Univerzitetni programerski maraton 2022
3. kolo — rešitve nalog

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

6. oktober 2022

Temperatura

Gre za enostavno nalogo, v kateri moramo samo narediti, kar piše, pa še to ni prav težko. Vrstice razbijemo na presledkih, da dobimo datum, uro in vrednost vsake meritve, nato pa meritve združimo v skupine po datumih. Za vsako skupino izračunamo minimum, maksimum in povprečje, ki ga moramo primerno zaokrožiti.

Nakup kriptovalut

UPM kovanec lahko kupimo na N različnih borzah. Opazimo, da nam vsaka borza za vsak naslednji kupljeni kovanec zaračuna več kot za prejšnjega. Da bi kupili M UPM kovancev, bomo v prvem koraku izbrali borzo z najcenejšo ponudbo. Po nakupu kovanca popravimo ponudbo borze in postopek ponovimo M-krat.

Težava nastane v ponavljajočem se iskanju borze s trenutno najcenejšo ponudbo. Vse ponudbe lahko hranimo v prioritetni vrsti ali kakšni drugi urejeni drevesni strukturi, ki nam omogoča iskanje najmanjšega elementa in spreminjanje le-tega, kar lahko dosežemo tudi z odstranitvijo in vstavitvijo spremenjenega.

Merjenje razdalje

Podan imamo opis poti z zaporedjem točk. Pot moramo hraniti v strukturi, ki nam bo omogočala učinkovit izračun dolžine celotne poti ter dodajanje in brisanje vmesnih ali končnih točk (premik dosežemo z izbrisom in dodajanjem na novo mesto). Točke lahko hranimo v povezanem seznamu (linked list), kjer vsak element hrani tudi referenco na svojega predhodnika in naslednika. Pri vstavljanju poiščemo element v povezanem seznamu preko njegove identifikacijske številke in ga vstavimo v seznam, pri čemer se popravijo predhodniki in nasledniki v okolici vstavljanja. Ker gre za lokalno spremembo, lahko pri tem popravimo tudi dolžino poti - odštejemo dolžine povezav, ki so razpadle in prištejemo tiste, ki so vzpostavljene na novo.

Višine učencev

Predpostavimo, da so višine učencev permutacija števil od 1 do N. Preštejmo primere, kjer je zadnji v vrsti učenec x. Pred njim mora biti torej vrsta sestavljena iz višin $\{1, \ldots, N\} \setminus \{x\}$, ki se zaključi z učencem y, ki je višji ali nižji od x (odvisno od podane relacije). Izgleda, da je naš podproblem definiran s podmnožico učencev in zadnjim v vrsti. Vendar pa lahko to podmnožico preslikamo v permutacijo števil od 1 do $N-1$, ne da bi spremenili število rešitev. Tako imamo samo $O(n^2)$ podproblemov, vsak podproblem pa rešimo z obravnavo vseh $O(n)$ možnih predzadnjih elementov.

$$f(n,x) = \begin{cases} \sum_{y=1}^{x-1} f(n-1,y) & \text{if } s_{n-1} = \text{'<'} \\ \sum_{y=x}^{n-1} f(n-1,y) & \text{if } s_{n-1} = \text{'>'} \end{cases}$$

Obseg pravokotnika

Pravokotnik lahko formiramo v štiri smeri. Osredotočimo se na primer, ko se pravokotnik od klika razteza levo in navzdol (3. kvadrant) proti neki sidriščni točki. Ostale primere pokrijemo z uporabi istega postopka na podatkih zarotiranih za 90°.

Opazujmo klik na koordinati (a_i, b_i). Obseg pravokotnika je v tem primeru $2 * ((a_i - x_j) + (b_i - y_j))$. Ta vrednost bo največja, če minimiziramo $x_j + y_j$. Iščemo torej čim bolj oddaljeno diagonalo, ki vsebuje sidriščno točko, za katero velja $x_j < a_i$ in $y_j < b_i$.

Točke (klike in sidrišča) obravnavamo po naraščajočih vrednostih x koordinat. Tako je zagotovljeno, da vse do sedaj obravnavane točke zadoščajo $x_j < a_i$. Med obravnavanimi točkami z $y_j < b_i$ pa moramo poiskati tisto z minimalno vrednostjo $x_j + y_j$. V ta namen lahko uporabimo Fenwickovo drevo ali kakšno drugo drevesno strukturo.

Sestavljanka

V tej nalogi je več implementacijskega dela kot razmisleka o rešitvi.

- Poiskati moramo povezane komponente praznih mest na plošči, da najdemo luknje.
- Pripravimo si vse rotacije in zrcaljenja koščkov.
- Za vsako luknjo in orientacijo koščka moramo ugotoviti, ali košček paše v luknjo. Izberemo si poljubno polno polje koščka, ga poskusimo poravnati z vsemi praznimi polji luknje in preverimo, ali se v celoti ujemata.
- Poiskati moramo prirejanje koščkov in lukenj. Razne požrešne strategije ne delujejo. Uporabimo lahko algoritem za iskanje največjega ujemanja v dvodelnem grafu. Ker je malo lukenj in koščkov, lahko uporabimo tudi kakšno pametno različico rekurzivnega izčrpnega preiskovanja, kjer na vsakem koraku poskusimo prirediti par elementu z najmanj možnostmi.
- Poskrbeti moramo še za primeren izpis. Prav nam pride, če si zapomnimo transformacijo koščka, ki je vodila do ujemanja med luknjo in koščkom.