Univerzitetni programerski maraton 2022
2. kolo — rešitve nalog

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

5. maj 2022

Bočno parkiranje

Če opis parkirišča razrežemo na zasedenih mestih (#), dobimo seznam prostih strnjenih odsekov. Na vsak tak odsek lahko parkiramo avtomobile kar po vrsti od prvega proti zadnjemu mestu. Na odseku dolžine k bomo imeli tako $k-1$ enostavnih parkiranj. Od števila vseh avtomobilov n sedaj samo odštejemo skupno število enostavnih parkiranj.

Rezanje palice

Če prerežemo palico kjerkoli pred prvim podstavkom ali med prvim (x_0) in drugim (x_1) podstavkom, ne bo stabilna. Enako velja za drugi konec palice. Obravnavajmo sedaj primere, kjer naredimo rez nekje med sosednjima podstavkoma x_i in x_{i+1}. S tem določimo skrajne podstavke obeh delov. Razmislimo, kdaj bo levi del nestabilen. Do tega pride, če naredimo rez na takem mestu a, da bo težišče $a/2$ levo od x_0 ali pa desno od x_i. Podobno velja za desni konec palice. Vse lokacijo levo od $l_1 = 2x_0$ in desno od $r_1 = 2x_i$ so torej problematične. Na enak način dobimo tudi meji l_2 in r_2, ki sta posledica nestabilnosti desnega kosa. Združimo ju v $l = \max(l_1, l_2)$ in $r = \min(r_1, r_2)$. Poskrbeti moramo še, da sta l in r na intervalu $[x_i, x_{i+1}]$. Dolžine tako dobljenih problematičnih območij seštejemo po vseh parih sosednjih podstavkov i in $i+1$.

Efektivna obrestna mera

Iz podatkov o glavnici g in obrestni meri p moramo najprej izračunati mesečni obrok a. Če sledimo opisu odplačevanja, lahko zapišemo spodnjo enačbo. Iz nje z uporabo formule za vsoto geometrijske vrste izračunamo a.

$$\begin{aligned}(((g(1+p)-a)(1+p)-a)(1+p)-a)... &= 0\\ g(1+p)^3 - a(1+p)^2 - a(1+p) - a... &= 0\\ g(1+p)^m - a\sum_{i=1}^{m-1}(1+p)^i &= 0\end{aligned}$$

Sedaj lahko izračunamo efektivno obrestno mero (EOM), ki jo bomo poiskali z bisekcijo. Za nek q moramo sedaj preveriti, ali je premajhen ali prevelik. Večina členov v formuli za izračun EOM je enakih, kjer je denarni tok P enak mesečnemu obroku a, ki je povečan še za mesečne stroške. Učinkovito moramo torej izračunati spodnjo vsoto, kar lahko zopet storimo s pomočjo formule za vsoto geometrijske vrste.

$$\sum_i \frac{P}{((1+q)^{1/12})^i} = P\sum_i((1+q)^{-1/12})^i.$$

Omara

Najprej moramo prebrati opis omare, kar najlažje storimo z rekurzivno funkcijo, ki z vhoda prebere število razdelkov v omari (število vrstic m in število stolpcev n), nato pa po vrsti prebere opise teh razdelkov po istem principu (rekurzivno). Ob branju strukture razdelkov v omari lahko sproti računamo tudi velikost omare oz. razdelka iz velikosti podrazdelkov, ki smo jih prebrali. Za to moramo poiskati najširši (w) in najvišji (h) podrazdelek. Posamezni podrazdelki bodo morali biti torej velikosti $w \times h$, razdelek pa bo velikosti $nw \times mh$.

Ko imamo prebrane opise razdelkov v bolj prikladni strukturi in izračunane njihove velikosti, jih lahko prav tako z rekurzivnim postopkom izrišemo v naprej pripravljeno tabelo. Najprej izrišemo rob zunanjega razdelka, nato pa rekurzivno izrišemo vse podrazdelke na njihovih lokacijah, ki jih lahko izračunamo iz velikosti podrazdelkov. Paziti moramo, da pri tem slučajno ne prepišemo kakšnega stičišča '+' z robom '|' ali '-'.

Menjalnica kriptovalut

Opis podatkov predstavlja usmerjen graf na 6^3 vozliščih, kjer povezave predstavljajo množenje ali deljenje z nekim številom. Za vsako vozlišče bomo izračunali največjo vrednost poti, ki nas pripelje do njega. To bomo počeli iterativno tako, da v vsaki iteraciji upoštevamo povezave, ki izhajajo iz vozlišč, ki so se spremenila v prejšnji iteraciji (na začetku je to samo vozlišče, ki ustreza valuti EUR).

Težava nastane, da lahko s tem postopkom odkrijemo pozitiven cikel. Prej opisan postopek je podoben algoritmu Bellman-Ford za iskanje najkrajših poti. Tam lahko zaznamo negativen cikel s tem, da pride pri nekem vozlišču do spremembe po n iteracijah (ker so aciklične poti lahko dolge največ $n - 1$ povezav). Z enakim razmislekom lahko v naši simulaciji vsako spremembo po n iteracijah ali kasneje obravnavamo kot del pozitivnega cikla in nastavimo vrednost vozlišča na neskončno. Da se te neskončnosti propagirajo po preostanku grafa, pa naredimo kar $2n$ iteracij.

Zadnja težava je v največji možni vrednosti, ki je izven območja 64-bitnih števil. Lahko smo previdni in pred množenjem z deljenjem preverimo, ali bi bil produkt prevelik in ločeno obravnavamo taka "neskončna" števila. Lahko pa malo poguljufamo z uporabo GCC-jeve razširitve za 128-bitna števila (`__int218`).

Slovnica

Izberimo si termin `s`, ki bo ostal po krajšanju niza oz. preverimo vse možne s-je, ki se pojavijo v nizu. Podniza na obeh straneh se morata okrajšati v prazen niz. Naj nam funkcija $f(i, j)$ pove, ali je možno povsem skrajšati podniz $s[i : j]$. Ne vemo, kako bo potekalo krajšanje, vendar se bo nekje v tem procesu moral pokrajšati termin $s[i]$ z nekim drugim kompatibilnim terminom $s[k]$. Pri tem pa se morata v prazno okrajšati podniza $s[i : k]$ in $s[k+1 : j]$. Tako dobimo enostavno rešitev dinamičnega programiranja, ki se je lahko lotimo od krajših proti daljšim podnizom ali pa z memoizacijo. Rešiti moramo $O(n^2)$ podproblemov, za rešitev vsakega pa potrebujemo $O(n)$ časa zaradi preverjanja vseh možnih k-jev.