti najbolje popunjavanje ranca kapaciteta $W = 10kg$.

i	1	2	3	4
v_i [\$]	4	8	7	9
w_i $[kg]$	1	4	5	6

Rešenje. Formira se tabela formata 5×11, koja se popunjava na isti način kao i prethodnom zadatku.

		$K(i,w)$	0	1	2	3	4	5	6	7	8	9	10
v_i	w_i	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	0	4	4	4	4	4	4	4	4	4	4
8	4	2	0	4	4	4	8	12	12	12	12	12	12
7	5	3	0	4	4	4	8	12	12	12	12	15	19
9	6	4	0	4	4	4	8	12	12	13	13	15	19

Maksimalna vrednost ranca iznosi \$19 i on sadrži predmete $1, 2$ i 3.

Rešenje problema pomoću programskog paketa Matlab

```
% INPUT:        w - vektor tezina predmeta
%               v - vektor vrednosti predmeta
%               W - kapacitet ranca
% OUTPUT:       k - maksimalna vrednost ranca
%               r - vektor koji predstavlja spisak predmeta
%                   koji su u rancu.

function [k r] = knapsack(w, v, W)
    % Provera da li su w i v pozitivni celobrojni vektori
       i da li je W pozitivan ceo broj
    if ~all(w>0 & floor(w)==w) || ~all(v>0 & floor(v)==v)
   || ~(W>0 & floor(W)==W)
    error('Pogresan unos!');
    end
    % Provera da li su dimenzije vektora w i v jednake
    if numel(w) ~= numel(v)
        error('Uneti vektori nisu iste dimenzije!');
    end

    % Resenje problema
    K = zeros(length(w)+1,W+1); % Nula matrica
    for j = 1:length(w)
        for Y = 1:W
            if w(j) > Y
                K(j+1,Y+1) = K(j,Y+1);
            else
                K(j+1,Y+1) = ...
                    max( K(j,Y+1), v(j) + K(j,Y-w(j)+1));
            end
        end
    end
   k = K(end,end);

   % Odredjivanje predmeta koji popunjavaju ranac pomocu
   principa unazad
   r = zeros(length(w),1); % vektor nula
   a = k; % a je maksimalna vrednost ranca
   j = length(w);
```

```
    Y = W;
    while a > 0
        while K(j+1,Y+1) == a
            j = j - 1;
        end
        j = j + 1; % Ovaj predmet mora biti u rancu
        r(j) = 1;
        Y = Y - w(j);
        j = j - 1;
        a = K(j+1,Y+1);
    end
     r=find(r); % vraca pozicije elemenata razlicitih od 0
     r=r';
end
```

Primer 6.2. Izvršavanjem prethodnog koda za $w = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}, v = \begin{bmatrix} 5 & 4 & 3 \end{bmatrix}$ i $W = 5$ dobija se rezultat: $k = \$9$ i $r = \begin{bmatrix} 1 & 3 \end{bmatrix}$, pri čemu je k optimalna vrednost ranca, a r predstavlja spisak predmeta koji su u rancu. Dakle, optimalna vrednosti ranca iznosi $k = 9$ i u njemu se nalaze predmeti 1 i 3.

Primer 6.3. Za $w = \begin{bmatrix} 1 & 4 & 5 & 6 \end{bmatrix}, v = \begin{bmatrix} 4 & 8 & 7 & 9 \end{bmatrix}$ i $W = 10$ se dobija rešenje: $k = 19$ i $r = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$. Može se zaključiti da optimalna vrednost ranca iznosi \$19 i on sadrži predmete $1, 2$ i 3.

9 Transportni problem

Proučavanje transportnog problema primenom analitičkih metoda datira iz perioda pedesetih godina prošlog veka. Transportni problem je prvi put uočen u radovima ruskog matematičara L. V. Kantoroviča "Matematičke metode u organizaciji i planiranju proizvodnje" iz 1939. godine. Nakon toga je američki matematičar Hitchock 1941. godine formulisao i rešio model transportnog problema u radu "Distribucija proizvoda iz nekoliko izvora do brojnih lokaliteta" ("The Distribution of a Product from Several Sources to Numerous Localities"). Nezavisno od Hitchocka 1947. godine T. C. Koopmans je predstavio studiju nazvanu "Optimalno iskorišćavanje transportnog sistema" ("Optimum Utilization of the Transportation System"). Razvojem metodologije linearnog programiranja pokazano je da je transportni problem specijalni slučaj problema linearnog programiranja. Tipičan transportni problem je problem raspodele robe od proizvođača do potrošača, uz uslov da troškovi transporta robe budu minimalni. Danas se mnogi zadaci svode na rešavanje neke varijante transportnog problema, kao na primer problemi optimalnog razmeštaja mašina, postrojenja, skladišta, servisa ili energetskih objekata, zadaci optimalne raspodele prevoznih sredstava na korisnike i slično.

9.1 Dinamički transportni problem

Specijalnu klasu transportnih problema (takozvani nelinearni transportni problemi) čine oni problemi kod kojih se prevoz tereta obavlja transportnim sredstvom koje snabdeva robom više odredišta iz samo jednog ishodišta. Kriterijum optimalnosti se može definisati preko minimalne ukupne dužine puta, minimalnih troškova ili vremena putovanja. Osnovna ideja je da se problem dekomponuje u konačan niz etapa kako bi se mogao rešiti dinamičkim programiranjem. U slučaju da je teret nehomogen, potrebno je da teret koji se prevozi bude maksimalne vrednosti (cene, prioriteta) uz što veće iskorišćenje kapaciteta datog transportnog sredstva.

Neka je dato transportno sredstvo (kamion, vagon, brod i slično) kapaciteta (nosivosti) Q mernih jedinica. Takođe je dato n različitih tipova predmeta koje je potrebno utovariti i svi oni imaju različitu vrednost. Pod pojmom "vrednost" se najčešće podrazumeva cena ili prioritet robe i označava se sa $C_i,\ i = 1, 2, ..., n$. Neka je P_i težina predmeta $i-$tog tipa, a x_i broj utovarenih predmeta $i-$tog tipa.

Matematički zapis dinamičkog transportnog problema

$$\max \quad F(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} C_i x_i$$

$$\sum_{i=1}^{n} P_i x_i \leqslant Q, \quad x_i = 0, 1, 2, ...$$

Prilikom utovara samo predmeta prvog tipa maksimalna vrednost utovara je

$$f_1(Q) = \max C_1 x_1,$$

uz sledeće ograničenje

$$P_1 x_1 \leqslant Q.$$

Uz ovakvo ograničenje jasno je da je broj predmeta prvog tipa utovarenih u transportno spredstvo dat najvećim celim delom koji ne prelazi $\frac{Q}{P_1}$, odnosno

$$x_1 = \left\lfloor \frac{Q}{P_1} \right\rfloor.$$

Dakle, važi $f_1(Q) = \left\lfloor \frac{Q}{P_1} \right\rfloor C_1$.

Pri utovaru predmeta prvog i drugog tipa maksimalna vrednost utovara se označava sa $f_2(Q)$. Pod pretpostavkom da je utovareno x_2 predmeta drugog tipa, može se zaključiti da težina predmeta drugog tipa ne sme preći iznos $Q - P_2 x_2$. U tom slučaju vrednost utovara iznosi

$$C_2 x_2 + f_1(Q - P_2 x_2).$$

Maksimalna vrednost utovara iznosi

$$f_2(Q) = \max_{0 \leq x_2 \leq \lfloor \frac{Q}{P_2} \rfloor} \{C_2 x_2 + + f_1(Q - P_2 x_2)\}.$$

Analogno se nakon n koraka dolazi do relacije

$$f_n(Q) = \max_{0 \leq x_n \leq \lfloor \frac{Q}{P_n} \rfloor} \{C_n x_n + + f_{n-1}(Q - P_n x_n)\},$$

pri čemu je:

$f_n(Q)$ maksimalna vrednost utovara,gde je uzeto u obzir svih n predmeta,

$C_n x_n$ je vrednost koja odgovara utovaru x_n predmeta $n-$tog tipa,

$f_{n-1}(Q - P_n x_n)$ maksimalna vrednost utovara, pri čemu je uzeto u obzir $n-1$ prvih predmeta i njihova ukupna težina ne prelazi $Q - P_n x_n$.

Primer 9.1. [5] Zadate su 4 vrste roba pakovane u pakete u sledećim težinskim ili drugim jedinicama: $P_1 = 11,\ P_2 = 9,\ P_3 = 7, P_4 = 6$ (tona, m^3, komada i slično). Za svaku robu je od strane stručnih lica ocenjena vrednost (prioritet) i iznosi: $C_1 = 40,\ C_2 = 38,\ C_3 = 26$ i $C_4 = 19$. Poznato je da je transportno sredstvo nosivosti 25 težinskih jedinica. Zadatak je maksimizacija vrednosti prevoza uz što veće iskorišćenje kapaciteta.

Rešenje: Zadatak je potrebno podeliti u nekoliko etapa.

- **Etapa 1:** Neka se u transportno sredstvo utovaraju samo predmeti prvog tipa, odnosno roba koja je pakovana u pakete težine $P_1 = 11$ težinskih jedinica. Maksimalni broj predmeta tipa 1 koje je moguće utovariti iznosi $\left\lfloor \frac{Q}{P_1} \right\rfloor = \left\lfloor \frac{25}{11} \right\rfloor = 2$ komada. Dakle, $x_1 \in \{0, 1, 2\}$.

  Ako je $0 \leq Q \leq 10$ tada nije moguće utovariti ni jedan predmet tipa 1 (jer je težina tog predmeta $P_1 = 11$).

  Ako je $11 \leq Q \leq 21$ moguće je utovariti jedan predmet tipa 1.

  Ako je $22 \leq Q \leq 25$ moguće je utovariti dva predmeta tipa 1.

  Navedeni rezultati su dati u tabeli.

Q	$f_1(Q) = \left\lfloor \frac{Q}{P_1} \right\rfloor C_1$	$x_1 = \left\lfloor \frac{Q}{P_1} \right\rfloor$
$0 - 10$	0	0
$11 - 21$	40	1
$22 - 25$	80	2

- **Etapa 2:** U ovoj etapi se u transportno sredstvo utovaraju predmeti prvog i drugog tipa. S obzirom da je $P_2 = 9$, $\left\lfloor \frac{Q}{P_2} \right\rfloor = \left\lfloor \frac{25}{9} \right\rfloor = 2$, sledi $x_2 \in \{0, 1, 2\}$. Kako bi se odredilo $f_2(Q)$ potrebno je kapacitet transportnog sredstva podeliti na intervale u zavisnosti od vrednosti parametara P_1 i P_2 i na osnovu relacije

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\}$$

  i činjenice da $x_2 \in \{0, 1, 2\}$ odrediti vrednost funkcije $f_2(Q)$.

  - Ako je $0 \leq Q \leq 8$ nije moguće utovariti predmete ni prvog ni drugog tipa.

- Ako je $9 \leq Q \leq 10$ sledi

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\} = \max_{0 \leq x_2 \leq 1} \{38 x_2 + f_1(Q - 9x_2)\} =$$

$$= \max \begin{cases} 38 \cdot 0 + f_1(Q - 9 \cdot 0) \\ 38 \cdot 1 + f_1(Q - 9 \cdot 1) \end{cases} = \max \begin{cases} 0 + 0 \\ 38 + 0 \end{cases} = 38.$$

To znači da je pri ovom kapacitetu potrebno utovariti samo jedan predmet tipa 2.

- Ako je $11 \leq Q \leq 17$ tada je

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\} = \max_{0 \leq x_2 \leq 1} \{38 x_2 + f_1(Q - 9x_2)\} =$$

$$= \max \begin{cases} 38 \cdot 0 + f_1(Q - 9 \cdot 0) \\ 38 \cdot 1 + f_1(Q - 9 \cdot 1) \end{cases} = \max \begin{cases} 0 + f_1(Q) \\ 38 + f_1(Q - 9) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 38 + 0 \end{cases} = 40.$$

Pri ovom kapacitetu potebno je utovariti samo jedan predmet tipa 1.

- Ako je $18 \leq Q \leq 19$ sledi

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\} = \max_{0 \leq x_2 \leq 2} \{38 x_2 + f_1(Q - 9x_2)\} =$$

$$= \max \begin{cases} 38 \cdot 0 + f_1(Q - 9 \cdot 0) \\ 38 \cdot 1 + f_1(Q - 9 \cdot 1) \\ 38 \cdot 2 + f_1(Q - 9 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_1(Q) \\ 38 + f_1(Q - 9) \\ 76 + f_1(Q - 18) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 38 + 0 \\ 76 + 0 \end{cases} = 76.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 2.

- Ako je $20 \leq Q \leq 21$ tada je

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\} = \max_{0 \leq x_2 \leq 2} \{38 x_2 + f_1(Q - 9x_2)\} =$$

$$= \max \begin{cases} 38 \cdot 0 + f_1(Q - 9 \cdot 0) \\ 38 \cdot 1 + f_1(Q - 9 \cdot 1) \\ 38 \cdot 2 + f_1(Q - 9 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_1(Q) \\ 38 + f_1(Q - 9) \\ 76 + f_1(Q - 18) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 38 + 40 \\ 76 + 0 \end{cases} = 78.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti jedan predmet tipa 1 i jedan predmet tipa 2.

- Ako je $22 \leq Q \leq 25$ sledi

$$f_2(Q) = \max_{0 \leq x_2 \leq \left\lfloor \frac{Q}{P_2} \right\rfloor} \{C_2 x_2 + f_1(Q - P_2 x_2)\} = \max_{0 \leq x_2 \leq 2} \{38 x_2 + f_1(Q - 9x_2)\} =$$

$$= \max \begin{cases} 38 \cdot 0 + f_1(Q - 9 \cdot 0) \\ 38 \cdot 1 + f_1(Q - 9 \cdot 1) \\ 38 \cdot 2 + f_1(Q - 9 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_1(Q) \\ 38 + f_1(Q - 9) \\ 76 + f_1(Q - 18) \end{cases} =$$

$$= \max \begin{cases} 0 + 80 \\ 38 + 40 \\ 76 + 0 \end{cases} = 80.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 1.

Dakle, kada se za svaki interval izvrši proračun dolazi se do zaključka da se najveća vrednost $f_2(25)$ dobija prilikom utovara u transportno sredstvo dva predmeta prvog tipa.

Q	$f_2(Q)$	x_2	x_1
$0 - 8$	0	0	0
$9 - 10$	38	1	0
$11 - 17$	40	0	1
$18 - 19$	76	2	0
$20 - 21$	78	1	1
$22 - 25$	80	0	2

Etapa 3: Sada se razmatra mogućnost utovara predmeta prvog, drugog i trećeg tipa. Kako je $P_3 = 7$ i $\left\lfloor \frac{Q}{P_3} \right\rfloor = \left\lfloor \frac{25}{7} \right\rfloor = 3$, sledi da je $x_3 \in \{0, 1, 2, 3\}$.

- Ako je $0 \leq Q \leq 6$ nije moguće utovariti predmete ni prvog, ni drugog ni trećeg tipa.
- Ako je $7 \leq Q \leq 8$ sledi

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 1} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \end{cases} =$$

$$= \max \begin{cases} 0 + 0 \\ 26 + 0 \end{cases} = 26.$$

Potrebno je utovariti jedan predmet tipa 3.

- Ako je $9 \leq Q \leq 10$ tada je

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 1} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \end{cases} =$$

$$= \max \begin{cases} 0 + 38 \\ 26 + 0 \end{cases} = 38.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti jedan predmet tipa 2.

- Ako je $11 \leq Q \leq 13$ sledi

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 1} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 26 + 0 \end{cases} = 40.$$

Potrebno je utovariti jedan predmet tipa 1.

- Ako je $14 \leq Q \leq 15$ tada je

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 2} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \\ 26 \cdot 2 + f_2(Q - 7 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \\ 52 + f_2(Q - 14) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 26 + 0 \\ 52 + 0 \end{cases} = 52.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 3.

- Ako je $16 \leq Q \leq 17$ sledi

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 2} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \\ 26 \cdot 2 + f_2(Q - 7 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \\ 52 + f_2(Q - 14) \end{cases} =$$

$$= \max \begin{cases} 0 + 40 \\ 26 + 38 \\ 52 + 0 \end{cases} = 64.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti jedan predmet tipa 2 i jedan predmet tipa 3.

- Ako je $18 \leq Q \leq 19$ tada je

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 2} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \\ 26 \cdot 2 + f_2(Q - 7 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \\ 52 + f_2(Q - 14) \end{cases} =$$

$$= \max \begin{cases} 0 + 76 \\ 26 + 40 \\ 52 + 0 \end{cases} = 76.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 2.

- Ako je $20 \leq Q \leq 21$ sledi

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 2} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \\ 26 \cdot 2 + f_2(Q - 7 \cdot 2) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \\ 52 + f_2(Q - 14) \end{cases} =$$

$$= \max \begin{cases} 0 + 78 \\ 26 + 40 \\ 52 + 0 \end{cases} = 78.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti jedan predmet tipa i jedan predmet tipa 2.

- Ako je $Q = 22$ tada je

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 3} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(22 - 7 \cdot 0) \\ 26 \cdot 1 + f_2(22 - 7 \cdot 1) \\ 26 \cdot 2 + f_2(22 - 7 \cdot 2) \\ 26 \cdot 3 + f_2(22 - 7 \cdot 3) \end{cases} = \max \begin{cases} 0 + f_2(22) \\ 26 + f_2(15) \\ 52 + f_2(8) \\ 78 + f_2(1) \end{cases} =$$

$$= \max \begin{cases} 0 + 80 \\ 26 + 40 \\ 52 + 0 \\ 78 + 0 \end{cases} = 80.$$

Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 1.

- Ako je $23 \leq Q \leq 24$ sledi

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 3} \{26 x_3 + f_1(Q - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(Q - 7 \cdot 0) \\ 26 \cdot 1 + f_2(Q - 7 \cdot 1) \\ 26 \cdot 2 + f_2(Q - 7 \cdot 2) \\ 26 \cdot 3 + f_2(Q - 7 \cdot 3) \end{cases} = \max \begin{cases} 0 + f_2(Q) \\ 26 + f_2(Q - 7) \\ 52 + f_2(Q - 14) \\ 78 + f_2(Q - 21) \end{cases} =$$

$$= \max \begin{cases} 0 + 80 \\ 26 + 40 \\ 52 + 38 \\ 78 + 0 \end{cases} = 90.$$

  Pri ovom kapacitetu transportnog sredstva potrebno je utovariti jedan predmet tipa 2 i dva predmeta tipa 3.

- Ako je $Q = 25$ tada je

$$f_3(Q) = \max_{0 \leq x_3 \leq \left\lfloor \frac{Q}{P_3} \right\rfloor} \{C_3 x_3 + f_2(Q - P_3 x_3)\} = \max_{0 \leq x_3 \leq 3} \{26 x_3 + f_1(25 - 7x_3)\} =$$

$$= \max \begin{cases} 26 \cdot 0 + f_2(25 - 7 \cdot 0) \\ 26 \cdot 1 + f_2(25 - 7 \cdot 1) \\ 26 \cdot 2 + f_2(25 - 7 \cdot 2) \\ 26 \cdot 3 + f_2(25 - 7 \cdot 3) \end{cases} = \max \begin{cases} 0 + f_2(25) \\ 26 + f_2(18) \\ 52 + f_2(11) \\ 78 + f_2(4) \end{cases} =$$

$$= \max \begin{cases} 0 + 80 \\ 26 + 76 \\ 52 + 40 \\ 78 + 0 \end{cases} = 102.$$

  Pri ovom kapacitetu transportnog sredstva potrebno je utovariti dva predmeta tipa 2 i jedan predmet tipa 3.

U narednoj tabeli su prikazane vrednosti utovara za predmete prve, druge i treće vrste.

Q	$f_3(Q)$	x_3	x_2	x_1
$0-6$	0	0	0	0
$7-8$	26	1	0	0
$9-10$	38	0	1	0
$11-13$	40	0	0	1
$14-15$	52	2	0	0
$16-17$	64	1	1	0
$18-19$	76	0	2	0
$20-21$	78	0	1	1
22	80	0	0	2
$23-24$	90	2	1	0
25	102	1	2	0

- **Etapa 4:** Razmatra se mogućnost utovara predmeta prvog, drugog, trećeg i četvrtog tipa. Kako je $P_4 = 6$ i $\left\lfloor \frac{Q}{P_4} \right\rfloor = \left\lfloor \frac{25}{6} \right\rfloor = 4$, sledi da je $x_4 \in \{0, 1, 2, 3, 4\}$.

  - Ako je $0 \leq Q \leq 5$ nije moguće utovariti ni jedan predmet.
  - Ako je $Q = 6$ sledi

$$f_4(Q) = \max_{0 \leq x_4 \leq \left\lfloor \frac{Q}{P_4} \right\rfloor} \{C_4 x_4 + f_3(Q - P_4 x_4)\} = \max \begin{cases} 19 \cdot 0 + f_3(6 - 6 \cdot 0) \\ 19 \cdot 1 + f_3(6 - 6 \cdot 1) \end{cases} =$$

$$= \max \begin{cases} 0 + 0 \\ 19 + 0 \end{cases} = 19,$$

pa je potrebno utovariti jedan predmet tipa 4.

- Ako je $7 \leq Q \leq 8$ tada je

$$f_4(Q) = \max_{0 \leq x_4 \leq \left\lfloor \frac{Q}{P_4} \right\rfloor} \{C_4 x_4 + f_3(Q - P_4 x_4)\} = \max_{0 \leq x_4 \leq 1} \begin{cases} 19 \cdot x_4 + f_3(Q - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(Q - 6 \cdot 1) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(Q) \\ 19 \cdot 1 + f_3(Q - 6) \end{cases} = \max \begin{cases} 0 + 26 \\ 19 + 0 \end{cases} = 26,$$

pa je potrebno utovariti jedan predmet tipa 3.

- Ako je $9 \leq Q \leq 10$ sledi

$$f_4(Q) = \max_{0 \leq x_4 \leq \left\lfloor \frac{Q}{P_4} \right\rfloor} \{C_4 x_4 + f_3(Q - P_4 x_4)\} = \max_{0 \leq x_4 \leq 1} \begin{cases} 19 \cdot x_4 + f_3(Q - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(Q - 6 \cdot 1) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(Q) \\ 19 \cdot 1 + f_3(Q - 6) \end{cases} = \max \begin{cases} 0 + 38 \\ 19 + 0 \end{cases} = 38,$$

pa je potrebno utovariti jedan predmet tipa 2.

- Ako je $11 \leq Q \leq 12$ tada je

$$f_4(Q) = \max_{0 \leq x_4 \leq \left\lfloor \frac{Q}{P_4} \right\rfloor} \{C_4 x_4 + f_3(Q - P_4 x_4)\} = \max_{0 \leq x_4 \leq 1} \begin{cases} 19 \cdot x_4 + f_3(Q - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(Q - 6 \cdot 1) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(Q) \\ 19 \cdot 1 + f_3(Q - 6) \end{cases} = \max \begin{cases} 0 + 40 \\ 19 + 0 \end{cases} = 40,$$

pa je potrebno utovariti jedan predmet tipa 1.

- Ako je $Q = 13$ sledi

$$f_4(13) = \max_{0 \leq x_4 \leq \left\lfloor \frac{13}{P_4} \right\rfloor} \{C_4 x_4 + f_3(13 - P_4 x_4)\} = \max_{0 \leq x_4 \leq 2} \begin{cases} 19 \cdot x_4 + f_3(13 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(13 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(13 - 6 \cdot 2) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(13) \\ 19 \cdot 1 + f_3(7) \\ 19 \cdot 2 + f_3(1) \end{cases} = \max \begin{cases} 0 + 40 \\ 19 + 26 \\ 38 + 0 \end{cases} = 45,$$

pa je potrebno utovariti jedan predmet tipa 3 i jedan predmet tipa 4.

- Ako je $Q = 14$ tada je

$$f_4(14) = \max_{0 \leq x_4 \leq \left\lfloor \frac{14}{P_4} \right\rfloor} \{C_4 x_4 + f_3(14 - P_4 x_4)\} = \max_{0 \leq x_4 \leq 2} \begin{cases} 19 \cdot x_4 + f_3(14 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(14 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(14 - 6 \cdot 2) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(14) \\ 19 \cdot 1 + f_3(8) \\ 19 \cdot 2 + f_3(2) \end{cases} = \max \begin{cases} 0 + 52 \\ 19 + 26 \\ 38 + 0 \end{cases} = 52,$$

pa je potrebno utovariti dva predmeta tipa 3.

- Ako je $Q = 15$ sledi

$$f_4(15) = \max_{0 \le x_4 \le \left\lfloor \frac{15}{P_4} \right\rfloor} \{C_4 x_4 + f_3(15 - P_4 x_4)\} = \max_{0 \le x_4 \le 2} \begin{cases} 19 \cdot x_4 + f_3(15 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(15 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(15 - 6 \cdot 2) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(15) \\ 19 \cdot 1 + f_3(9) \\ 19 \cdot 2 + f_3(3) \end{cases} = \max \begin{cases} 0 + 52 \\ 19 + 38 \\ 38 + 0 \end{cases} = 57,$$

pa je potrebno utovariti jedan predmet tipa 2 i jedan predmet tipa 4.

- Ako je $Q = 16$ tada je

$$f_4(16) = \max_{0 \le x_4 \le \left\lfloor \frac{16}{P_4} \right\rfloor} \{C_4 x_4 + f_3(16 - P_4 x_4)\} = \max_{0 \le x_4 \le 2} \begin{cases} 19 \cdot x_4 + f_3(16 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(16 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(16 - 6 \cdot 2) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(16) \\ 19 \cdot 1 + f_3(10) \\ 19 \cdot 2 + f_3(4) \end{cases} = \max \begin{cases} 0 + 64 \\ 19 + 38 \\ 38 + 0 \end{cases} = 64,$$

pa je potrebno utovariti jedan predmet tipa 2 i jedan predmet tipa 3.

- Ako je $Q = 17$ sledi

$$f_4(17) = \max_{0 \le x_4 \le \left\lfloor \frac{17}{P_4} \right\rfloor} \{C_4 x_4 + f_3(17 - P_4 x_4)\} = \max_{0 \le x_4 \le 2} \begin{cases} 19 \cdot x_4 + f_3(17 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(17 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(17 - 6 \cdot 2) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(17) \\ 19 \cdot 1 + f_3(11) \\ 19 \cdot 2 + f_3(5) \end{cases} = \max \begin{cases} 0 + 64 \\ 19 + 40 \\ 38 + 0 \end{cases} = 64,$$

pa je potrebno utovariti jedan predmet tipa 2 i jedan predmet tipa 3.

- Ako je $Q = 18$ tada je

$$f_4(18) = \max_{0 \le x_4 \le \left\lfloor \frac{18}{P_4} \right\rfloor} \{C_4 x_4 + f_3(18 - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(18 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(18 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(18 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(18 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(18) \\ 19 \cdot 1 + f_3(12) \\ 19 \cdot 2 + f_3(6) \\ 19 \cdot 3 + f_3(0) \end{cases} = \max \begin{cases} 0 + 76 \\ 19 + 40 \\ 38 + 0 \\ 57 + 0 \end{cases} = 76,$$

pa je potrebno utovariti dva predmeta tipa 2.

- Ako je $Q = 19$ sledi

$$f_4(19) = \max_{0 \le x_4 \le \left\lfloor \frac{19}{P_4} \right\rfloor} \{C_4 x_4 + f_3(19 - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(19 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(19 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(19 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(19 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(19) \\ 19 \cdot 1 + f_3(13) \\ 19 \cdot 2 + f_3(7) \\ 19 \cdot 3 + f_3(1) \end{cases} = \max \begin{cases} 0 + 76 \\ 19 + 40 \\ 38 + 26 \\ 57 + 0 \end{cases} = 76,$$

pa je potrebno utovariti dva predmeta tipa 2.

- Ako je $Q = 20$ tada je

$$f_4(20) = \max_{0 \le x_4 \le \left\lfloor \frac{20}{P_4} \right\rfloor} \{C_4 x_4 + f_3(Q - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(20 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(20 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(20 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(20 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(20) \\ 19 \cdot 1 + f_3(14) \\ 19 \cdot 2 + f_3(8) \\ 19 \cdot 3 + f_3(2) \end{cases} = \max \begin{cases} 0 + 78 \\ 19 + 52 \\ 38 + 26 \\ 57 + 0 \end{cases} = 78,$$

pa je potrebno utovariti jedan predmet tipa 1 i jedan predmet tipa 2.

- Ako je $Q = 21$ sledi

$$f_4(21) = \max_{0 \le x_4 \le \left\lfloor \frac{21}{P_4} \right\rfloor} \{C_4 x_4 + f_3(21 - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(21 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(21 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(21 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(21 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(21) \\ 19 \cdot 1 + f_3(15) \\ 19 \cdot 2 + f_3(9) \\ 19 \cdot 3 + f_3(3) \end{cases} = \max \begin{cases} 0 + 78 \\ 19 + 52 \\ 38 + 38 \\ 57 + 0 \end{cases} = 78,$$

pa je potrebno utovariti jedan predmet tipa 1 i jedan predmet tipa 2.

- Ako je $Q = 22$ tada je

$$f_4(22) = \max_{0 \le x_4 \le \left\lfloor \frac{22}{P_4} \right\rfloor} \{C_4 x_4 + f_3(22 - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(22 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(22 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(22 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(22 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(22) \\ 19 \cdot 1 + f_3(16) \\ 19 \cdot 2 + f_3(10) \\ 19 \cdot 3 + f_3(4) \end{cases} = \max \begin{cases} 0 + 80 \\ 19 + 64 \\ 38 + 38 \\ 57 + 0 \end{cases} = 83,$$

pa je potrebno utovariti po jedan predmet tipa $2, 3$ i 4.

- Ako je $Q = 23$ sledi

$$f_4(23) = \max_{0 \le x_4 \le \left\lfloor \frac{23}{P_4} \right\rfloor} \{C_4 x_4 + f_3(23 - P_4 x_4)\} = \max_{0 \le x_4 \le 3} \begin{cases} 19 \cdot x_4 + f_3(23 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(23 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(23 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(23 - 6 \cdot 3) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(23) \\ 19 \cdot 1 + f_3(17) \\ 19 \cdot 2 + f_3(11) \\ 19 \cdot 3 + f_3(5) \end{cases} = \max \begin{cases} 0 + 90 \\ 19 + 64 \\ 38 + 40 \\ 57 + 0 \end{cases} = 90,$$

pa je potrebno utovariti po jedan predmet tipa 2 i dva predmeta tipa 3.

- Ako je $Q = 24$ tada je

$$f_4(24) = \max_{0 \le x_4 \le \left\lfloor \frac{24}{P_4} \right\rfloor} \{C_4 x_4 + f_3(24 - P_4 x_4)\} = \max_{0 \le x_4 \le 4} \begin{cases} 19 \cdot x_4 + f_3(24 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(24 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(24 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(24 - 6 \cdot 3) \\ 19 \cdot x_4 + f_3(24 - 6 \cdot 4) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(24) \\ 19 \cdot 1 + f_3(18) \\ 19 \cdot 2 + f_3(12) \\ 19 \cdot 3 + f_3(6) \\ 19 \cdot 4 + f_3(0) \end{cases} = \max \begin{cases} 0 + 90 \\ 19 + 76 \\ 38 + 40 \\ 57 + 0 \\ 76 + 0 \end{cases} = 95,$$

pa je potrebno utovariti po dva predmeta tipa 2 i jedan predmet tipa 4.

- Ako je $Q = 25$ sledi

$$f_4(25) = \max_{0 \le x_4 \le \left\lfloor \frac{25}{P_4} \right\rfloor} \{C_4 x_4 + f_3(25 - P_4 x_4)\} = \max_{0 \le x_4 \le 4} \begin{cases} 19 \cdot x_4 + f_3(25 - 6 \cdot 0) \\ 19 \cdot x_4 + f_3(25 - 6 \cdot 1) \\ 19 \cdot x_4 + f_3(25 - 6 \cdot 2) \\ 19 \cdot x_4 + f_3(25 - 6 \cdot 3) \\ 19 \cdot x_4 + f_3(25 - 6 \cdot 4) \end{cases} =$$

$$= \max \begin{cases} 19 \cdot 0 + f_3(25) \\ 19 \cdot 1 + f_3(19) \\ 19 \cdot 2 + f_3(13) \\ 19 \cdot 3 + f_3(7) \\ 19 \cdot 4 + f_3(1) \end{cases} = \max \begin{cases} 0 + 102 \\ 19 + 76 \\ 38 + 40 \\ 57 + 26 \\ 76 + 0 \end{cases} = 102,$$

pa je potrebno je utovariti po dva predmeta tipa 2 i jedan predmet tipa 3.

U nastavku je data tabela za vrednosti utovara za predmete sva četiri tipa.

Q	$f_3(Q)$	x_4	x_3	x_2	x_1
$0-5$	0	0	0	0	0
6	19	1	0	0	0
$7-8$	26	0	1	0	0
$9-10$	38	0	0	1	0
$11-12$	40	0	0	0	1
13	45	1	1	0	0
14	52	0	2	0	0
15	57	1	0	1	0
$16-17$	64	0	1	1	0
$18-19$	76	0	0	2	0
$20-21$	78	0	0	1	1
22	83	1	1	1	0
23	90	0	2	1	0
24	95	1	0	2	0
25	102	0	1	2	0

Optimalno rešenje polaznog problema se dobija pri prevoženju dva predmeta tipa 2 i jednog predmeta tipa 1. U ovom slučaju je kapacitet transportnog sredstva u potpunosti iskorišćen ($2 \cdot 9 + 7 = 25$). Takođe, rešenje nije dato samo za transportno sredstvo kapaciteta 25 težinskih jedinica, već i za ona transportna sredstva čija se nosivost kreće u intervalu $0 \leq Q \leq 25$ težinskih jedinica i time je rešen jedan skup sličnih problema. Osim toga moguće je i odrediti kolika je vrednost transporta i koliko je iskorišćenje transportnog sredstva u slučaju da je doneta odluka koja nije optimalna.

10 Zaključak

Dinamičko programiranje predstavlja tehniku rešavanja problema optimizacije koji su u osnovi višeetapni procesi upravljanja. To znači da se proces sastoji od konačnog broja etapa i pri tome je na svakoj etapi potrebno doneti neku odluku. Ova klasa problema optimizacije predstavlja veliki izazov sa numeričkog aspekta, jer je obično vreme koje je potrebno za rešavanje ovih problema eksponencijalna funkcija koja zavisi od broja etapa kao i od broja parametara upravljanja.

Osnovna ideja dinamičkog programiranje je identifikacija potproblema manje veličine koji se potom rešavaju, a zatim kombinuju za rešenje glavnog problema. Svaki potproblem se rašava samo jednom i njegovo rešenje se pamti u memoriji. Dakle, osnovna prednost ovakvog pristupa je to što se vreme potrebno za rešavanje problema drastično smanjuje, na račun memorijskog zauzeća.

Pomoću dinamičkog programiranje efikasno se mogu rešavati brojni problemi koji su prisutni u mnogim područjima prirodnih, društvenih i tehničkih nauka. U radu su predstavljeni mnogobrojni primeri koji ilustruju primenu dinamičkog programiranja.

Literatura

[1] Wayne L. Winston, Jeffrey B. Goldberg. *Operations Research: Applications and Algorithms*, Belmont, CA: Thomson/Brooks/Cole, 2004.

[2] Jeff Erickson, *Algorithms*, University of Illinois, Urbana-Champaign, January 2015.

[3] S. Krčevinac, M. Cangalović, V. Kovačević-Vujčić, M. Martić, M. Vujošević, *Operaciona istraživanja 2*, FON, Beograd, 2004.

[4] Stuart Dreyfus, *Richard Bellman on the Birth of Dynamic Programming*, Operations Research 50(1):48-51, 2002.

[5] Predrag S. Stanimirović, Nebojša V. Stojković, Marko D. Petković, *Matematičko programiranje*, Prirodno-matematički fakultet u Nišu, 2007.

[6] Dejan Bogdanović, Ivan Jovanović, *Operaciona istraživanja 1,* Tehnički fakultet, Univerzitet u Beogradu, 2019.

[7] S.K. Sathua, M.R. Kabat, R. Mohanty, *Lecture notes on design and analysis of algorithms*, University of tehnology, Burla.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, Second Edition, The MIT Press.

[9] Abha Singhal, Priyanka Pandey, *Traveling Salesman Problems by Dynamic Programming Alorithm*, International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-2, Issue-1, January 2016.

[10] Z. Stojaković, I. Bošnjak, *Elementi linearne algebre*, Symbol, Novi Sad, 2010.

[11] Eric V. Denardo, *Dynamic Programming: Models and Applications.*

[12] Vladimir Baltić, *Teorija grafova*, Beograd, 2008.

[13] C. Vasudev, *Graph Theory with Applications*, 2006.

[14] https://www.javatpoint.com/matrix-chain-multiplication-example

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms,* Third Edition The MIT Press Cambridge, Massachusetts London, England.

[16] Dave Mount, *Dynamic Programming: Longest Common Subsequence,* Thursday, Oct 5, 2017.

[17] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* 2007.

[18] Chunchun Zhao, Sartaj Sahni, *Linear space string correction algorithm using the Demirau-Levenshtein distance,* BMC Bioinformatics, 2020.

[19] Bonnie Berger, Michael S. Waterman, Yun William, *Levenshtein Distance, Sequence Comparison and Biological Database Search,* IEEE Transactions on Information Theory, 21 May 2020.

[20] Bo Lincoln, *Dynamic Programming and Time-Varying Delay Systems*, Department of Automatic Control Lund Institute of Technology, Sweden, 2003.

[21] Michael T. Goodrich, Roberto Tamassia, *Algorithm Design and Applications,* Wiley, 2015.

[22] https://www.sanfoundry.com/dynamic-programming-solutions-0-1-knapsack-problem/

[23] Tarequl Islam Sifat, *Revisiting Sparse Dynamic Programming for the 0/1 Knapsack problem*, Department of Computer Science, Colorado State University, Spring 2019.

Biografija

Milica Matijević je rođena 26. avgusta 1997. godine u Sremskoj Mitrovici. Osnovnu školu "Dušan Jerković" u Rumi je završila 2012. godine. Potom je upisala Gimnaziju "Stevan Puzić" u Rumi, opšti smer, koju je završila 2016. godine. Nakon završetka srednje škole upisala je osnovne akademske studije Matematike na Prirodno-matematičkom fakultetu u Novom Sadu. Diplomirala je 2019. godine, kada je i upisala master studije, smer Primenjena matematika. Položila je sve ispite predviđene nastavnim planom i programom i stekla uslov za odbranu master rada.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR
Identifikacioni broj:
IBR
Tip dokumentacije: Monografska dokumentacija
BF
Tip zapisa: Tekstualni štampani materijal
TZ
Vrsta rada: Master rad
VR
Autor: Milica Matijević
AU
Mentor: Prof. dr Sanja Rapajić
MN
Naslov rada: Dinamičko programiranje i primena
NR
Jezik publikacije: srpski (latinica)
JP
Jezik izvoda: s/e
JI
Zemlja publikovanja: Republika Srbija
ZP
Uže geografsko područje: Vojvodina
UGP
Godina: 2021.
GO
Izdavač: autorski reprint
IZ Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4
MA
Fizički opis rada: (10/104/23/28/10)
(broj poglavlja/strana/referenci/tabela/slika)
FO
Naučna oblast: matematika
NO
Naučna disciplina: primenjena matematika
ND

Ključne reči: dinamičko programiranje, višeetapni procesi upravljanja, potproblemi, Belmanov princip optimalnosti.
PO
UDK
Čuva se: u biblioteci Departmana za matematiku i informatiku, Prirodno-matematičkog fakulteta, u Novom Sadu
ČU
Važna napomena:
VN
Izvod: Rad se bavi dinamičkim programiranjem i njegovim primenama. U radu su definisani osnovni pojmovi i data je istorija razvoja dinamičkog programiranja. Predstavljeni su osnovni koncepti dinamičkog programiranja, prosta raspodela jednorodnog resursa, kao i složena raspodela jednorodnog resursa. Primena dinamičkog programiranja je ilustrovana na mnogobrojnim realnim primerima, pri čemu su neki od njih rešeni u programskom paketu Matlab.
Datum prihvatanja teme od strane NN veća:
DP
Datum odbrane:
DO
Članovi komisije:
KO
Predsednik: dr Nenad Teofanov, redovni profesor PMF-a
Član: dr Goran Radojev, docent PMF-a
Član: dr Sanja Rapajić, redovni profesor PMF-a, mentor.

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
KEY WORD DOCUMENTATION

Accession number:
ANO
Identification number:
INO
Document type: monograph type
DT
Type of record: printed text
TR
Contents code: Master thesis
CC
Author: Milica Matijević
AU
Mentor: Prof. Dr. Sanja Rapajić
MN
Title: Dynamic programming and its application
TI
Language of text: Serbian
LT
Language of abstract: Serbian and English
LA
Country of publication: Republic of Serbia
CP
Locality of publication: Vojvodina
LP
Publication year: 2021.
PY
Publisher: author's reprint
PU
Publ. place: Novi Sad, Trg Dositeja Obradovića 4
PP
Physical description: (10/104/23/28/10)
(chapters/pages/literature/tables/pictures)
PD Scientific field: mathematics
SF
Scientific discipline: applied mathematics
SD

Subject/Keywords: dynamic programming, multistage decision process, subproblems, Bellman's principle of optimality.
SKW UC Holding data: The library of the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad
HD
Note:
N
Abstract: The subject of this master thesis is dynamic programming and its applications. The basic terms of dynamic programming, its history and the basic concept of dynamic programming are presented in this thesis. Some real-life problems are presented and solved using techniques of dynamic programming, such as allocation resource problem, longest common subsequence problem, matrix chain multiplication problem and transportation problem. Some of these problems are solved using Matlab.
AB
Accepted by the Scientific Board on:
ASB
Defended:
DE
Thesis defend board:
DB
President: Dr. Nenad Teofanov, Full professor, Faculty of Sciences, University of Novi Sad
Member: Dr. Goran Radojev, Assistant Professor, Faculty of Sciences, University of Novi Sad
Member: Dr. Sanja Rapajić, Full professor, Faculty of Sciences, University of Novi Sad, supervisor.