Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Nikola Dmitrović, Marin Kišić, Josip Klepec, Vedran Kurdija, Ivan Lazarić, Daniel Paleka, Ivan Paljak, Stjepan Požgaj i Paula Vidas. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima.

Zadatak: Dinamo

Predložio: Nikola Dmitrović
Potrebno znanje: naredba učitavanja i ispisivanja, jednostavna naredba odlučivanja

Na osnovi ulaznih podataka o klubovima s kojima je Dinamo igrao u prva tri kola, naredbom odlučivanja lako možemo provjeriti s kojim je klubovima igrao u četvrtom, petom i šestom kolu.

S kojim je klubovima igrao u tim kolima pronaći ćemo u tekstu zadatka.

Programski kod (pisan u `Python 3`):

```
A = int(input())
C = int(input())
S = int(input())
X = int(input())

if X == 4: print(S)
if X == 5: print(A)
if X == 6: print(C)
```

Zadatak: Lijepi

Predložio: Nikola Dmitrović
Potrebno znanje: učitavanje unaprijed poznatog broja brojeva zapisanih u istom retku, naredba ponavljanja, određivanje broja znamenki u broju, string

Pretpostavimo da ne znamo opće rješenje ovog zadatka. Pogledajmo u sekciji *Bodovanje* postoji li neka parcijala, neki dio zadatka koji bismo mogli riješiti.

U prvoj parcijali, vrijednoj 18 bodova, dobit ćemo samo jedan par dvoznamenkastih brojeva X i Y. Da bi od njih dobili novi broj, očito je da X trebamo pomnožiti sa 100 i tako dobivenoj vrijednosti pribrojiti Y. Uočite da nam za ovo treba samo naredba učitavanja dva broja koji su zapisani u istom retku i naredba odlučivanja. Nužno je učitati i broj N, iako je njegova vrijednost uvijek jedan i nigdje u rješenju zadatka ga ne koristimo. *Programski kod (pisan u `Python 3`):*

```
# prva parcijala
N = int(input())
X, Y = map(int, input().split()) # Python učitavanje dva broja u istom retku
zbroj = X * 100 + Y
print(zbroj)
```

Ako se odlučimo rješavati drugu parcijalu, vrijednu 22 boda, morat ćemo poznavati i naredbu ponavljanja. N puta ćemo dobiti po dva dvoznamenkasta broja X i Y te ćemo na svakom tom paru ponoviti prethodno opisani algoritam. Naravno, kako imamo N takvih parova bit će nužno kontinuirano zbrajati nove vrijednosti u posebnoj varijabli `zbroj`.

Programski kod (pisan u Python 3):

```
# druga parcijala
N = int(input())
zbroj = 0
for i in range(N):
  X, Y = map(int, input().split())
  novi = X * 100 + Y
  zbroj = zbroj + novi # može i ovako: zbroj += novi
print(zbroj)
```

Da bismo riješili zadatak za sve bodove, moramo za svaki par brojeva X i Y kreirati novi broj na način da prvo odredimo koliko znamenki ima broj Y, broj X pomnožimo s potencijom broja 10 na broj tih znamenki i tako dobivenoj vrijednosti pribrojimo vrijednost Y. Broj znamenki u nekom broju možemo odrediti uzastopnim cjelobrojnim dijeljenjem broja s 10 sve dok taj broj ne postane nula. Traženje broja znamenki moramo raditi na kopiji broja Y jer ćemo u protivnom izgubiti broj Y koji nam je nužan za nastavak algoritma.

Programski kod (pisan u Python 3):

```
# opće rješenje
N = int(input())
zbroj = 0
for i in range(N):
  X, Y = map(int, input().split())
  brznY = 0
  kopija = Y
  while kopija > 0:
    brznY += 1
    kopija //= 10
  zbroj += X * pow(10, brznY) + Y
print(zbroj)
```

Za kraj, riješimo zadatak koristeći stringove. Brojeve X i Y ćemo učitati kao "napisane brojeve", nalijepiti jedan na drugi koristeći operator "konkateniranja" i tako dobiveni string pretvoriti u prirodan broj naredbom `int()`.

Programski kod (pisan u Python 3):

```
# string rješenje
N = int(input())
zbroj = 0
for i in range(N):
  X, Y = input().split()
  zbroj += int(X + Y)
print(zbroj)
```

Zadatak: Trol

Predložio: Ivan Paljak
Potrebno znanje: dokazivanje slutnji, matematika, suma intervala u periodičkom nizu

Ovaj zadatak se na prvi pogled čini pomalo zastrašujuć jer je potrebno riješiti više upita, brojevi u upitima su vrlo veliki, promjene nad elementima niza su relativno kompleksne, itd. No, budući da se radi o trećem zadatku naziva kao što je "trol", može se naslutiti da se iza svega krije nešto elegantno.

No, krenimo redom. U test podacima vrijednima 10 bodova radimo nad elementima niza koje Marin uopće nije mijenjao (jer su već bili jednoznamenkasti). Za osvajanje ovih bodova bilo je dovoljno za svaki upit ispisati vrijednost $l + (l + 1) + \cdots + (r - 1) + r$.

Za dodatnih 20 bodova ograničenja su bila dovoljno mala da je bilo moguće simulirati postupak opisan u tekstu zadatka. Odnosno, krečući se petljom po brojevima od l prema r, bilo je potrebno za svaki od njih odsimulirati Marinove zamjene. Problem pronalaska zbroja znamenki nekog broja relativno je klasičan i dobro opisan u raznoj literaturi (**hint:** Kako dobiti zadnju znamenku nekog broja? Kako tu znamenku obrisati iz broja?). Ovo je rješenje implementirano u izvornom kodu `trol_brute.cpp`.

Za osvajanje svih bodova na ovom zadatku najprije je potrebno uočiti jednu pravilnost. Tu pravilnost možete uočiti na dva načina koja ćemo vam dočarati istinitim događajima koji su se odvili prilikom sastavljanja zadataka za ovo kolo.

Prvi način

Autor je ovaj zadatak najprije ispričao Marinu. Marin je iskusan natjecatelj i zna da prijedlog za treći zadatak na HONI-ju ne smije biti pretjerano težak pa je odmah naslutio da se iza kompliciranih operacija krije neka pravilnost. Unutar 3 minute Marin je napisao prethodno opisano rješenje (za 20 bodova) i pogledao što se događa s nizom. Brzo je zaključio da nakon zamjena niz A postaje periodičan. Preciznije, uočio je da vrijedi $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, \dots\}$. Marin nije odmah znao zašto vrijedi ova pravilnost, ali je eksperimentalno ~~dokazao~~ pokazao svoju slutnju.

Drugi način

Druga osoba kojoj je autor ispričao ovaj zadatak je Stjepan. Stjepan uistinu jest završio preddiplomski studij matematike pa se ubrzo sjetio da je u osnovnoj školi naučio kako je "broj djeljiv s 9 ako mu je zbroj znamenaka djeljiv s 9". Također je zaključio da vrijedi i jača tvrdnja, da je ostatak koji broj daje pri dijeljenju s 9 jednak ostatku koji zbroj njegovih znamenaka daje pri dijeljenju s 9. Budući da se ostatak nekog broja pri dijeljenju s 9 ne mijenja kada ga zamijenimo zbrojem njegovih znamenaka, a svaka znamenka od 1 do 9 ima jedinstven ostatak pri dijeljenju s 9, mora vrijediti da je taj broj u konačnici dovoljno zamijeniti onom znamenkom koja ima taj ostatak. Sada je jasno kako izgleda niz.

Bili vi Marin ili Stjepan, potpuno je svejedno, jer kada otkrijete ovu pravilnost, dalje je lagano. Svaki blok od uzastopnih 9 elemenata zadanog intervala ima istu sumu koja iznosi $1 + 2 + 3 \cdots + 9 = 45$. Takvih (potpunih) blokova ima $\lfloor \frac{r-l+1}{9} \rfloor$. Ostalo je još pribrojiti "višak", odnosno elemente zadnjeg, nepotpunog bloka. Tih elemenata ima $(r - l + 1)$ `mod` 9, a prvi je jednak l. Dakle, na svaki upit možemo odgovoriti u konstantnoj složenosti.

Zadatak: Lutrija

Predložio: Marin Kišić
Potrebno znanje: matematika, brza provjera je li broj prost, bfs/dfs

Da biste osvojili 14 bodova bilo je dovoljno najprije provjeriti je li $|A - B|$ prost broj, ako jest, onda je niz jednosavno $\{A, B\}$. Ako razlika nije prosta možemo fiksirati neki x između 2 i 1000 i provjeriti je li x prost, je li $|A - x|$ prost i je li $|x - B|$ prost. Ako je svo troje prosto onda je rješenje niz $\{A, x, B\}$, inače nema rješenja.

Rješenje sljedeće parcijale je ekvivalentno rješenju cijelog zadatka samo što se za provjeru je li neki broj prost može samo provjeriti svaki broj između 2 i tog broja, a u pravom rješenju to treba napraviti u $\mathcal{O}(\sqrt{n})$ operacija.

Konačno, za potpuno rješenje je bilo potrebno uočiti da je apsolutna razlika neka dva prosta broja veća od 2 uvijek paran broj. Kako se traži da apsolutna razlika između susjedna dva elementa niza bude prosta, možemo zaključiti da jedini brojevi koji se mogu pojaviti u nizu su $A - 2$, A, $A + 2$, $B - 2$, B, $B + 2$ i 2. Od tih sedam brojeva ostavimo samo proste. Sada, svaki broj možemo zamisliti kao čvor u grafu u kojem veza između dva čvora postoji ako je razlika pripadajućih brojeva prosta. Na kraju, bilo kojim algoritmom za traženje puta na grafu pronađemo neki put od A do B.

Zadatak: Džumbus

Predložio: Vedran Kurdija
Potrebno znanje: stablo, dinamičko programiranje, analiza složenosti

Kao što to obično i biva, sve ćemo podzadatke riješiti dinamičkim programiranjem.

U prvom i drugom podzadatku iz ograničenja zaključujemo da parovi čine točno jedan lanac. Prvi ćemo podzadatak riješiti koristeći dinamiku koja u stanju ima poziciju u lancu P, količinu džumbusa koju imamo na raspolaganju K te bit B sa vrijednošću 0 ili 1 koji označava je li osoba na poziciji P u lancu razmjenjivala bilješke. Vrijednost dinamike $dp[P][K][B]$ odnosit će se samo na prvih P ljudi u lancu, a bit će jednaka najvećem broju ljudi koji mogu barem jednom razmjenjivati bilješke uz optimalnu raspodjelu K decilitara džumbusa te uz uvjet da je osoba na poziciji P u lancu razmjenjivala bilješke (slučaj $B = 1$) ili nije (slučaj $B = 0$). Neka $L[i]$ označava oznaku i-te osobe u lancu. Prijelaz je (uz obraćanje pozornosti na granice i rubne slučajeve koje prepuštamo čitatelju) sljedeći:

$$dp[P][K][0] = \max\{dp[P-1][K][0], dp[P-1][K][1]\}$$

$$dp[P][K][1] = \max \begin{cases} dp[P-1][K - D_{L[P]}][1] + 1 \\ dp[P-1][K - D_{L[P]} - D_{L[P-1]}] + 2 \end{cases}$$

Odgovor na upit S_i pronalazimo kao $\max\{dp[L[N]][S_i][0], dp[L[N]][S_i][1]\}$. Složenost je $\mathcal{O}(N \cdot \max S_i)$.

Budući da u drugom podzadatku nema dodatnog ograničenja na vrijednosti S_i, dinamika iz prvog zadatka bila bi daleko prespora i zauzela daleko previše memorije. Zato ćemo stvari lukavo obrnuti. U stanju će ponovno biti pozicija u lancu i bit koji označava je li trenutna osoba razmjenjivala bilješke, no sada ćemo količinu džumbusa iz stanja premjestiti u vrijednost dinamike, a broj ljudi koji su barem jednom razmjenjivali bilješke premjestiti iz vrijednosti u stanje. Prijelaz koristi sličnu ideju kao i u prošlom podzadatku te ga ostavljamo čitatelju za samousavršavanje. Odgovor na upit S_i pronalazimo kao najveći R za koji je $\min\{dp[L[N]][R][0], dp[N][R][1]\} \leq S_i$. Detalje oko implementacije odgovaranja na upite ostavljamo čitatelju, budući da nisu zahtjevni. Složenost izračuna dinamike je $\mathcal{O}(N^2)$.

U trećem podzadatku više nemamo lanac, već šumu stabala. Šumu možemo svesti na jedno stablo tako da stvorimo novi čvor kojega ćemo tretirati kao osobu kojoj je potrebna beskonačna količina džumbusa da se oraspoloži (u praksi za ovo naprosto uzimamo neki broj veći od $\max S_i$, tj. od 10^9). Na novi ćemo čvor spojiti sva stabla iz šume. Time smo dobili jedno veliko stablo, sigurni da nam novi čvor neće pokvariti računanje rezultata. Koristimo sličnu dinamiku kao u drugom podzadatku, samo ovoga

puta na stablu. Stablo ćemo ukorijeniti u bilo kojem čvoru (npr. čvoru 1). Vrijednost $dp[P][R][B]$ je najmanja količina džumbusa potrebna da u podstablu čvora P postignemo da R ljudi razmijeni bilješke, uz uvijet da je osoba P razmjenjivala bilješke ako je $B = 1$, ili nije ako je $B = 0$. Prije izračuna vrijednosti dinamike za čvor P, trebamo imati izračunate sve vrijednosti dinamika njegove djece. Koristit ćemo i pomoćnu dinamiku $dp\_prefiks$, koja u stanju ima broj ljudi unutar podstabla čvora P koji su barem jednom razmjenjivali bilješke R, bit za vrijednošću 0 ili 1 koji, kao i prije, označava je li osoba P razmjenjivala bilješke, te broj K koji označava da smo do sada uzeli u obzir prvih K djece čvora P. Očito je onda $dp[P][R][B] = dp\_prefiks[R][B]$[broj djece od P]. Neka $C[i]$ označava i-to dijete čvora P. Za ovu dinamiku imamo prijelaze:

$$dp\_prefiks[R][0][K] = \min_{A+B=R}\{dp\_prefix[A][0][K-1] + \min\{dp[C[K]][B][0], dp[C[K]][B][1]\}\}$$

$$dp\_prefiks[R][1][K] = \min_{A+B=R}\begin{cases} dp\_prefiks[A-1][0][K-1] + D_P + dp[C[K]][B][1] \\ dp\_prefiks[A-1][0][K-1] + D_P + dp[C[K]][B-1][0] + D_{C[K]} \\ dp\_prefiks[A][1][K-1] + D_P + dp[C[K]][B][1] \\ dp\_prefiks[A][1][K-1] + D_P + dp[C[K]][B-1][0] + D_{C[K]} \\ dp\_prefiks[A][1][K-1] + dp[C[K]][B][0] \end{cases}$$

(Implementacijske detalje i rubne slučajeve ostavljamo čitatelju.) Kada pozbrojimo po svim čvorovima, ukupno postoji $\mathcal{O}(N^2)$ stanja dinamike $dp\_prefiks$, i svaki prijelaz se računa u najviše $\mathcal{O}(N)$ koraka, pa složenost možemo ograničiti sa $\mathcal{O}(N^3)$. Znam što mislite. Nema teorije da se ovo može još ubrzati, zar ne!?

Za rješavanje četvrtog podzadatka potrebno je uočiti da je, uz pažljivu implementaciju, složenost opisanog rješenja zapravo $\mathcal{O}(N^2)$. Očito je nemoguće da u nekom skupu ljudi više nijh razmijeni bilježnice nego što je veličina tog skupa. Stoga, pri računanju $dp\_prefiks$, nije potrebno za A i B razmatrati sve moguće vrijednost, već samo one za koje je ta situacija moguća (dakle A ne može biti veći od zbroja veličina podstabala do sada obrađene djece plus 1, a B ne može biti veći od veličine podstabla trenutnog djeteta). Dokazat ćemo da je za svaki čvor ukupna složenost izračuna dinamike za njega i sve čvorove u njegovom podstablu $\mathcal{O}$(veličina podstabla^2). Dokaz provodimo induktivno po dubini stabla. Tvrdnja očito vrijedi ako je čvor list. Pretpostavimo da smo za neki čvor izračunali dinamiku za njegovu djecu u željenoj složenosti. Neka taj čvor ima X djece, čije su veličine podstabala $Y_1, Y_2, ...Y_X$. Ukupna složenost izračuna dinamike je

$$\begin{aligned}
\mathcal{O}\left(\sum_{i=1}^{X} Y_i^2 + \sum_{i=1}^{X}(1 + \sum_{j=1}^{i-1} Y_j)\cdot Y_i\right) &= \mathcal{O}\left(\sum_{i=1}^{X} Y_i^2 + \sum_{i=1}^{X} Y_i + \sum_{i=1}^{X}\sum_{j=1}^{i-1} Y_i Y_j\right) \\
&= \mathcal{O}\left(1 + \sum_{i=1}^{X} Y_i^2 + 2\sum_{i=1}^{X} Y_i + 2\sum_{i=1}^{X}\sum_{j=1}^{i-1} Y_i Y_j\right) \\
&= \mathcal{O}\left((1 + \sum_{i=1}^{X} Y_i)^2\right) \\
&= \mathcal{O}(\text{veličina podstabla}^2)
\end{aligned}$$

Stoga je ukupna složenost cijelog rješenja $\mathcal{O}(N^2)$.

Zadatak: Trobojnica

Predložio: Daniel Paleka
Potrebno znanje: konstruktivni algoritmi, matematička indukcija, amortizirana složenost

Kako bismo osvojili podzadatak od 10% bodova, trebamo pronaći nužan i dovoljan uvjet postojanja domoljubne triangulacije za mnogokut s obojanih stranica.

Tvrdnja 1: U svakoj triangulaciji za $N \geq 4$ postoje barem dva "uha", tj. trokut koji dijeli dvije stranice s mnogokutom.

Skica dokaza: Indukcija. Vrijedi za kvadrat. Neka dijagonala dijeli mnogokut na dva manja mnogokuta. Ako je neki od njih trokut, to je traženo uho; inače se možemo induktivno pozvati dalje.

Neka na početku ima a, b i c stranica u bojama 1, 2 i 3 redom. Vrijedi $a + b + c = N$. Jasno je da mora biti $\max\{a, b, c\} < n$, inače ne može postojati uho.

Tvrdnja 2: Ako postoji domoljubna triangulacija, brojevi $\{a, b, c\}$ su iste parnosti.

Skica dokaza: Dvostruko prebrojavanje. Prebrojimo li broj bridova boje 1 po trokutima, dobivamo:

$$2 \cdot (\text{broj dijagonala boje 1}) + a = N - 3,$$

iz čega zaključujemo da je a iste parnosti kao $N - 3$. Analogno dobivamo isto i za b i c.

Navedeni uvjeti su zapravo dovoljni; to se najbolje dokazuje algoritmom koji je rješenje ovog zadatka. Ukratko rečeno, induktivno ćemo raditi uši koristeći stranice dvije različite boje, čime nećemo promijeniti (dokažite!) jednakost parnosti brojeva $\{a, b, c\}$ u novom mnogokutu, koji ima jednu stranicu manje. Jedino na što treba paziti je mogućnost da novi mnogokut ima sve stranice iste boje.

Jedna lijepa implementacija je sljedeća: za svaki vrh mnogokuta pamtimo njegovog sljedbenika u trenutnom mnogokutu, te boju brida koji ih povezuje. Traženje mjesta na kojem možemo napraviti uho zapravo nije veliki problem. Naime, možemo samo ići u krug i napraviti uho na prvom vrhu čiji su prethodni i sljedbeni brid iste boje, te je barem jedna od tih boja ona boja koje ima najviše među trenutnim stranicama. Ukoliko svaki put krenemo od mjesta gdje smo napravili zadnje uho, moguće je pokazati da je vremenska složenost amortizirani $\mathcal{O}(N)$.

Postoje i kompliciranija $\mathcal{O}(N \log N)$ rješenja, koja također trebaju dobiti sve bodove.

Zadatak: Zoo

Predložio: Ivan Paljak
Potrebno znanje: bfs/dfs, određivanje dubine stabla

Primijetimo da nam je odmah poznato koja je životnja zadnja prošla preko pravokutnog područja. Ako se na početnom i završnom polju nalazi znak `T`, tada je to bio tigar, a inače bik. Također, primijetimo da, ako je poljem prošao minimalan broj životinja, tada nikad nisu dvije životinje iste vrste išle jedna za drugom.

Zadatak rješavamo "unazad". Pronađimo najprije sva polja na koja je mogla stati zadnja životinja prije nego što je napustila područje. Ta polja čine povezanu komponentu tragova iste vrste (u četiri smjera), a možemo je pronaći pretragom u širinu. Budući da je tim svim poljima mogla proći i prethodna životinja, možemo u sva polja te komponente upisati tragove životnje druge vrste i dodati 1 u rješenje. Ovaj postupak je potrebno ponavljati sve dok na polju ne ostanu tragovi samo jedne vrste životinja. Naivna implementacija ovog algoritma osvojit će 45 bodova.

Zamislimo svaku povezanu komponentu tragova iste vrste kao čvor u grafu. Dva su čvora povezana ako su odgovarajuće komponente susjedne, odnosno postoje dva susjedna polja u matrici od kojih jedno pripada prvoj, a drugo drugoj komponenti. Primijetimo da je taj graf zapravo stablo. Ukorijenimo li stablo u komponenti kojoj pripada gornje lijevo polje, tada broj koraka prethodno opisanog algoritma odgovara dubini tog stabla. Stablo možemo izgraditi u vremenskoj složenosti $\mathcal{O}(R \cdot S)$, a broj čvorova u stablu ograđen je istim izrazom.