DESCRIEREA SOLUȚIILOR, OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, CLASA A IX-A

COMISIA ȘTIINȚIFICĂ

Pietricele

Propusă de: Stud. Mihnea-Vicențiu Bucă - Universitatea din Bucuresti, Facultatea de Matematica si Informatica

Cerința 1. Pentru rezolvarea acestei cerințe se cere să se afle subsecvența de lungime $n-k+1$ cu valoarea maximă. O soluție pentru a găsi răspunsul ar fi să se parcurgă șirul de la stânga la dreapta și să se calculeze valoarea primei subsecvențe de lungime $n-k+1$. Apoi, pe măsură ce înaintăm în șir vom actualiza valoarea curentă adunând valoarea noului element parcurs și scăzând ultima valoare a subsecventei curente, la final vom afișa maximul dintre toate valorile calculate

Cerința 2. Pentru rezolvarea acestei cerințe vom porni de la cea de-a patra restricție a problemei. Parcurgem fiecare subsecvență a șirului: $[i,j]$ cu $1 \leq i \leq j \leq n$, îi calculăm valoarea, pe care o vom nota cu *val* și presupunem că această subsecvență reprezintă raftul de valoare minimă. În continuare, vom "sparge" subsecvențele $[1,i-1]$ și $[j+1,n]$ astfel încât toate subsecvențele rezultate să aibă cel puțin valoare *val*, iar dacă se pot sparge în cel puțin k subsecvențe (inclusiv subsecvența $[i,j]$) însemnă că avem o posibilă soluție. Această soluție are o complexitate de $O(n^3)$.
Pentru a reduce complexitatea soluției de mai sus, vom face următarea observație: Dacă putem "sparge" un șir astfel încăt valoarea minimă să fie cel mult V_x, atunci există o modalitate de a "sparge" șirul folosindu-ne de aceleași subsecvențe astfel încât valoarea minimă să fie cel mult V_y cu $V_y > V_x$, astfel ne vine ideea de a căuta binar. Vom căuta binar soluția astfel: dacă putem împărți șirul în x subsecvențe, cu $x > k$, atunci valoarea maximă minimă este mai mică decâ valoare curentă, iar dacă putem împărții șirul în x subsecvențe, cu $x \leq k$, atunci valoarea maximă minimă este cel puțin valoarea curentă. Această soluție are o complexitate de $O(n\log(V_{MAX}))$ unde V_{MAX} reprezină valoarea întregului șir de pietricele.

Bolovani

Propusă de: Lect. univ. Csaba György Pătcaş - Universitatea Babes-Bolyai, Cluj-Napoca

Cerința 1 – $\mathbf{n \leq 10}$. Vom încerca planificarea celor n sarcini de spargerea bolovanilor în toate ordonările posibile și putem alege cea care maximizează cerința. Această soluție obține 20 de puncte.

Cerința 2 – $\mathbf{d_1 = d_2 = \cdots = d_n}$. În cazul în care toate valorile d_i sunt egale, observăm că putem planifica spargerile în ordinea crescătoare a valorilor z_i și astfel obținem mereu o soluție optimă. Această abordare obține 30 de puncte, iar combinând cele două idei, se poate ajunge la 40 de puncte.

Soluție oficială. Pentru 100 de puncte aplicăm următoarea strategie. Sortăm sarcinile în ordinea crescătoare a termenelor limită. Încercăm să adăugăm pe rând fiecare sarcină i la mulțimea sarcinilor, care se vor finaliza la timp. Notăm cu S_i mulțimea sarcinilor alese după prelucrarea bolovanului i.

Avem două cazuri:

(1) Dacă $z_i + \sum_{k \in S_{i-1}} z_k \leq d_i$ putem adăuga sarcina i la soluție, adică $S_i \to S_{i-1} \cup \{i\}$

(2) În caz contrar, căutăm sarcina $p \in S_{i-1}$ având z_p maxim. Dacă $z_p > z_i$, eliminăm sarcina p din soluție și adăugăm sarcina i, adică $S_i \to S_{i-1} \setminus \{p\} \cup \{i\}$

După parcurgerea tuturor sarcinilor, planificăm sarcinile din soluția obținută în mulțimea S_n în ordinea valorilor d_i, pe urmă planificăm celelalte sarcini în orice ordine.

O implementare naivă are complexitatea $O(n^2)$, care se poate îmbunătăți la $O(n\log(n))$ prin folosirea unor structuri de date avansate, cum ar fi heap-urile binare, dar o astfel de implementare nu este necesară pentru obținerea punctajului maxim.

Metoda prezentată este cunoscută în literatura de specialitate sub numele Algoritmul Moore-Hodgson(1) și a fost publicată în 1968. Algoritmul găsește o soluție, care nu doar maximizează numărul sarcinilor care se finalizează la timp, dar și minimizează suma lungimilor sarcinilor alese în S_n. Putem intui acest lucru din faptul, că în cazul 1 numărul sarcinilor alese crește cu 1, iar în cazul 2 numărul sarcinilor alese rămâne la fel, dar suma lungimilor lor scade. Există numeroase demonstrații riguroase pentru corectitudinea algoritmului, care folosesc metodele standard, cum ar fi reducerea la absurd sau inducția matematică.

Nisip

Propusă de: Stud. Vlad-Alexandru Gavrila-Ionescu - University of Cambridge
Stud. Theodor-Gabriel Tulbă-Lecu - Universitatea Politehnica din București
Prof. Liliana Șchiopu - Colegiul Național Frații Buzești, Craiova

Cerința 1

Putem simula secundă cu secundă starea minei. Astfel, când trecem de la o secundă la alta actualizăm starea minei în $O(n)$, iar la o întrebare putem răspunde în $O(1)$ prin afișarea valorii de pe poziția p la secunda t.

Cerința 2

Pentru a rezolva cerința a 2-a, în urma ștergerii unei pietre, va trebui să găsim secțiunea de nisip-aer care este afectată de această stergere.

În continuare este nevoie să găsim o formulă rapidă pentru a răspunde la o întrebare. Astfel, vom analiza cum evoluează în timp o secțiune prin care curge nisip:

#	#	#	#	#	.	.	.	.	.	.
#	#	#	#	.	#	.	.	.	.	.
#	#	#	.	#	.	#	.	.	.	.
#	#	.	#	.	#	.	#	.	.	.
P	.	#	.	#	.	#	.	#	.	.
.	.	.	#	.	#	.	#	.	#	.
.	.	.	.	#	.	#	.	#	.	#
.	.	.	.	.	#	.	#	.	#	#
.	.	.	.	.	.	#	.	#	#	#
.	.	.	.	.	.	.	#	#	#	#

Table 1. Evoluția curgerii unei secțiuni de 5 celule de nisip. Cu # am notat nisip, cu . aer, iar cu P piatra

Putem observa că pentru a verifica dacă într-o celulă se află nisip este suficient să ne uităm în colțurile stânga-sus și dreapta-jos și pe diagonalele principale de o anumită paritate. Putem deduce o formulă compusă bazată pe diagonalele principale și secundare din această matrice spațiu × timp.

Pentru a găsi rapid secțiunea de nisip-aer care ne interesează pentru o întrebare putem să ne precalculăm două lucruri:

(1) vom memora secțiunile de nisip într-o singură celulă ca o pereche de (pos, cnt) reprezentând poziția unde se află fundul secțiunii de nisip, repsectiv numărul de celule de nisip din acea secțiune
(2) 2 tablouri unidimensionale *next* și *prev*, care rețin următoarea, respectiv anterioara poziție a unei celule solide (care nu este aer).

Cerința 3

Pentru a rezolva cerința a 3-a, se va folosi soluția de la cerința a 2-a, dar nu este nevoie de prima precalculare, și se vor muta între ștergerile de pietre secțiunile de nisip la fundul secțiunii de nisip-aer. Această soluție are complexitate $O(cnt_{nisip} \cdot cnt_{\text{ștergeri}})$.

Soluția oficială

Pentru a obține punctajul maxim, vom porni de la soluția pentru cerința a 3-a. Vom observa că atunci când facem ștergeri ale unor pietre trebuie să actualizăm *next*-ul și *prev*-ul celulelor solide vecine pietrei șterse. Totodată unele celule de aer, nu vor mai arăta către o celulă solidă, ci vor trebui parcurse aceste liste de mai multe ori. Optimizarea necesară ce trebuie făcută

este că odată parcurs un astfel de lanț, toate celulele vizitate vor fi actualizate către celula cu adevărat solidă, pentru ca accesările ulterioare ale acestor celule să fie mult mai rapide.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Conf. univ. Doru Popescu Anastasiu - Universitatea din Pitesti, Departamentul de Matematica-Informatica
- Lect. univ. Csaba György Pătcaş - Universitatea Babes-Bolyai, Cluj-Napoca
- Prof. Gheorghe-Eugen Nodea - Centrul Județean de Excelență Gorj
- Prof. Liliana Șchiopu - Colegiul Național Frații Buzești, Craiova
- Prof. Petru Simion Opriță - Liceul "Regina Maria", Dorohoi
- Stud. Mihaela Cismaru - Universitatea din Craiova, NetRom Software
- Stud. Theodor-Gabriel Tulbă-Lecu - Universitatea Politehnica din București
- Stud. Ioan-Cristian Pop - Universitatea Politehnica din București
- Stud. Vlad-Alexandru Gavrila-Ionescu - University of Cambridge
- Stud. Bogdan Iordache - Universitatea din Bucuresti, Facultatea de Matematica si Informatica
- Stud. Denis Andrei Banu - Facultatea de Informatica, Universitatea "Alexandru Ioan Cuza", Iasi
- Stud. Mihnea-Vicențiu Bucă - Universitatea din Bucuresti, Facultatea de Matematica si Informatica

Referințe

[1] An n job, one machine sequencing algorithm for minimizing the number of late jobs, J. Michael Moore