Opisi algoritama

Primjeri implementiranih rješenja dani su u priloženim izvornim kodovima koji nužno ne odgovaraju u svim detaljima ovdje opisanim algoritmima.

Zadatak: 2048

Primijetimo da je broj poteza mali te možemo za svaki potez smjer. To je ukupno 4^n mogućih igri. Za svaku simuliramo kako će izgledati ploča te zapamatimo najbolju ploču.

Zadatak: Geni

Za početak primjetimo da ćemo traženi gen dobiti tako da u svakom potezu uzememo najveće slovo od prvih k slova, stavimo ga na kraj i nakon nekoliko poteza, gen će biti leksikografski najmanji.

Primijetimo sada da za $k \geq 2$ na taj način možemo sortirati slova u genu.

Ostaje nam još za rješiti slučaj kada je $k = 1$. Tada zapravo tražimo najmanju leksikografsku rotaciju zadanog gena. Za 90% bodova bilo je moguće simulirati taj postupak, dok za sve bodove bilo je potrebno znati neke naprednije algoritme na stringovima.

Jedno od mogućih rješenja za sve bodove je binarnom pretragom i hash-om implementirati operator usporedbe stringova pa uz pomoć njega pronaći najmanji leksikografski gen.

Za one koji žele znati više postoji i linearno rješenje ovog problema.

Zadatak: Trik

Primijetimo da za svaku dužinu postoji interval $[L, R]$ za koji vrijedi da ako nacrtamo kružnicu radiusa r koji je iz tog intervala, tada ta kružnicu siječe tu dužinu. Za rješenje sada je potrebno za svaku dužinu odrediti njezin interval $[L, R]$. Nakon što smo to napravili problem je sljedeći: imamo n intervala, pitamo se koliko najviše intervala prekriva neku točku?

To možemo izračunati da sortiramo krajeve intervala te ih obilazimo redom. Kada naiđemo na početak intervala, brojač povećamo za 1, a kada naiđemo na kraj, smanjimo za 1.

Zadatak: Klizanje

Za rješavanje ovog problema možemo koristiti pretraživanje grafa. Svako polje na klizalištu može biti čvor u grafu, a bridovi između čvorova predstavljaju moguće poteze klizača (pomak za jedno polje u jednom od četiri smjera). Za svako polje i svaki smjeri unaprijed izračunamo do kamo bi klizač otklizao ako se pomakne u tom smjeru.

Nakon stvaranja grafa, možemo koristiti pretraživanje u širinu za pronalaženje najkraćeg puta od početnog do završnog polja za svaki upit.

Zadatak: Kockica

Glavna ideja rješenja je simulirati kretanje kockice tako dugo dok se ona opet ne nađe u korijenu stabla nakon što je obišla sve čvorove. Primjetimo da sada na svaki upit možemo dodati sumu svih vrijednosti iz podstabla zadanog čvora pomnoženu s brojem koliko puta kockica obiđe cijelo stablo u tom upitu.

Ostaje nam još za rješiti za svaki upit ostatak kretanja kockice, tj. put od korijena do čvora u kojem staje. Primijetimo da i sada nam je dovoljno znati brzo odgovoriti na pitanje koja je suma brojeva u podstablu nekog čvora.

Sumu vrijednosti čvorova u podstablu nekog čvora možemo odgovoriti tako da napravimo *preorder*

obilazak stabla te tada na odgovorajuća mjesta u nizu zapisujemo vrijednosti. Upit se tada svodi na sumu u intervalu što možemo podržati s npr. *tournament* stablom ili *logaritamskom* strukturom.