Univerzitetni programerski maraton 2024
1. kolo — rešitve nalog

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

22. februar 2024

6174

Naloga jasno opisuje postopek, ki ga je treba implementirati. Tudi učinkovitost implementacije ni pomembna, ker imamo zagotovilo, da se proces ustavi v 10 korakih. Implementacija vključuje pretvarjanje med nizi in števili ter urejanje. To bo lažje in hitreje izvesti npr. v Pythonu kot v C-ju.

Stiskanje

Odločiti se moramo, katere datoteke stisniti, da njihova skupna velikost ne bo presegla kapacitete C, pri tem pa jih želimo stisniti čim manj. Recimo, da je skupna velikost nestisnjenih datotek enaka x. Če stisnemo i-to datoteko, se bo skupna velikost spremenila na $x - d_i + s_i = x - (d_i - s_i)$. Da bomo velikost čim bolj zmanjšali, se nam torej splača izbrati datoteke z največjo razliko med stisnjeno in nestisnjeno velikostjo $d_i - s_i$. Izberemo jih toliko, da velikost ustreza kapaciteti ključka.

Križci in krožci

Stanja igre predstavimo z grafom, kjer vozlišče ustreza nekemu stanju na igralni plošči. Iz stanja je jasno, kateri igralec je na potezi. Povezave so usmerjene in predstavljajo možne poteze do novih stanj. Ker je na plošči vedno več potez, ne moremo dobiti cikla. Opravka imamo torej z usmerjenim acikličnim grafom z dokaj majhnim številom vozlišč in povezav, pri tem pa so nekatera vozlišča posebna, ker predstavljajo konec igre oz. zmagovalno pozicijo.

Na začetku lahko iz vozlišča, ki predstavlja prazno ploščo izračunamo vsa možna dosegljiva stanja. Tako lahko za vsako poizvedbo ugotovimo, ali je stanje sploh nedosegljivo ali morda zmagovalno. Sicer pa moramo izračunati število vozlišč, ki so dosegljiva iz vozlišča, ki ustreza podani plošči. To naredimo na enak način, kot smo na začetku generirali vsa dosegljiva stanja iz prazne plošče.

Bančna kartica

Nalogo lahko rešitev (vsaj) na dva načina. Prvi način je z dinamičnim programiranjem. Recimo, da smo zapolnili prvih $i - 1$ mest s števkami in nas zanima, na koliko načinov lahko dokončamo veljaven PIN. Pri tem je pomembno samo koliko (r) različnih števk smo uporabili do sedaj, ne pa tudi kakšne so in na katerih mestih. Na i-tem mestu lahko uporabimo katero od že uporabljenih števk na r načinov, ali pa izberemo kakšno novo na $N - r$ načinov: $f(i, r) = r \cdot f(i + 1, r) + (N - r) \cdot f(i + 1, r + 1)$. Uporabimo lahko memoizacijo ali pristop od spodaj navzgor.

Drugi način je, da poznate Stirlingova števila druge vrste, ki predstavljajo število načinov za delitev množice velikosti n v k nepraznih podmnožic. Za vsako število različnih uporabljenih števk $k \geq K$ v PINu lahko izračunate število veljavnih rešitev: P mest v PINu razbijete v k skupin, izberete nabor k števk izmed N razpoložljivih in jih razporedite med k skupin ($\sum_{k \geq K} S(P, k)C(N, k)k!$).

Topografska prominenca

Rezultate bomo računali od višjih proti nižjim celicam, zato jih najprej uredimo po višinah. Pri nižanju te meje se pojavljajo nove celice, obstoječe komponente pa se združujejo. V vsaki taki povezani komponenti imajo prominenco definirane vse celice razen tistih z največjo višino. Te bomo hranili za vsako komponento v svojem seznamu. Pri združitvi dveh komponent lahko izračunamo prominenco tistim najvišjim vrhovom, ki so nižji med obema komponentama, saj mora pot do višjih vrhov potekati skozi ravnokar dodano najnižjo celico. Vsakemu vrhu bomo prominenco določili največ enkrat. Če so najvišji vrhovi v obeh komponentah enako visoki, združimo seznama najvišji vrhov, ki še nimajo določene prominence. Združevanje seznamov izvedemo z dodajanjem elementov manjšega seznama v večjega, s čimer si zagotovimo največ $O(n \log n)$ vstavljanj, če je n število celic v mreži.

Čakanje

Pričakovano število znakov je odvisno od zaključka do sedaj zgeneriranega niza oz. od dolžine najdaljše predpone iskanega vzorca, ki se pojavi kot pripona na koncu zgeneriranega niza. Pri vsakem dodanem znaku se to stanje (dolžina) spremeni. Prehode med stanji ob dodatku novega znaka lahko učinkovito izračunamo z algoritmom KMP v linearnem času. Z $f(i, a)$ označimo novo stanje, če v stanju i dodamo znak a. Za izračun te tabele prehodov potrebujemo $O(nS)$ časa in prostora, kjer je n dolžina iskanega gesla, S pa velikost abecede.

Sedaj lahko sestavimo sistem enačb, ki opisujejo pričakovano število potez e_i, če smo v stanju i, kjer se na koncu zgeneriranega niza nahaja prvih i znakov vzorca (gesla). Na primer:

$$\begin{aligned} e_n &= 0 \\ e_i &= 1 + 1/S \cdot e_{i+1} + 1/S \cdot e_i + 1/S \cdot e_{i-4} + ... \\ e_0 &= 1 + 1/S \cdot e_1 + (1 - 1/S) \cdot e_0 \end{aligned}$$

Če ga malenkost preoblikujemo, dobimo trikoten sistem.

$$\begin{aligned} e_n &= 0 \\ e_{i+1} &= Se_i - S - e_i - e_{i-4} - ... \\ e_1 &= Se_0 - S - (S - 1) \cdot e_0 \end{aligned}$$

Enačba za e_1 je očitno oblike $Xe_0 + Y$. Lahko pa tudi vse ostale izrazimo v taki obliki, saj so enačbe za e_i odvisne samo od $e_{j<i}$. Na koncu iz enačbe $e_n = X'e_0 + Y' = 0$ izračunamo iskano vrednost e_0.