Univerzitetni programerski maraton 2023
3. kolo — rešitve nalog

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

5. oktober 2023

Raznobarvna naloga

Naloga zahteva pretvarjanje med tremi različnimi formati barv: desetiškim, šestnajstiškim in decimalnim. Največjo oviro predstavlja šestnajstiški sistem. Lahko implementiramo pretvarjanje v in iz šestnajstiškega sistema, ali pa si pomagamo kar s formati za branje in pisanje ("d", "02x", ".8f").

Dvojiško iskalno drevo

Podan je graf v obliki, ki za vsako vozlišče navaja levega in desnega otroka. Preveriti moramo, ali gre za drevo in ali je iskalno, torej so vrednosti pravilno urejene v njem. V drevesu imajo vsa vozlišča razen korena vhodno stopnjo 1, koren pa ima vhodno stopnjo 0. Strukturo lahko torej enostavno preverimo s štetjem vhodnih stopenj. Glede razporeditve vrednost pa lahko preverimo, da je vozlišče večje od vseh na levi strani in manjše od vseh na desni. Pozor, ni dovolj preverjati samo, ali je vozlišče večje od levega otroka in manjše od desnega. Lahko pa enostavno izvedemo vmesni (in-order) obhod in preverimo, ali je rezultat urejen seznam.

Izris drevesa

Gre za vajo iz rekurzije. Način izrisa je natančno opisan že v nalogi. Pripravimo si dovolj veliko polje in nanj vpisujemo primerne znake. Glavni izziv je branje opisa strukture. Opis drevesa moramo razbiti na opis levega in desnega poddrevesa. Mejo med njima lahko najdemo z vejico, ki ni gnezdena v kakšnih oklepajih. Nivo gnezdenja lahko preverimo s štetjem oklepajev in zaklepajev. Kjer je razlika 0, vsebina ni znotraj oklepajev.

Police na zidu

Na zid dodajamo in odstranjujemo police, pri tem pa moramo odgovarjati na poizvedbe, kakšna je največja razdalja od vrha zidu do najvišje police na tisti koordinati. Izkoristimo lahko lastnost, da so police dokaj kratke. Za vsako x koordinato (recimo ji stolpec) bomo hranili urejeno množico višin, ki so zasedene s policami. Ker bomo dodajali in odstranjevali elemente, lahko za to uporabimo neko drevesno strukturo (npr. multiset). Sedaj bi morali za odgovor na poizvedbo preveriti vse stolpce. Lahko pa vzdržujemo dodatno drevesno strukturo, ki hrani urejene rezultate za vsak posamezni stolpec, za kar je spet primeren npr. multiset. Pri vsaki spremembi moramo torej popraviti strukture stolpcev in globalno strukturo.

Kockice

Rekonstruirali bomo postavitev kockic in nato odgovarjali na poizvedbe o razdaljah. Kocke lahko po vrsti zlagamo po nivojih od spodaj navzgor, znotraj nivoja po vrsticah in znotraj vrstice od leve proti desni. Vogalna kocka mora imeti na stranicah tri ničle. Izberemo lahko katerokoli in jo orientiramo tako, da

so ničle na zunanjih ploskvah. Nato lahko določimo njeno sosedo, ki ima pravo vrednost na skupni stranici. Preizkusimo vse orientacije in izberemo tisto, ki se ujema s prej postavljenimi kockami in zunanjimi ploskvami. Če obstaja več orientacij, lahko izberemo katerokoli. Ideja je enostavna, izvedba pa nekoliko težja. Prav nam pride vnaprejšnja priprava vseh možnih rotacij kockic. V ta namen si pripravimo pomožne funkcije za rotacijo kocke v vsaki od treh dimenzij.

Povezanost

Za rešitev bomo uporabili dinamično programiranje s podmnožicami. Za vsako množico vozlišč S bomo izračunali verjetnost $f(S)$, da formirajo povezan graf. To verjetnost izračunamo tako, da od 1 odštejemo verjetnosti, da najmanjše vozlišče v množici (recimo mu x) formira svojo povezano komponento T. Komponenta T mora biti povezana, povezave iz komponente T v preostanek vozlišč $S \setminus T$ ne smejo obstajati, ostale povezave pa niso pomembne. Prvo vrednost lahko izračunamo kot $f(T)$, za drugo vrednost pa moramo izračunati število povezav med obema deloma. Če $g(X)$ predstavlja število povezav med vozlišči iz množice X, lahko izračunamo število povezav kot $g(S) - g(T) - g(S \setminus T)$. Funkcijo g lahko tabeliramo vnaprej. Zaradi preverjanja podmnožic vsake podmnožice, je časovna zahtevnost $O(3^n)$, kar se zdi prepočasi za $n = 19$. Vendar mora biti najmanjši element x vedno del podmnožice T, zato smo lahko nekoliko hitrejši.