Univerzitetni programerski maraton 2023
2. kolo — rešitve nalog

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

11. maj 2023

Zemljevid

Preberemo tabelo znakov in preštejemo, koliko jih predstavlja kopno, koliko morje in koliko je ostalih.

Rastrski krogi

V iskanem območju, ki je velikosti največ 100×100, moramo za vsak piksel ugotoviti, ali pripada črti, notranjosti ali zunanjosti kroga. Za določitev barve posameznega piksla preverimo vsa štiri njegova oglišča. Zaradi celoštevilskih koordinat se ne more zgoditi, da bi krog vseboval samo kakšno stranico piksla, ne pa tudi oglišča. Glede na oddaljenost oglišča od središča kroga lahko določimo, ali je na robu, izven kroga ali znotraj. Če je kakšno oglišče na robu ali če obstaja par oglišč, kjer je eno znotraj drugo pa zunaj, pobarvamo piksel z barvo roba. Če je kakšno oglišče znotraj, ga pobarvamo z barvo notranjosti, sicer pa z barvo zunanjosti.

Brisanje

V optimalni rešitvi bo obstajala ločnica med elementi seznama, ki bodo prispevali k prvemu členu in elementi, ki prispevajo k drugemu členu. Preverimo lahko torej vsa razbitja seznama na levi in desni del. Izraz bo imel največjo vrednost, če iz levega dela izberemo $M/2$ največjih elementov, iz desnega pa $M/2$ najmanjših. Vsoto kvadratov $M/2$ največjih elementov lahko učinkovito vzdržujemo pri povečevanju množice oz. pri premikanju ločnice od leve proti desni. Hranimo jih urejene v drevesni strukturi ali prioritetni vrsti, ki hrani največjih $M/2$ elementov. Ko dodamo nov element, iz strukture samo odstranimo najmanjšega. Podobno naredimo še v drugi smeri za izračun vsote kvadratov $M/2$ najmanjših elementov v vedno daljših priponah seznama. Pazite, ker je lahko rešitev tudi negativna. Časovna zahtevnost postopka je $O(n \log n)$.

Vezja

Naloga je predvsem implementacijska vaja iz rekurzije. Začetek vezja najdemo z najbolj levim znakom '-'. Zaporedno vezavo bomo obdelali iterativno v zanki. Vsak osnovni element 'o' dodamo v seznam zaporedno vezanih elementov, če naletimo na '+', pa gre za vejitev in dodamo rekurziven rezultat vzporedno vezanih podvezav, katerih začetke najdemo kot '+' v istem stolpcu. Do težave pride, ker težko ugotovimo, kje se vzporedna vezava zaključi. Rekurzivni klici za vzporedne vezave lahko vrnejo tudi x-koordinate, kjer je prišlo do zaključka obravnavane vezave. Vse vrnjene vrednosti bi morale biti seveda enake. Ugotoviti moramo, kje se torej zaključi neko vezje oz. zaporedna vezava. Lahko zmanjka povezav in naletimo na prazno mesto, ali pa gre za konec podvezja, ki je del vzporedne vezave, in se bo zato zaključilo z znakom '+'. Pri znaku '+' torej ne vemo, ali gre za začetek ali konec vejitve. To lahko ugotovimo tako, da poiščemo najbolj zgornji '+' vejitve v istem stolpcu in preverimo znaka levo in desno od tega stolpca, saj bo en od njih enak '-' in lahko iz tega sklepamo o začetku oz. koncu vejitve.

Tetracija

Pri potenciranju števila a po modulu m, se bodo ostanki hitro začeli ponavljati (najkasneje po m množenjih). Recimo, da se cikel začne po d korakih in je dolžine c. Z enostavnim naivnim izračunom lahko ugotovimo, ali bo potenciranje sploh doseglo cikel. Izračun tetracije preprosto ustavimo, ko rezultat preseže mejo d. Sicer vemo, da bomo dosegli cikel in nas sedaj zanima ostanek potence $a \uparrow\uparrow (b-1)$ po modulu c. Ta problem rekurzivno rešujemo po enakem principu. Recimo, da je izračunan ostanek potence enak x. Dejansko mesto v periodi je zaradi začetnega dela pred ciklom enako $f = (x - d) \mod c$, iskani rezultat pa $a^{d+f} \mod c$.

Alternativna rešitev uporablja kitajski izrek o ostankih. Nalogo je enostavneje rešiti, če sta si a in m tuja. Če nista, modul m faktoriziramo v potence praštevil ($m = p_1^{k_1} p_2^{k_2} \ldots$) in izračunamo ostanek po modulu vsakega od teh faktorjev (p^k), kar nato s kitajskim izrekom o ostankih združimo v ostanek po originalnem modulu m. Če a ni deljiv s praštevilom p, imamo enostaven primer in moramo izračunati tetracijo $a \uparrow\uparrow (b-1)$ po modulu $\varphi(p^k)$. Če pa je a deljiv s p, bo potenciranje števila a hitro pripeljalo do tega, da bo število praštevil p v potenci a-ja preseglo k in bo zato iskani ostanek 0.

Dvovladje

Naloga sprašuje po maksimalni vsoti uteži povezav, ki jih lahko dodamo drevesu, ne da bi pri tem ustvarili cikel z sodim številom povezav. Najprej ugotovimo, da pride posamezna povezava v poštev samo, če je pot v drevesu med njenima krajiščema sode dolžine. Vse ostale povezave lahko ignoriramo. Za izračun dolžin poti po drevesu, si vnaprej izračunamo globine vozlišč in najnižje skupne prednike parov vozlišče v $O(n^2)$. Sedaj vemo, katere povezave lahko dodamo drevesu.

Kaj pa se zgodi, če jih dodamo več hkrati? Ugotovimo lahko, da v primeru, da si dva ustvarjena liha cikli delita kakšno povezavo, dobimo tudi cikel sode dolžine. Recimo, da sta cikla dolžine l_1 in l_2, ki si delita pot po drevesu dolžine p. Dolžina cikla brez skupne poti je $l_1 + l_2 - 2p$, kar je sodo število (l_1 in l_2 sta liha, $2p$ pa je sodo).

Formirati želimo torej disjunktne cikle (glede na povezave, ne pa nujno tudi vozlišča) s čim večjo utežjo dodanih povezav. Naredimo rekurzivni razmislek na korenu drevesa. Če ne uporabimo nobene povezave, pri kateri bi pot med krajišči šla skozi koren, lahko problem rešujemo neodvisno v vsakem poddrevesu. Sicer pa lahko za vsako tako povezavo prerežemo vse povezave na poti in rešujemo problem na poddrevesih z nekaterimi prerezanimi povezavami. Ker ima vsako vozlišče stopnjo največ $d \leq 12$, moramo rešiti $O(n2^d)$ podproblemov. Pri tem moramo preveriti vse povezave, katerih najvišje vozlišče na poti je ravno obravnavni koren drevesa, in sešteti rešitve podproblemov na poti po drevesu med krajiščema oz. od vsakega krajišča do korena. Časovna zahtevnost te rešitve je $O(m2^d n)$, kar je prepočasi. Opazimo pa lahko, da so vsote rešitev podproblemov neodvisne od množice prerezanih vozlišč pri korenu, zato si jih lahko izračunamo v naprej (pred obravnavo vseh podmnožic prerezanih povezav iz vozlišča) in dobimo rešitev s časovno zahtevnostjo $O(m2^d + n^2)$.