# Backtracking zadaci

**Zadatak 1.** Na stolu se nalazi $n \leq 100$ domina. Svaka domina može proizvoljno da se rotira. Dve domine mogu da se nadovezu jedna na drugu ako je drugi broj prve domine jednak prvom broju druge domine. Napisati program koji za date domine odredjuje duzinu najduzeg lanca koji od njih moze da se formira.

U prvom redu ulaza nalazi se broj $n$. U svakom od narednih $n$ redova nalaze se po dva broja, i oni predstavljaju brojeve jedne domine. Brojevi na domini mogu biti od $0$ do $6$ (standardne domine).

**Rešenje.** Primetimo da je ukupan broj različitih domina $49$. Ovaj zadatak se rešavanja prolaženjem kroz sve mogućnosti, ali su potrebne optimizacije kako bi program bio dovoljno brz. Prva optimizacija, koja se lako uočava, jeste da prvi put kad se lanac završava brojem $x$, nadovezaćemo na lanac sve domine koje na oba kraja imaju broj $x$. Druga optimizacija je malo teža za uočiti, međutim ona je ključna u ovom zadatku. Označimo sve domine brojevima od $1$ do $n$, tako da svaka domina ima jedinstven broj. Ukoliko postoje $2$ iste domine, neka su označene jedinstvenim brojevima $d_1$ i $d_2$ i neka važi $d_1 < d_2$, i ukoliko domina $d_1$ nije ubačena u trenutni lanac, onda nema svrhe ubacivati ni dominu $d_2$. Ovo opravdavamo time što ukoliko ubacimo dominu $d_2$ pre domine $d_1$, to je isto kao da smo ubacili dominu $d_1$ pre domine $d_2$, te kako ne bismo radili istu stvar $2$ puta, mi ćemo se opredeliti da ukoliko postoji više istih domina prvo ubacimo uvek onu sa manjim brojem kojim je označena.

**Zadatak 2. (USACO)** Na papiru je bila napisana važna poruka, te su krave smislile sjajan način da kriptuju tu poruku. One mogu da ubace slova C, O i W bilo gde u rečenicu tako da je C pre O, i O pre W, te uzmu deo rečenice između slova C i O, i deo rečenice između O i W, i zamene mesta tim delovima rečenice. Npr. neka je početan tekst bio “International Olympiad in Informatics” te kad se stave slova C, O, W na neka mesta u rečenici dobijamo “International **C**Olympiad **O**in Informatics**W**” i posle zamene delova rečenica, tekst izgleda “International **C**in Informatics**O**Olympiad **W**”. Da bi enkriptovanje bilo teže, krave mogu više puta da upotrebe svoj način kriptovanja, svaki put urade kriptovanje na novonastaloj rečenici. Vama je dat kriptovan tekst dužine do $75$ karaktera, i vaš je zadatak da odredite da li je moguće da je početni tekst bio “Begin the Escape execution at the Break of Dawn”.

**Rešenje.** Kako prethodno stanje nije jednoznačno određeno ukoliko postoji više slova C, O ili W, naš zadatak je da prođemo kroz sve mogućnosti, ali naravno optimizacije su neophodne.

**Optimizacija 1.** Primetimo da tekst koji treba da bude početni ima tačno $47$ karaktera, što znači da tekst koji nam je dat mora da ima bar toliko karaktera i da preostali deo mora da bude deljiv sa $3$, pošto kriptovanje uvek dodaje tačno $3$ karaktera (C, O, W).

**Optimizacija 2.** Za svaki karakter koji nije C, O ili W treba proveriti da li se pojavljuje isti broj puta kao i u tekstu “Begin the Escape execution at the Break of Dawn”. Isto treba proveriti i da li se karakteri C, O i W pojavljuju isti broj puta.

**Optimizacija 3.** Posmatrajno deo teksta u kome nisu C, O ili W. Takav isti tekst mora da postoji i u početnom tekstu. Npr. ukoliko imamo tekst "**C**n the B**W**" možemo da primetimo da se tekst "n the B" ne nalazi u početnom tekstu, te možemo da zaključimo da početan tekst ne može da bude "Begin the Escape execution at the Break of Dawn".

**Optimizacija 4.** Ukoliko posmatramo tekst od početka, prvo slovo za kriptovanje (C, O, W) mora da bude C. Isto tako poslednje kriptovano slovo mora da bude W.

**Optimizacije 5.** Posmatrajmo trenutni tekst od početka do prvog slova C, tj. početak teksta do prvo slova C mora da odgovara početku početnog teksta "Begin the Escape execution at the Break of Dawn". Isto tako, kraj teksta od poslednjeg karaktera W mora da odgovara kraju početnog nekriptovanog teksta.

U ovom zadatku je moguće primeniti još neke optimizacije, međutim potrebno je poznavanje listi o kojima će biti reči u nekim narednim lekcijama.

**Zadatak 3.** Na papiru je bilo napisano $n$ algebarski korektnih jednakosti. Međutim neko je promenio sve karaktere u tim jednakostima, te je svakoj cifri, operacijskom ili relacionom znaku dodelio neko slovo. Znamo da svaka dva različita slova odgovaraju različitim ciframa ili znakovima. U jednakosetima su se pojavljivali karakteri iz skupa $A = \{0,1,2,3,4,5,6,7,8,9,+,*,=\}$, gde se znak $'='$ pojavljivao tačno jednom u svakoj jednakosti. Sada se u jednakostima mogu pronaći slova iz skupa $B = \{a,b,c,d,e,f,g,h,i,j,k,l,m\}$, gde znamo da svako slovo iz skupa $B$ odgovara tačno jednom elementu iz skupa $A$. Vaš zadatak je da rekostruišete što više jednakosti, tj. bar nekih njihovih delova. Preciznije potrebno je da ispišete parove $(a,b)$ gde $a \in A$ i $b \in B$, što znači da je element $a$ sigurno zamenjen elementom $b$.

**Primer.**

| Ulaz | Izlaz |
|---|---|
| 2<br>abcdec<br>cdefe | a6<br>b*<br>d=<br>f+ |

**Objasnjenje.** Postoje 3 mogućnosti zamene slova sa ciframa/znakovima koje dovode do istinitih jednakosti i to su sledeće:

$$6 * 2 = 12 \text{ i } 2 = 1 + 1 \qquad 6 * 4 = 24 \text{ i } 4 = 2 + 2 \qquad 6 * 8 = 48 \text{ i } 8 = 4 + 4$$

Možemo zaključiti da umesto slova $a$ mora da stoji $6$, umesto $b$ znak $*$, $d$ je $=$, a $f$ je zamenio znak $+$.

**Rešenje.** Zadatak nam je da prođemo kroz sve moguće kombinacije koje zadovoljavaju jednakosti i da vidimo koji su elementi fiksni. Iz razloga što postoji previše kombinacija da dodelimo svakoj cifri/znaku po jedno slovo, ovde su nam potrebne optimizacije.

Znamo da samo jedno slovo može biti znak $'='$, te ćemo prvo njega da postavimo. Neko slovo može da bude znak jednakosti ukoliko se pojavljuje tačno jednom u svakoj jednakosti i nije ni prvo ni poslednje slovo jednakosti.

Posle znaka jednakosti, postavljamo znakove $'+'$, gde ponovo moramo da pazimo da to ne može da bude ni prvo ni poslednje slovo ni u jednoj jednakosti, niti može da stoji pored nekog drugog znaka $'+'$ ili $'='$. Ovde je potrebno paziti da ne moramo da imamo znak $'+'$ u jednakostima.

Sledeće dolaze znakovi $'*'$, za koje važe ista pravila kao za znak plus, sa dodatkom da znak za množenje ne može da stoji pored znaka za sabiranje. Isto kao i kod znakova za sabiranje, ne moramo da imamo nijedan znak $'*'$ u jednakostima.

Kad smo postavili sve znakove, ostaju cifre. Najvažnije optimizacije su ovde potrebne, pošto postoji mnogo kombinacija sa ciframa, jer postoji $10$ cifara za svako preostalo slovo.

Dato nam je $n$ jednakosti, gde ćemo posle svakog mapiranja slova u cifru proći kroz sve jednakosti i proveriti da li ima svrhe nastavljati pretraživanje, ili možemo da zaključimo da trenutno posmatrano uparivanje slova sa znakovima i ciframa nije moguće. Za svaku jednakost ćemo posmatrati levu i desnu stranu jednakosti u odnosu na znak $'='$.

Pokušaćemo da posle postavljanja svake cifre izračunamo što preciznije za svaku jednakost najmanji i najveći broj koji možemo da dobijemo sa leve i sa desne strane jednakosti. Nazovimo te brojeve $najmanjeLevo, najvećeLevo, najmanjeDesno\ i\ najvećeDesno$. Ukoliko važi jedna od nejednakosti $najmanjeLevo > najvećeDesno$ ili $najvećeLevo < najmanjeDesno$ možemo da zaključimo da trenutno posmatrano uparivanje nije moguće.

Sada ćemo da vidimo kako da izračunamo najmanji (najveći) broj sa jedne strane od znaka jednakosti. Pošto smo već postavili znakove za sabiranje i množenje, možemo da podelimo ceo izraz sa jedne strane znaka jednakosti u delove, gde će svaki deo da predstavlja jedan broj. Dok će između delova da stoje znaci za sabiranje i množenje.

Da bismo minimizovali (maksimizovali) ceo izraz, dovoljno je da minimizujemo (maksimizujemo) svaki deo tog izraza. Znamo da svaki deo predstavlja jedan broj, gde su već neke cifre već postavljene. Preostaje nam da uparimo neuparena slova sa ciframa na takav način da trenutno posmatrani broj bude što veći (manji). To možemo da uradimo ukoliko se pridržavamo pravila da slovo koje prvo dolazi u broju mora da bude manje (veće) od broja koje dolazi posle njega u broju. Pravilo se odnosi samo na slova koja nisu uparena sa nekom cifrom.

Najveći problem ovog zadatka je da se opisano rešenje iskuca, pošto i pored svih pomenutih optimizacija, mora da se pazi na efikasnost koda. Na primer moramo da čuvamo liste slova koje jos uvek nismo uparili, umesto nizova gde je izbacivanje sporije.