Domača naloga #3: seznami
Priprave na računalniške olimpijade 2018/19

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

A. Anagram Search

Kako ugotovimo, ali je niz p anagram niza s? Preštejemo pojavitve posameznih črk v njih, če se ujemajo, sta anagrama, sicer pa ne. Kaj pa, če se v nizu s lahko pojavijo vprašaji, ki nadomestijo poljubno črko. Če sta niza enako dolga in pogostost posamezne črke v nizu p ne preseže pogostosti te črke v s, lahko preostanek kompenziramo z vprašaji in sta niza anagrama.

Preostane nam poiskati vse podnize s-ja, ki so anagrami p-ja ($n = |s|, m = |p|$). Za vsak podniz s-ja dolžine m moramo prešteti frekvence črk in jih primerjati. Če to počnemo vsakič znova, dobimo časovno zahtevnost $O(n(m + a))$ (kjer je a velikost abecede), kar je prepočasi. Na srečo pa se frekvence črk v dveh zaporednih podnizih ne razlikujejo prav veliko. Frekvence črk naslednjega podniza lahko izračunamo z dvema popravkoma frekvenc črk v trenutnem podnizu. Taka rešitev ima časovno zahtevnost $O(m + na)$.

B. Sagheer and Nubian Market

Opravka imamo z muhastim prodajalcem, ki nam prodaja izdelke po spremenljivih cenah. Dejanske cene izdelka ne poznamo, dokler se ne odločimo, koliko izdelkov bomo kupili (k). Več izdelkov kot bomo kupili, višje bodo njihove cene in prej bomo prekoračili svoj proračun ter obratno. Naloge se lahko torej lotimo z bisekcijo. Za neko fiksno število izdelkov k lahko izračunamo dejanske cene vseh izdelkov. Seveda se nam splača kupiti k najcenejših izdelkov, da minimiziramo skupno ceno. Če pri tem ne presežemo svojega proračuna S, očitno lahko kupimo k izdelkov in nadaljujemo iskanje oz. bisekcijo z večjimi k-ji, sicer pa z manjšimi.

C. Holes

Podano imamo tabelo vrednosti $a_1, \ldots, a_n$, ki pomenijo, da se z i-tega mesta premaknemo na mesto $i + a_i$. Zanima nas, po koliko potezah bomo ob podanem začetnem mestu x padli s tabele in s katerega mesta se bo to zgodilo. Lahko bi za vsako poizvedbo simulirali skakanje po tabeli, vendar nas lahko vhod prisili, da vsakič obiščemo skoraj celo tabelo. Če si na začetku izračunamo izhodna mesta in število skokov za vsako mesto v tabeli (kar lahko storimo v linearnem času od konca proti začetku tabele), lahko na poizvedbe odgovorimo v konstantnem času. Na težavo pa naletimo ob spremembi vrednosti v tabeli, kjer je treba popraviti vsa mesta, iz katerih lahko pridemo do ravnokar spremenjenega (to so lahko ponovno skoraj vsa).

Kompromis lahko naredimo s korensko dekompozicijo (angl. *sqrt decomposition*). Mesta v tabeli razdelimo v zaporedne skupine velikost $k = \sqrt{n}$. Za vsako mesto lahko izračunamo, kje v pripadajoči skupini končamo skakanje (vzpostavimo neke vrste 'avtoceste' znotraj skupin). Pri poizvedbi naredimo premik znotraj posamezne skupine v enem koraku, premakniti pa se moramo med $O(n/k) = O(\sqrt{n})$ skupinami. Ob spremembi moramo popraviti podatke za vsa mesta v pripadajoči skupini, za kar potrebujemo prav tako $O(k) = O(\sqrt{n})$ operacij.