

Opisi algoritama

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

Eventualna pitanja, mišljenja ili prijedloge vezane uz zadatke uputite na adrian.kurdija@fer.hr.

Zadatak: PVC

Dvostrukom for petljom ispisujemo prozore. Vanjska for petlja prolazi po redovima, dok unutarnja for petlja prolazi po prozorima te za svaki prozor crta odgovarajuća četiri znaka za taj red. Za implementacijske detalje pogledajte priloženi kod.

Zadatak: Slova

Postavljamo početno polje $(1, 1)$ pa obilazimo matricu while petljom. Unutar te while petlje imamo još četiri while petlje za svaki smjer: redom za desno, dolje, lijevo i gore. Svaka od unutarnjih while petlji pomiče nas u odgovarajućem smjeru, sve dok ne dođemo do ruba matrice ili do polja koje smo već posjetili (posjećena polja označavamo točkom). Vanjska while petlja staje u trenutku kada se nijedna od unutarnjih petlji nije pomaknula ni za jedno polje. Za implementacijske detalje pogledajte priloženi kod.

Zadatak: Tromine

Za svaki mogući odabir uzastopnih triju redaka zasebno provjeravamo mogu li se popločati trominama. Za provjeru koristimo dinamičko programiranje. Prolazimo po stupcima slijeva nadesno održavajući niz *može*(s), koji za neki indeks stupca s ima vrijednost 0 ako nije moguće prvih s stupaca popločati trominama, a 1 ako jest.

Pretpostavimo da se u prolasku slijeva nadesno trenutačno nalazimo na stupcu i . Inicijalno postavljamo vrijednost *može*(i) na 0. Želimo li popločati prvih i stupaca trominama, vrijedit će ili da se u stupcu i nalazi okomita tromina, ili da se u stupcima $i - 2$, $i - 1$ i i nalaze tri vodoravne tromine.

Provjeravamo je li moguće postaviti okomitu trominu: ako jest, postavljamo *može*(i) na vrijednost *može*($i - 1$) jer preostaje popločati prethodnih $i - 1$ stupaca. Potom provjeravamo je li moguće postaviti tri vodoravne tromine: ako jest, postavljamo *može*(i) na vrijednost *može*(i) *OR* *može*($i - 3$) jer preostaje popločati prethodnih $i - 3$ stupaca. Za implementacijske detalje pogledajte priloženi kod.

Zadatak: Prozori

Prozore ispisujemo trostrukom for petljom. Vanjska for petlja s brojačem j prolazi po M redova prozora, srednja for petlja s brojačem i prolazi po 4 reda koji sačinjavaju prozore, a unutarnja for petlja s brojačem p prolazi po N prozora u redu j te za svaki prozor crta odgovarajuća četiri znaka za red i . Za implementacijske detalje pogledajte priloženi kod.

Zadatak: Domine

Za svaki mogući odabir uzastopnih dvaju redaka zasebno provjeravamo mogu li se popločati dominama. Za provjeru koristimo dinamičko programiranje. Prolazimo po stupcima slijeva nadesno održavajući niz *može*(s), koji za neki indeks stupca s ima vrijednost 0 ako nije moguće prvih s stupaca popločati dominama, a 1 ako jest.

Pretpostavimo da se u prolasku slijeva nadesno nalazimo na stupcu i . Inicijalno postavljamo vrijednost *može*(i) na 0. Želimo li popločati prvih i stupaca dominama, vrijedit će ili da se u stupcu i nalazi okomita domina, ili da se u stupcima $i - 1$ i i nalaze dvije vodoravne domine.

Provjeravamo je li moguće postaviti okomitu dominu: ako jest, postavljamo *može*(i) na vrijednost

$može(i - 1)$ jer preostaje popločati prethodnih $i - 1$ stupaca. Potom provjeravamo je li moguće postaviti dvije vodoravne domine: ako jest, postavljamo $može(i)$ na vrijednost $može(i)$ OR $može(i - 2)$ jer preostaje popločati prethodnih $i - 2$ stupaca. Za implementacijske detalje pogledajte priloženi kod.

Zadatak: Spirala

Opišimo najprije rješenje za 50% bodova, a potom ćemo pokazati kako ga ubrzati. Postavljamo početno polje $(1, 1)$ pa obilazimo matricu while petljom. Unutar te while petlje imamo još četiri while petlje za svaki smjer: redom za desno, dolje, lijevo i gore. Svaka od unutarnjih while petlji pomiče nas u odgovarajućem smjeru, sve dok ne dođemo do ruba matrice ili do polja koje smo već posjetili (posjećena polja označavamo točkom). Vanjska while petlja staje u trenutku kada se nijedna od unutarnjih petlji nije pomaknula ni za jedno polje. Na svakom polju koje posjetimo provjeravamo nalazi li se ono u retku X ili stupcu Y : ako da, pribrajaemo ga rješenju, pazeći da polje na presjeku brojimo samo jednom.

Složenost ovog rješenja iznosi $O(R \cdot S)$, što nije dovoljno brzo za sve bodove. Ubrzavamo ga tako što primjećujemo da unutarnje četiri while petlje ne moraju ići jedno po jedno polje, već mogu "preskočiti" sva polja odjednom, pod uvjetom da uvijek znamo početno i završno polje za svaki smjer. Početno i završno polje nam je uvijek poznato budući da ukupni obilazak spirale možemo promatrati kao obilazak jednog po jednog okvira. Najprije obilazimo vanjski okvir koji se sastoji od prvog retka i stupca ($r_1 = 1$ i $s_1 = 1$) te posljednjeg retka i stupca ($r_2 = R$ i $s_2 = S$), a svaki sljedeći okvir ima vrijednosti (r_1, s_1, r_2, s_2) pomaknute na $(r_1 + 1, s_1 + 1, r_2 - 1, s_2 - 1)$.

Ostaje nam odrediti kako prilikom preskakanja polja možemo zbrojiti sva ona koja se nalaze u retku X ili stupcu Y . Postoje dva slučaja. Jedan je da dio koji preskačemo u potpunosti pripada retku X ili stupcu Y . U tom slučaju zbrojit ćemo sve vrijednosti koje preskačemo korištenjem formule za sumu brojeva u intervalu $[a, b]$: $b(b + 1)/2 - a(a - 1)/2$. Ako taj slučaj nije ispunjen, provjeravamo siječe li preskočeni dio redak X ili stupac Y : ako da, rješenju pribrajaemo samo vrijednost sjecišta. Ukupna je složenost $O(R + S)$. Za implementacijske detalje pogledajte priloženi kod.