Univerzitetni programerski maraton 2020
3. kolo — rešitve nalog

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

1. oktober 2020

Čini

Podan je seznam oseb, njihovih činov ter drevesna struktura relacij med osebami. Za vsako osebo moramo izračunati oddaljenost od korena drevesa. Preostane nam samo še urejanje oseb glede na podane kriterije - primarno glede na čin, nato glede na pod/nad čin in nazadnje glede na izračunano oddaljenost. Če imata v urejenem seznamu dve zaporedni osebi povsem enake kriterije, rešitev ne obstaja.

Travca

Z analizo primera na mreži 8x4 z odgovorom 6, lahko pridemo do splošne strategije. Če je vrt kvadratne oblike, lahko zasadimo samo diagonalo in se zaraste cel vrt. V primeru vodoravnega pravokotnika, pa moramo poskrbeti še za preostanek s sajenjem trave v enem kvadratku vsakega drugega preostalega stolpca.

Za diagonalo potrebujemo $d = \min(n, m)$ kvadratkov, s čimer zasadimo kvadrat dimenzije $d \times d$. Ostane nam $r = max(n, m) - d$ vrstic ali stolpcev, kjer moramo zasaditi vsakega drugega. Rešitev je torej $d + \lceil \frac{r}{2} \rceil$.

Dokažimo, da je to optimalen rezultat. V vsakem koraku se obseg vseh zaplat trave zmanjša ali ostane enak, ker novo zarasel kvadrat meji na vsaj 2 prejšnja. Končen obseg travnika je $2(n + m)$, torej moramo zasaditi travo na vsaj $\lceil \frac{2(n+m)}{4} \rceil$ kvadratkih. Prej opisana strategija doseže ravno to število.

Ulam

Naj bo U število Ulamovih števil. Za vsako izmed N števil po vrsti lahko preverimo, ali ga je mogoče zapisati kot vsoto dveh že odkritih Ulamovih števil. Naivna rešitev bi potrebovala $O(NU^2)$ časa, kar lahko enostavno izboljšamo na $O(NU)$. Recimo, da testiramo število n. Pri izbiri prvega Ulamovega števila x, lahko v $O(1)$ ugotovimo, ali je $n - x$ tudi Ulamovo število. Žal pa tudi to ni dovolj hitro.

Ker je U precej manjši od N, lahko namesto testiranja števil uporabimo filtriranje po principu sita. Ko odkrijemo neko Ulamovo število u, lahko izločimo vse vsote $u + v$, kjer je v manjše Ulamovo število. Potrebujemo samo seznam Ulamovih števil. Časovna zahtevnost take rešitve je $O(U^2 + N)$. Z vsakim Ulamovim številom imamo $O(U)$ dela, za vsako izmed N števil pa v konstantnem času ugotovimo, ali je bilo izločeno ali ne.

Absolutna resnica

Naš cilj je prešteti izraze z oklepaji, ki bodo imeli resnično vrednost. Izberemo lahko najbolj zunanji operator, kar nam problem razdeli na dva manjša podproblema (oz. enega v primeru negacije). Če je bil zunanji operator XOR, mora biti levi izraz resničen, desni pa neresničen oz. obratno. Tako lahko definiramo $O(n^2)$ podproblemov $f(i, j, t)$ - na koliko načinov lahko izrazu med i-tim in j-tim členom dodamo oklepaje, da bo imel resnično vrednost t. Podproblem razbijemo na dva manjša z izbiro zunanjega operatorja, za kar imamo $O(n)$ možnosti. Paziti moramo še ne možnost, da (pod)izraz ni veljaven. Podprobleme rešujemo z dinamičnim programiranjem od krajših proti daljšim izrazom ali pa uporabimo memoizacijo.

Scenografija

Podane imamo točke in poizvedbe o številu točk znotraj določenega območja. Območja so v obliki enakokrakih trikotnikov, ki se od svojega vrha raztezajo do tal. Ker so si stranice trikotnikov vzporedne, lahko izvedemo linearno transformacijo prostora, da dobimo pravokotne poizvedbe (q_x, q_y) o točkah (x, y), za katere velja $x \leq q_x \wedge y \leq q_y$.

Poizvedbe in točke uredimo po x-koordinati in jih obdelujemo po vrsti od manjših proti večjim x-om. Ob poizvedbi (q_x, q_y) nas zanima, koliko točk smo že obravnavali, za katere velja $y \leq q_y$ (drugi pogoj $x \leq q_x$ je namreč resničen za vse prejšnje točke). Obravnavane točke hranimo v drevesni strukturi, kar nam omogoča dodajanje in preštevanje točk v $O(\log n)$. Ker potrebujemo samo predponske poizvedbe, je dober kandidat Fenwickovo drevo.

Enakovredna rešitev je, da namesto eksplicitne transformacije obravnavamo točke in poizvedbe v drugačnem koordinatnem sistemu, kjer je prva os pravokotna na desne stranice trikotnikov (urejeno po vrednosti $2x + y$), druga pa na leve stranice (urejeno po vrednosti $-2x + y$). Tako premikamo "desne stranice trikotnikov" od leve proti desni, obravnavane točke pa hranimo urejene glede na "leve stranice" od spodaj navzgor.

Mikado

Za vsak par daljic določimo, ali se sekata ali ne. Pri tem nam ni treba izračunati dejanskega presečišča. Če sta daljici kolinearni, preverimo, ali se sekata očrtana pravokotnika oz. projekciji daljic na x-os in y-os. Za nekolinearen primer pa z vektorskim produktom preverimo, ali sta obe krajišči druge daljice na isti strani prve daljice ter obratno.

Naj spremenljivka D_i označuje, ali bomo obdržali i-to daljico. Če se sekata daljici i in j, moramo vsaj eno od njiju odstraniti, kar zapišemo kot $\neg D_i \vee \neg D_j$. Izmed vseh daljic iste barve pa lahko odstranimo največ eno. Za vse pare daljic iste barve (npr. k in l) mora torej veljati, da je vsaj ena prisotna, kar zapišemo kot $D_k \vee D_l$. S tem smo dobili izraz v konjunktivni normalni obliki (CNF), kjer v vseh disjunkcijah nastopata samo dve spremenljivki (2-CNF). Ugotavljanje izpolnjivosti takih formul je klasičen 2-SAT problem.

2-SAT problem izpolnjivosti lahko rešimo v linearnem času. Zgradimo graf implikacij in v njem poiščemo močno povezane komponente. Če sta spremenljivka in njena negacija del iste komponente, rešitev ne obstaja. Sicer lahko komponentam v obratnem topološkem vrstnem redu določamo logične vrednosti s požrešno strategijo.