



Opisi algoritama

Zadatke, testne primjere i rješenja pripremili: Ivan Janjić, Sofija Velkovska i Jakov Celin.

Primjeri implementiranih rješenja dani su u priloženim izvornim kodovima.

Zadatak Upisi

Pripremila: Sofija Velkovska

Potrebno znanje: naredbe učitavanja i ispisivanja

Prema tajnoj formuli, broj bodova koje će Jakov osvojiti na rang-listi jednak je umnošku rezultata koje je postigao na maturi iz matematike i maturi iz fizike. Rezultati mature iz matematike i fizike su redom drugi i treći broj u ulazu. Stoga trebamo učitati sva četiri broja s ulaza i ispisati umnožak drugog i trećeg broja.

Programski kod (pisan u Python 3):

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
```

```
print(b * c)
```

Zadatak Put II

Pripremila: Sofija Velkovska

Potrebno znanje: naredba odlučivanja (if)

Ako je sat dolaska malih medvjedića manji od sata polaska autobusa (možemo zanemariti minute) tj. $a < x$, medvjedići su sigurno stigli na polazak autobusa. Ako su sati jednaki tj. $a = x$, medvjedići su stigli na polazak autobusa samo ako su minute njihovog dolaska manje ili jednake minuti polaska autobusa tj. $b < y$. Inače, nisu stigli na autobus. Ako je sat dolaska malih medvjedića na autobusni kolodvor veći od sata polaska autobusa tj. $a > x$, ne moramo ni gledati minute - medvjedići sigurno nisu stigli na autobus na vrijeme.

Programski kod (pisan u Python 3):

```
x, y = map(int, input().split())
a, b = map(int, input().split())

if a < x:
    print("DA")
elif a == x and b <= y:
    print("DA")
else:
    print("NE")
```

Zadatak Zadatak

Pripremila: Sofija Velkovska

Potrebno znanje: naredba odlučivanja (if), naredba ponavljanja (for), rastavljanje broja na znamenke

Izračunajmo broj koji se dobije nakon primjene prve operacije k puta i broj koji se dobije nakon primjene druge operacije k puta. Odgovor na profesorovo pitanje je veći od ta dva broja - ispišemo ga.



Neka je y broj koji smo dobili nakon primjene prve operacije nekoliko puta. Da bismo još jednom primijenili prvu operaciju trebamo pomnožiti broj y samim sobom, izračunati zbroj znamenaka dobivenog broja i uzeti taj zbroj za novi y . Jedan način računanja zbroja znamenaka prirodnog broja y : uzmimo zadnju znamenku broja pomoću operacije ostatka pri dijeljenju sa 10 ($y\%10$), uklonimo zadnju znamenku cjelobrojnim dijeljenjem sa 10 ($y//10$), ponavljajmo postupak dok broj ne postane 0.

Neka je y broj koji smo dobili nakon primjene druge operacije nekoliko puta. Da bismo još jednom primijenili drugu operaciju trebamo pomnožiti broj y samim sobom i uzeti posljednje dvije znamenke tog broja za novi y . Broj dobiven od posljednje dvije znamenke broja y jednak je ostatku broja y pri dijeljenju sa 100 tj. $y\%100$.

Programski kod (pisan u Python 3):

```
x, k = map(int, input().split())

y1 = x
for i in range(k):
    y1 = y1 * y1
    zbroj = 0
    while y1 > 0:
        zbroj = zbroj + y1 \% 10
        y1 = y1 // 10
    y1 = zbroj

y2 = x
for i in range(k):
    y2 = y2 * y2
    y2 = y2 \% 100

if y1 > y2:
    print(y1)
else:
    print(y2)
```

Zadatak Škare

Pripremila: Sofija Velkovska

Potrebno znanje: naredba ponavljanja (for), naredba odlučivanja (if), polja

Jedan način rješavanja ovog zadatka jest praćenje položaja rezova u odnosu na početnu traku, umjesto spremanja svih trenutačnih traka zasebno. Neka polje *rez* predstavlja početnu traku. Neka je i -ti element polja *rez* jednak 1 ako je nakon i -tog centimetra početne trake napravljen rez, a 0 inače. Tada je svaka traka dobivena rezanjem početne trake predstavljena segmentom polja, pri čemu jedinice određuju gdje traka počinje i završava (a u slučaju prve i posljednje trake početak, odnosno kraj, određen je početkom, odnosno krajem polja), dok se između njih nalaze nule.

Pretpostavimo da želimo prerezati x -tu traku nakon njezina l -tog centimetra. Ako želimo prerezati prvu traku, promatramo segment od prvog elementa polja *rez* do prve jedinice. Ako želimo prerezati posljednju traku, promatramo segment od posljednje jedinice do posljednjeg elementa polja *rez*. Inače se x -ta traka nalazi između $(x - 1)$ -te i x -te jedinice u polju. Označimo položaj $(x - 1)$ -te jedinice u polju s i . l -ti centimetar x -te trake odgovara $(i + l)$ -tom centimetru početne trake. Na tom mjestu treba napraviti rez - toj poziciji u polju *rez* pridružujemo vrijednost 1.

Nakon što obavimo sve rezove, duljinu svake trake računamo kao razliku između položaja x -te i $(x - 1)$ -te jedinice u polju *rez*, uz odgovarajuću prilagodbu za posebne slučajeve prve i posljednje trake. Broj



različitih duljina traka možemo pratiti pomoću drugog 0/1 polja *duljine*. U početku postavimo vrijednost svakog elementa polja *duljine* na 0. Kada naiđemo na traku duljine l , postavimo vrijednost *duljine*[l] na 1. Na kraju je broj različitih duljina traka jednak broju jedinica u polju *duljine*.

Zadatak Težina

Pripremio: Jakov Celin

Potrebno znanje: eratostenovo sito, analiza složenosti

Zadatak se mogao riješiti na razne načine, no opisati ćemo dva najlakša.

Za svaku vrstu utega i gdje vrijedi $1 \leq i \leq k$ potrebno je izračunati tzv. snagu utega. Za svaki predmet mase a_x računa se vrijednost $\lfloor a_x/i \rfloor \cdot (a_x + 2)$, pri čemu se rezultat, ako je veći od 10^8 , zamjenjuje s 10^8 . Snaga utega i jednaka je zbroju tih vrijednosti za sve predmete, a traženi odgovor je zbroj snaga svih utega.

Direktno računanje za svaki par i i x bilo bi presporo jer bi zahtijevalo $O(nk)$ operacija.

Jedno od rješenja bilo je prolaziti redom po utezima te grupirati vrijednosti za koje količnik $\lfloor a_x/i \rfloor$ poprima istu vrijednost.

Za fiksni i vrijedi da je $\lfloor a_x/i \rfloor = j$ za sve vrijednosti a_x koje zadovoljavaju $i \cdot j \leq a_x < i \cdot (j + 1)$. Dakle, za svaku vrijednost j možemo promatrati interval $[i \cdot j, i \cdot (j + 1) > i$ obraditi sve elemente niza čije mase pripadaju tom intervalu odjednom.

Neka je C broj elemenata niza u tom intervalu, a S njihov zbroj. Doprinos svih tih elemenata ukupnom rezultatu jednak je $j \cdot (a_x + 2)$ za svaki element, pa njihov ukupni doprinos iznosi $j \cdot (S + 2C)$.

Potrebno je još uzeti u obzir ograničenje od 10^8 . Budući da vrijedi $a_x \leq 10^5$, za dovoljno velike vrijednosti i količnik $j = \lfloor a_x/i \rfloor$ postaje vrlo malen, pa tada nije moguće da izraz $j \cdot (a_x + 2)$ prijeđe granicu 10^8 . Zato je dovoljno za prvih približno 101 vrijednosti i eksplicitno provjeravati ograničenje i po potrebi ograničiti vrijednost na 10^8 . Za veće vrijednosti i ta provjera više nije potrebna.

Za fiksni i broj različitih vrijednosti količnika j približno je a/i , gdje je a maksimalna vrijednost mase (do 10^5). Ukupan broj iteracija preko svih utega tada iznosi $a + a/2 + a/3 + \dots + a/k$, što je $O(a \log k)$.

Ukupna vremenska složenost algoritma je $O(B \cdot n + a \log k)$, gdje je B broj prvih utega za koje još može doći do ograničenja na 10^8 (otprilike 101). Ova složenost je dovoljna za zadana ograničenja. Alternativno (i službeno) rješenje koristilo je tvrdnju da za fiksnu vrijednost x , $\lfloor a_x/i \rfloor$ može poprimiti $O(\sqrt{x})$ različitih vrijednosti te zatim direktno računalo doprinosi jednog elementa niza za $O(\sqrt{x})$ intervala utega.

Zadatak Pet

Pripremio: Jakov Celin

Potrebno znanje: bitset, dinamičko programiranje, ad-hoc

Polja matrice s vrijednosti 1 promatramo kao vrhove grafa. Iz polja (x, y) Maša može skočiti u bilo koje drugo polje u istom retku ili stupcu, pri čemu se tip skoka mora izmjenjivati (redak, stupac, redak, ...). Trebamo pronaći broj puteva koji posjećuju točno 5 različitih polja.

Najprije, izračunajmo broj svih putova duljine 5 bez uvjeta da su sva polja različita. Definirajmo:

$f(x, y, i)$ - broj načina da nakon i skokova završimo u (x, y) ako je zadnji skok promijenio redak

$g(x, y, i)$ - broj načina da nakon i skokova završimo u (x, y) ako je zadnji skok promijenio stupac.

Primjetimo da su $f(x, y, k)$ i $g(x, y, k)$ jednaki 0 ako je polje (x, y) u matrici jednako 0.

Za preostala polja, prijelazi u dinamikama su:



$$f(x, y, i) = \sum_{z=1}^m g(x, z, i-1) - g(x, y, i-1)$$

$$g(x, y, i) = \sum_{z=1}^n f(z, y, i-1) - f(x, y, i-1).$$

Sume po retcima i stupcima možemo održavati tijekom računanja pa ukupan broj putova duljine 5 možemo izračunati u $O(nm)$.

U ovom brojanju dopušteno je ponavljanje polja. U putu duljine 5 to je jedino moguće samo ako put obilazi pravokutnik $(r_1, c_1) \rightarrow (r_1, c_2) \rightarrow (r_2, c_2) \rightarrow (r_2, c_1) \rightarrow (r_1, c_1)$.

Dakle, potrebno je oduzeti sve takve putove. Svaki pravokutnik daje 8 različitih putova (biramo četiri početna vrha i dva smjera kretanja).

Broj pravokutnika računamo ovako: za svaki par redaka izračunamo broj stupaca x u kojima oba retka imaju vrijednost 1. Tada oni doprinose s $x(x-1)/2$ pravokutnika. Presjek jedinica dvaju redaka možemo brzo dobiti korištenjem bitsetova, u vremenu $O(\frac{m}{64})$.

Konačan rezultat dobivamo kao (broj svih putova duljine 5) $- 8 \cdot$ (broj pravokutnika). Ukupna vremenska složenost algoritma jednaka je $O(\frac{n^2m}{64})$.

Zadatak Slaganje

Pripremio: Ivan Janjić

Potrebno znanje: teorija grafova, ad hoc

Jedan mogući smjer koji dovodi do rješenja je pažljivo odabrati jedno ulaganje stabla u mnogokut i rotirati ga N puta za jedan vrh u lijevo. Kada smo se na ovaj način ograničili, pitanje postaje kada rotacije stabla uspiju pokriti sve strane i dijagonale.

Definiramo *duljinu* strane ili dijagonale između vrhova x i y (bez smanjenja općenitosti neka je $x < y$) kao $d(x, y) := \min\{y - x, N - y + x\}$. Primjetimo da vrijedi $1 \leq d(x, y) \leq \lfloor \frac{N}{2} \rfloor$ te da se svaka vrijednost u tom intervalu postiže za neke x i y . Također primjetimo da rotacijama nekog brida pokrijemo sve dijagonale iste duljine.

To nas motivira da u traženom ulaganju stabla za svaku moguću duljinu postoji brid te duljine. Ako postoji vrh koji je zajednički svim bridovima, to jest ako je stablo zvijezda, onda će bilo kakvo ulaganje stabla pokriti sve duljine. To nas motivira da napravimo dekompoziciju stabla u male zvijezde u nadi da će broj bridova koje te zvijezde pokriju biti veći ili jednak $\lfloor \frac{N}{2} \rfloor$.

Postoji više načina za konstrukciju takve dekompozicije, ovdje predstavljamo možda najjednostavniju. Ukorijenimo stablo u proizvoljnom vrhu te promotrimo proizvoljan vrh na najvećoj dubini, neka mu je oznaka x . Neka je $p(x)$ oznaka njegovog roditelja. U dekompoziciju dodamo zvijezdu čiji je centar vrh $p(x)$, a ostali vrhovi zvijezde su njegova djeca (uključujući vrh x). Maknemo vrh $p(x)$ i njegovu djecu iz stabla te ponavljamo postupak dokle možemo.

Uzmimo neki brid između y i $p(y)$ koji nije dio niti jedne zvijezde. Prema konstrukciji to znači da je y centar neke zvijezde, to jest prema prethodnoj notaciji postojao je x takav da $p(x) = y$. Za brid stabla koji nije dio niti jedne zvijezde (između y i $p(y)$) pronašli smo brid koji je dio neke zvijezde (između x i $p(x) = y$). Očito je ovakva identifikacija između bridova injekcija pa zaključujemo da bridova u zvijezdama ima barem koliko i bridova van zvijezda, odnosno ima ih barem $\lceil \frac{N-1}{2} \rceil = \lfloor \frac{N}{2} \rfloor$.

Preostaje posložiti dobivene zvijezde tako da pokriju sve duljine. Maknimo neke zvijezde i smanjimo druge tako da unija zvijezda ima točno $\lfloor \frac{N}{2} \rfloor$ bridova. Imamo četiri pokazivača a, b, c, d na vrhovima. Ako je N neparan, u početku $a = c = 1, b = \lfloor \frac{N}{2} \rfloor + 1, d = \lfloor \frac{N}{2} \rfloor + 2$. Ako je N paran, u početku $a = c = 1, b = d = \lfloor \frac{N}{2} \rfloor + 1$. Ako prva zvijezda ima k (ako N paran pobrinemo se da $k \geq 2$) bridova, slažemo centar zvijezde na vrh a , a ostalih k vrhova na vrhove $b, b-1, \dots, b-k+1$. Promijenimo vrijednosti na sljedeći način: $a := a, b := b-k, c := c-1, d := d+k-1$. Naravno, vrh 0 je isto što i vrh N , vrh $N+1$ je isto što i vrh 1 itd. Ako druga zvijezda ima l bridova, slažemo centar zvijezde na



vrh c , a ostalih l vrhova na vrhove $d, d + 1, \dots, d + k - 1$. Promijenimo vrijednosti na sljedeći način: $a := a + 1, b := b - k + 1, c := c, d := d + k$. Za treću zvijezdu ponavljamo postupak za prvu zvijezdu, za četvrtu zvijezdu ponavljamo postupak za drugu itd. Nije se teško uvjeriti da prva zvijezda pokrije duljine $\lfloor \frac{N}{2} \rfloor, \lfloor \frac{N}{2} \rfloor - 1, \dots, \lfloor \frac{N}{2} \rfloor - k + 1$, druga $\lfloor \frac{N}{2} \rfloor - k, \lfloor \frac{N}{2} \rfloor - k - 1, \dots, \lfloor \frac{N}{2} \rfloor - k - l + 1$ itd.

Za implementacijske detalje pogledajte službeno rješenje.

Zadatak Struktura

Pripremio: Jakov Celin

Potrebno znanje: modularna aritmetika, brzo potenciranje, potenciranje matrica, kombinatorika

Petar bira niz duljine n tako da je svaki element odabran potpuno nasumično iz skupa $1, 2, \dots, k$. Ukupan broj mogućih nizova jednak je k^n koji je potrebno izračunati tehnikom brzog potenciranja. Tražena vjerojatnost jednaka je omjeru broja nizova koji zadovoljavaju uvjete strukture i ukupnog broja mogućih nizova.

Ako je $k < n$, tada niz ne može biti struktura jer se svaki broj od 1 do n mora pojaviti točno jednom. U tom slučaju odgovor je 0.

Pretpostavimo stoga da je $k \geq n$. Budući da se svaki broj od 1 do n pojavljuje točno jednom, niz a je zapravo permutacija tih brojeva. Dodatni uvjet strukture je $|a_i + i - n - 1| \leq 1$ za svaki i .

Neka je $f(n)$ broj permutacija koje zadovoljavaju uvjet strukture. Promotrimo položaj najvećeg broja n . On može biti samo na prvoj ili drugoj poziciji (slijedi direktno iz promatranja uvjeta za svaki i).

Ako je $a_1 = n$, tada preostalih $n - 1$ elemenata mora tvoriti strukturu iste vrste na brojevima $1, \dots, n - 1$. U tom slučaju imamo $f(n - 1)$ mogućnosti.

Ako je $a_2 = n$, tada mora vrijediti $a_1 = n - 1$, a preostalih $n - 2$ elemenata opet čine strukturu na brojevima $1, \dots, n - 2$. U tom slučaju imamo $f(n - 2)$ mogućnosti.

Slijedi da vrijedi $f(n) = f(n - 1) + f(n - 2)$. Početni uvjeti su $f(1) = 1$ i $f(2) = 2$, pa je $f(n)$ upravo n -ti Fibonaccijev broj.

Vrijednost $f(n)$ možemo izračunati u vremenu $O(\log n)$ koristeći potenciranje matrica za standardnu Fibonacci rekurziju. Ukupan broj povoljnih nizova jednak je $f(n)$, a ukupan broj svih nizova je k^n . Tražena vjerojatnost je $f(n)/k^n$. Vremenska složenost algoritma je $O(\log n)$.