Univerzitetni programerski maraton 2021
3. kolo — rešitve nalog

Tomaž Hočevar
tomaz.hocevar@fri.uni-lj.si

7. oktober 2021

Vpis

Za vsak program hranimo seznam prijavljenih kandidatov, ki ga uredimo po njihovih točkah. Podatkov je dovolj malo, da lahko to počnemo na poljuben način. Prav nam pridejo slovarji (npr. map, dict, HashMap). Če število prijav ne presega števila mest, smo končali. Sicer pa moramo ugotoviti, koliko oseb ima enako število točk, kot V-ti v seznamu, če je V razpoložljivih mest. Če je samo en, potem njegove točke predstavljajo omejitev vpisa. Sicer pa moramo poiskati prvo osebo, ki ima več točk od kandidata. Če takega ni, ne moremo vpisati nikogar.

GCD zaporedje

Začetek zaporedja lahko izračunamo neposredno po formuli. Podani primer pa namiguje, da se pri večjih številih pojavi ponavljajoč vzorec x, 2, 1, y, $x+8$, 2, 1, $y+4$, ...

Da se zaporedje res ponavlja, lahko dokažemo z opazovanjem, kaj se zgodi, ko dobimo pri sodem številu $n = 2m$ vrednost $f(n) = 1$. Pri vrednosti $n = 2m+4$, ki je prav tako soda, se zopet ponovi $f(n) = 1$.

n	$2m$	$2m+1$	$2m+2$	$2m+3$	$2m+4$
$f(n)$	1	$2m+3$	$4m+6$	2	1

Vsote števk

Naloga sprašuje po podmnožicah števil od 1 do m. Vsakemu številu x iz te množice pripada vrednost, ki je enaka vsoti njegovih števk $v(x)$. Vsota vrednosti izbranih števil mora biti enaka n. Gre za problem, ki je podoben nahrbtniku. Tudi rešujemo ga na podoben način, ker so vrednosti celoštevilske in dovolj majhne.

Naj bo $f(i,k)$ število podmnožic števil od 1 do i z vsoto vrednosti enako k. Pri tem je i-to število del podmnožice ali pa ne: $f(i,k) = f(i-1,k-v(i))+f(i-1,k)$. Časovna zahtevnost algoritma je $O(nm)$, prostorsko pa lahko zmanjšamo na $O(n)$, ker za izračun vrednosti $f(i,*)$ potrebujemo samo prejšnji "stolpec" tabele $f(i-1,*)$.

Mikroorganizmi

Najprej moramo iz slike izločiti posamezne mikroorganizme oz. povezane komponente. To lahko storimo s preiskovanjem ali poplavljanjem (flood fill) v širino ali globino. Posameznih komponent ne moremo predstaviti z mrežo zasedenih polj, ker v primeru npr. diagonalnih organizmov porabijo preveč prostora in časa.

Organizme predstavimo s seznamom koordinat zasedenih polj. Za vsak organizem bomo izračunali normalizirano predstavitev, da jih bomo lahko primerjali med seboj. Najprej organizme poravnamo tako, da bo najbolj levo polje na x-koordinati 0, najbolj spodnje pa na y-koordinati 0. Zasedene koordinate uredimo. Za primerjanje organizmov med seboj pa izračunamo hash vrednost zasedenih koordinat in se

tako izognemo primerjanju seznamov. Koordinate organizma lahko štirikrat zavrtimo za 90 stopinj in izberemo kot normalizirano predstavitev tisto z najmanjšo hash vrednostjo.

Sedaj nas zanima še število različnih normaliziranih organizmov. Število organizmov je lahko preveliko, da bi primerjali vsakega z vsakim. Izkoristimo pa lahko njihovo predstavitev s hash vrednostmi, ki jih vstavimo v množico z logaritemsko ali boljšo časovno zahtevnostjo. Odgovor, ki ga iščemo, je velikost množice.

Rabutanje

Stanje v procesu prenašanja jabolk lahko opišemo s podmnožico jabolk j, ki so trenutno na vrtu, in indikatorjema b in c, ki povesta, ali smo že uporabili prvo oz. drugo operacijo. Iščemo najkrajšo pot do stanja, kjer ni na vrtu več nobenega jabolka. Dolžina poti pa je definirana s številom odhodov z vrta.

Problem lahko modeliramo z grafom stanj, povezave pa predstavljajo odnašanje ali prinašanje podmnožice jabolk. Prva operacija ima ceno 0, druga pa 1. V takem grafu lahko najkrajšo pot poiščemo z enostavnim iskanjem v širino. Spreminjanje vonja ima smisel samo pri jabolkih, ki jih nameravamo v naslednji potezi odnesti z vrta. Obravnavati moramo vse podmnožice jabolk, ki jih ni na vrtu in bi jih lahko prinesli nazaj. Prav tako pa moramo obravnavati vse podmnožice jabolk, ki so na vrtu, hkrati pa še vse možnosti spreminjanja vonja enega oz. dveh izmed teh jabolk. Ker imamo opravka s preverjanjem podmnožic podmnožic, to doprinese faktor $3^n \cdot n$, poleg tega pa še n^2 zaradi izbire jabolka oz. para jabolk, ki jima želimo spremeniti vrednost. V praksi pa je to število precej manjše, ker ni smiselno spreminjati jabolk, ki jih za tem ne odnesemo ven ali pa jih sploh ni na vrtu. Prav nam pridejo tudi bitne operacije za manipulacijo podmnožic jabolk, katerih prisotnost zakodiramo po posameznih bitih.

Vse računske operacije lahko izvedemo v celih številih. Paziti moramo, da imata lahko dve jabolki necelo vrednost po spremembah, njuna vsota pa je celo število. Dejanske operacije lahko rekonstruiramo po poti od končnega stanja proti začetku, če si na vsaki povezavi shranimo tudi operacijo, ki jo moramo izvesti.

Jamarja

Jamaraja A in B izvajata preiskovanje jame po principu iskanja v globino (depth-first search). Ker vedno preiščeta celo poddrevo in se kasneje ne vračata vanj, lahko problem razbijemo rekurzivno na podprobleme na posameznih poddrevesih. Recimo, da nas zanima optimalna rešitev podproblema s korenom v vozlišču x, pri čemer je vozlišče x raziskal jamar A, povezavo v poddrevo pa izbira drugi jamar B. Poddrevesa z visokim številom točk za jamarja niso tako zanimiva, če hkrati tudi drugi jamar doseže visoko število točk ob optimalnih izbirah. Pomembna je razlika med doseženimi točkami. Opazimo lahko tudi, da bo isti jamar ponovno na vrsti za izbor poddrevesa, če je imelo prejšnje poddrevo sodo število vozlišč. Jamarja namreč zamenjata vlogi ob vsaki vrnitvi po rovu proti korenu.

Definirajmo $f(x)$ kot razliko med točkami jamarja A in B, če je na vrsti za raziskovanje dvorane x jamar A. Ta bo zagotovo prejel Z_x točk.

- Jamarju B, ki je na vrsti za izbor poddrevesa, se splača obiskati vse poddrevesa (s korenom y) sode velikosti in negativnim $f(y)$. Po vrnitvi iz poddrevesa bo namreč na vrsti za izbor povezave isti jamar, hkrati pa je povečal razliko do nasprotnika.
- Poddreves sode velikosti in pozitivnim $f(y)$ se oba izogibata, ker jima na koristijo. Vse take primere lahko združimo v vrednost g, ker jih bo enega za drugim prisiljen izbrati kdorkoli od njiju ne bo imel druge izbire.
- Ostanejo še poddrevesa lihe velikosti. Tu imata oba jamarja enako taktiko izbiranja poddrevesa z minimalno vrednostjo (ki najbolj škodi nasprotniku). Izbirala jih bosta torej izmenično od minimalnih proti maksimalnim vrednostim $f(y)$.

Jamar A bo torej pobral Z_x. Nato bo jamar B izbral vsa poddrevesa sode velikosti z negativnim $f(y)$. Nato bosta izmenično izbirala poddrevesa lihe velikosti od manjših proti večjim vrednostim $f(y)$. Na koncu pa bo prisiljen kdorkoli je na vrsti izbrati še vsa poddrevesa sode velikosti s pozitivnim $f(y)$. Iz vrednosti $f(1)$ lahko enostavno izračunamo točke posameznega jamarja, ker poznamo skupno število točk.