Opisi algoritama

IBT 2011. MASTERS LIGA - 1. kolo

Komentare i pitanja uputite na askurdija@gmail.com *ili na* frane.kurtovic@gmail.com.

Niz (P1-1)

Izračunajmo razliku drugog i prvog člana, te razliku trećeg i drugog člana. Ako su razlike različite, niz nije aritmetički pa ispisujemo "nemoguce". Inače, dobivenu razliku dodajemo trećem članu kako bismo dobili četvrti član, pa dodavanjem razlike četvrtom članu dobivamo peti član i tako dalje do N-tog člana.

Ugljikovodik (P1-2, P2-1)

Iz drugog retka ulaznih podataka izračunat ćemo koliko ugljikovih, a koliko vodikovih atoma ima traženi ugljikovodik.

Odabrat ćemo broj uzetih ugljikovih atoma na sve moguće načine. Iz tog podatka računamo broj uzetih vodikovih atoma, a potom računamo koliko će molekula ugljikovodika nastati. Potonje nalazimo npr. *while* petljom: gradimo molekule dok god još imamo atoma na raspolaganju.

Kad smo dakle - za svaki mogući odabir koliko ćemo uzeti ugljikovih, a koliko vodikovih atoma - izračunali koliko nastaje molekula ugljikovodika, ispisujemo nađeni broj molekula koji je za sve odabire najmanji.

Domine (P1-3, P2-2)

Za svaku dominu, jedino što je bitno jest početno i završno slovo te njezina duljina.

Ako ne postoji ni jedna domina koja počinje i završava međusobno različitim slovima, uzet ćemo sve domine koje počinju i završavaju sa B, ili sve domine koje počinju i završavaju sa C - što nam se od toga više isplati.

Ako pak postoji domina koja počinje i završava međusobno različitim slovima, sigurno nam se isplati staviti barem jednu takvu u lanac (jer inače gradimo lanac samo od domina koje počinju i završavaju istim slovima, a ovu možemo barem dodati na početak ili kraj takvog lanca). S druge strane, uzmemo li u lanac takvu dominu (npr. neka ona počinje sa B i završava sa C), na početak te domine možemo dodati sve one koje počinju i završavaju sa B (jer one ni na što ne utječu osim na duljinu lanca), a na kraj te domine možemo dodati sve one koje počinju i završavaju sa C. Dakle, sve domine koje počinju i završavaju međusobno jednakim slovima možemo uračunati u rješenje i zanemariti.

Ostaje nam situacija u kojoj imamo samo domine koje počinju i završavaju međusobno različitim slovima. Tada promatramo dva slučaja: lanac počinje slovom B, ili lanac počinje slovom C. U svakom od tih slučajeva gradimo lanac na pohlepan način: u trenutku kad biramo koju sljedeću dominu staviti, ionako sve one koje dolaze u obzir završavaju istim slovom, pa nam se isplati uzeti najveću moguću, i tako dalje gradimo dok god možemo. Konačno, kao rješenje uzimamo povoljniji od ta dva slučaja (koji daje veću duljinu lanca).

Otoci (P1-4)

Za svaki otok x izračunat ćemo dvije vrijednosti: $f(x)$, najmanji broj vožnji od otoka x do otoka 1, te $g(x)$, najmanji broj vožnji od otoka x do otoka 2. Primijetimo da su tada, budući da su vožnje dvosmjerne, $f(x)$ i $g(x)$ ujedno i najkraći putevi od otoka 1 i 2 do otoka x. Zato ćemo vrijednosti $f(x)$ i $g(x)$ izračunati BFS širenjem od otoka 1 i 2.

Tada je nakraće putovanje od x do y, koje koristi trajekt između otoka 1 i 2, duljine $f(x) + 1 + g(y)$ (ako idemo od otoka x do otoka 1, pa na trajekt, pa od otoka 2 do otoka y), ili duljine $g(x) + 1 + f(y)$ (ako idemo od otoka x do otoka 2, pa na trajekt, pa od otoka 1 do otoka y). Od te dvije mogućnosti uzmimamo, naravno, povoljniju. (U te su mogućnosti uključena i ona putovanja koja dvaput koriste trajekt.)

Prolaskom po svim parovima otoka x, y izračunat ćemo duljinu najkraćeg putovanja za njih (opisano u prethodnom odlomku) i zbrajajući te duljine dobiti rješenje.

Vjerojatnost (P2-3)

Zadatak rješavamo dinamičkim programiranjem. Stanje je, uz fiksan broj B (kraj intervala), opisano brojem A (početkom intervala), brojem N (koliko još brojeva odabiremo) i brojem M (ostatak pri dijeljenju s K koji trebamo postići). Vrijednost za takvo stanje jest vjerojatnost da, u intervalu $[A, B]$, postignemo zbroj koji daje ostatak M pri dijeljenju s K odabirom N brojeva.

Vjerojatnost za pojedino stanje, $f(A, N, M)$, računamo na sljedeći način. Imamo dva slučaja: ili ćemo odabrati broj A na kojem se nalazimo - pa prelazimo na stanje $(A + 1, N - 1, M - A)$, ili ga nećemo odabrati - pa prelazimo na stanje $(A + 1, N, M)$. Ako vjerojatnost da ćemo uopće odabrati broj A iznosi p, onda dakle imamo relaciju $f(A, N, M) = p \cdot f(A + 1, N - 1, M - A) + (1 - p) \cdot f(A + 1, N, M)$.

Koliki je p? U intervalu postoji $B - A + 1$ brojeva, a za svakog od njih jednaka je vjerojatnost da će biti odabran. Vjerojatnost da će nasumičan odabir N brojeva iz tog intervala sadržavati određeni broj (npr. broj A) iznosi $p = N/(B - A + 1)$. To je intuitivno prihvatljivo, a može se i matematički dokazati.

U implementaciji je potrebno pripaziti na još neke detalje. Primjerice, novi ciljani ostatak $M - A$ može ispasti negativan, pa mu u tom slučaju moramo dodavati broj K dok ne postane nenegativan. Nadalje, treba pripaziti na krajnje slučajeve: ako N postane negativan vjerojatnost je 0; zatim, kad je $A = B + 1$ (tj. kad interval više ne sadrži nikakve brojeve), tražena vjerojatnost iznosi 1 ako je $N = 0$ i $M = 0$ (jer zbroj 0 brojeva daje ostatak 0 pri dijeljenju s K, pa je traženi ostatak M sigurno ostvaren), a inače je 0 (jer za $N > 0$ ne možemo odabrati još brojeva, a za $M \neq 0$ ne možemo ostvariti traženi ostatak).

Budući da je prijelaz konstantne složenosti, složenost algoritma je broj stanja dinamike: $O(B \cdot N \cdot K)$.

Matrice (P1-5, P2-4)

Neka je $G(A, B)$ zbroj brojeva u najvećem pravokutniku čije je donje desno polje (A, B). Tada imamo relaciju:

$$F(A, B) = \max\{F(A - 1, B), F(A, B - 1), G(A, B)\}.$$

Naime, za pravokutnik s najvećim zbrojem u maloj matrici određenoj poljem (A, B) vrijedi: ili se nalazi u maloj matrici određenoj poljem $(A - 1, B)$, ili se nalazi u maloj matrici određenoj poljem

$(A, B-1)$ - a možda i u obje, ili pokriva polje (A, B) pa je prema tome uračunat u $G(A, B)$. Pritom valja imati na umu: za $A = 1$ polje $(A-1, B)$ ne postoji, pa ga ne treba uračunati u gornju formulu (i analogno za $B = 1$).

Trebamo dakle izračunati vrijednosti funkcije G. Odaberimo na sve moguće načine prvi i posljednji stupac nekog pravokutnika u velikoj matrici (označimo ih s K i L). Zatim za svaki redak M izračunajmo najveći zbroj u nekom takvom pravokutniku, koji završava u retku M. Kako? Dinamički: takav pravokutnik za redak M jest ili sam taj redak od K-tog do L-tog stupca, ili je to pravokutnik s najvećim zbrojem koji završava u retku $M-1$ uvećan za M-ti redak. Kad smo to izračunali, ažurirat ćemo vrijednost $G(M, L)$, koja će se povećati ako je zbroj u upravo nađenom pravokutniku veći od dotad zapisane vrijednosti u $G(M, L)$.

Da bismo gornji algoritam efikasno proveli, trebamo biti u mogućnosti brzo računati zbroj brojeva u nekom (M-tom) retku od K-tog do L-tog stupca. To ćemo riješiti na sljedeći način: izračunamo li unaprijed zbrojeve prvih Q polja u svakom retku (za svaki Q), tada je traženi zbroj zapravo razlika zbroja prvih L polja u M-tom retku i zbroja prvih $K-1$ polja u M-tom retku.

Nakon gornjeg algoritma, za svako polje (A, B) pisat će točna vrijednost $G(A, B)$, jer svaki pravokutnik koji nekom polju (A, B) pridaje tu vrijednost bit će nekada obrađen u gornjem postupku. Pomoću vrijednosti $G(A, B)$ računamo vrijednosti $F(A, B)$ dinamički po formuli iz prvog odlomka. Ukupna složenost jednaka je složenosti algoritma koji računa vrijednosti G te iznosi $O(N^3)$.

Arhitekti (P2-5)

Potpunim izbacivanjem nekog arhitekta rješenje se ne može povećati. Primijetimo stoga da, ako za arhitekta (x, y) postoji arhitekt (x', y') takav da je $x \leq x'$ i $y \leq y'$, možemo ih staviti u istu skupinu i tada prvi arhitekt ne igra nikakvu ulogu pa ga možemo i izbaciti. Dakle, promatrajmo samo one arhitekte za koje ne postoji neki drugi arhitekt koji je bolji ili jednak u oba nacrta.

Promatrane arhitekte sortirajmo uzlazno po broju katova koji predviđaju (to je prvi broj u uređenom paru koji označava arhitekta). Dokažimo da će u tom poretku brojevi stanova koje arhitekti predviđaju (to je drugi broj u uređenom paru koji označava arhitekta) biti padajući. U suprotnom bi postojala dva arhitekta (x_1, y_1) i (x_2, y_2) takva da je prvi ispred drugoga u načinjenom poretku (što znači $x_1 \leq x_2$) i da drugi predviđa veći broj stanova (što znači $y_1 < y_2$). No to nije moguće jer bismo tada prvog arhitekta već bili izbacili.

Primijetimo dalje: uzmemo li arhitekte (x_1, y_1) i (x_3, y_3) u istu skupinu, tada nam se isplati u istu tu skupinu uzeti i arhitekta (x_2, y_2) ako se on u načinjenom poretku nalazi između prve dvojice, jer on tada ne igra nikakvu ulogu zbog $x_2 \leq x_3$ i $y_2 \leq y_1$ pa je tako praktički izbačen. Dakle: isplati se arhitekte podijeliti u skupine tako da u istoj skupini budu uzastopni arhitekti u načinjenom poretku.

Sad se nameće rješenje dinamičkim programiranjem: ako se trenutno nalazimo na arhitektu A (i pritom smo sve prethodne već smjestili u skupine), onda na sve moguće načine biramo koliko ćemo arhitekata uzeti u skupinu zajedno s A (to su arhitekti koji u načinjenom poretku slijede uzastopno nakon A), izračunamo koliko će stanova sagraditi njihova skupina i prelazimo na dinamičko stanje za prvog neodabranog arhitekta. Tako imamo $O(N)$ stanja i prijelaz je složenosti $O(N)$, pa je ukupna složenost algoritma $O(N^2)$.