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

Microbuz

Propusă de: Prof. Marinel Șerban

Soluție prof. Marinel Șerban. — CN E. Racoviță Iași

Observații. Problema testează cunoștințe despre:

(1) lucrul cu vectori
(2) utilizarea unui algoritm de tip succesor
(3) lucru cu baze de numerație
(4) determinarea minimului și maximului unei mulțimi de valori

Cerințele 1 și 2. Se vor genera toate combinațiile posibile de bilete în mod succesiv.

Pentru început, vom reține în tabloul $max\_cnt_i = \min(3, \frac{n}{i})$, pentru i de la 1 la 10, numărul maxim de bilete de tip i pe care le putem cumpăra fără a depăși n kilometri.

În continuare, vom genera fiecare combinație cu ajutorul tabloului cnt_i = numărul de bilete de tip i din combinație, în mod succesiv utilizând un *algoritm de tip succesor*. Pentru fiecare combinație vom calcula distanța parcursă de combinația:

$$distanță = 1 \cdot cnt_1 + 2 \cdot cnt_2 + \cdots + 10 \cdot cnt_10$$

și suma costurilor biletelor din acea combinație

$$cost = km_1 \cdot cnt_1 + km_2 \cdot cnt_2 + \cdots + km_10 \cdot cnt_10$$

Dacă distanța este egală cu n, atunci se actualizează răspunsul optim.

Pseudocodul arată astfel:

pentru i de la 1 la 10 **execută**
    $max\_cnt_i \leftarrow \min(3, n/i)$
sfârșit pentru
$cost\_minim \leftarrow \infty$; $ok \leftarrow 1$; $cost \leftarrow 0$; $distanță \leftarrow 0$
cât timp $ok = 1$ **execută**
    $i \leftarrow 10$; $cnt_i \leftarrow cnt_i + 1$; $distanță \leftarrow distanță + i$; $cost \leftarrow cost + km_i$
    cât timp $i > 0$ și $cnt_i > max\_cnt_i$ **execută**
        $sumă \leftarrow sumă - cnt_i \cdot km_i$; $cost \leftarrow cost - cnt_i \cdot i$; $cnt_i \leftarrow 0$; $i \leftarrow i - 1$
    sfârșit cât timp
    dacă $i = 0$ **atunci**
        $ok = 0$
        sfârșit
    sfârșit dacă
    dacă $distanță = n$ **atunci**
        dacă $cost\_minim > cost$ **atunci**
            $cost\_minim \leftarrow cost$
            pentru i de la 1 la 10 **execută**
                $răsp_i \leftarrow cnt_i$
            sfârșit pentru
        sfârșit dacă
    sfârșit dacă
sfârșit cât timp

Existența soluției pentru cerința 3. În condițiile problemei:
Suma minimă a unei submulțimi este 10 – mulțime cu un bilet de cost 10.
Suma maximă a unei submulțimi este 855 – mulțimea de bilete $\{91, 92, 93, 94, 95, 96, 97, 98, 99\}$.
Deci există $855–10 + 1 = 846$ sume posibile.
În același timp există $C_{10}^1 + C_{10}^2 + \cdots + C_{10}^9 = 1022$ submulțimi.
Conform principiului cutiei lui Dirichlet există cel puțin două submulțimi de sumă egală.

Cerința 3. Se vor genera toate combinațiile posibile de submulțimi de două seturi bilete disjuncte în mod consecutiv.

Se va proceda similar ca la primele două cerințe, dar acum vom encoda starea unui bilet cu una din cele 3 valori:

0 – biletul nu face parte din nici un set
1 – biletul face parte din setul 1
2 – biletul face parte din setul 2

Pentru fiecare combinație se determină cele 2 seturi de bilete. Se reține maximul și combinația care produce acest maxim.

Soluție prof. Claudiu-Cristian Gorea-Zamfir. – Școala Gimnazială Alexandru cel Bun, Iași

Cerințele 1 și 2.

- Se generează toate numerele de 10 cifre, în baza 4 și se rențin pentru un număr cifrele sale într-un tablou *cif*. Acest tablou va reține numărul de bilete cumpărate pentru fiecare tip de bilet *i* de la 1 la 10;
- Se calculează prețul astfel:

$$preț = cif_1 \cdot p_1 + cif_2 \cdot p_2 + \cdots + cif_{10} \cdot p_{10}$$

- Se calculează distanța astfel:

$$dist = cif_1 \cdot 1 + cif_2 \cdot 2 + \cdots + cif_{10} \cdot 10$$

- Dacă $dist = n$ și $preț < preț\_min$ atunci reținem configurația actuală a tabloului *cif* și actualizăm $preț\_min = preț$;

Cerința 3.

- Fiecărui bilet îi asociem 3 stări:
  0 – biletul nu este ales
  1 – biletul este ales în mulțimea 1
  2 – biletul este ales în mulțimea 2
- Se generează toate numerele de 10 cifre în baza 3 în mod similar cu cerințele anterioare;
- În funcție de mulțimea asociată, vom calcula sumele sum_1 și sum_2, iar dacă acestea sunt egale și maxime, vom reține suma maximă și configurația.

Soluție prof. Daniela-Elena Lica. -– Centrul Județean de Excelență Prahova)

Cerințele 1 și 2. : Problema acceptă și o abordare de tip programare dinamică, asemănătoare cu cea folosită pentru rezolvarea Problemei Rucsacului.

Astfel, considerăm tabloul unidimensional:

$cost_i$ = costul minim necesar pentru a călători i kilometri

Fie:

$bilet_i$ = prețul unui bilet ce asigură parcurgerea unei distanțe de i km

Construirea tabloului cost. Să presupunem că la pasul curent considerăm achiziționarea unui bilet ce asigură parcurgerea unei distanțe de d km, cu prețul egal cu $bilet_d$;

Obținem astfel următoarea relație de recurență:

$$cost_{v+d} = \min(cost_{v+d}, cost_v + bilet_d)$$

Pentru orice valoare v, parcursă descrescător (pentru a evita suprascrieri incorecte) de la $(165 - d)$ până la 0.

De remarcat este faptul că fiecare tip de bilet (dintre cele 10) poate fi cumpărat de cel mult trei ori, astfel că se va efectua adăugarea distanței de d km în *rucsac* de exact trei ori.

Pentru prima cerință este suficient să calculăm doar tabloul unidimensional *cost*, cu o complexitate de timp egală cu: $\mathcal{O}(n \cdot S_{max})$, unde $n = 10$ și $S_{max} = 165$ pentru problema noastră.

Pentru a rezolva și cea de-a doua cerință, pe lângă acest tablou, putem reține, pentru fiecare valoare v de la 0 la 165, și modul în care $cost_v$ a fost obținut. Astfel, în cadrul egalității, dacă:

$$cost_v + bilet_d < cost_{v+d}$$

în adiție cu actualizarea valorii lui $cost_{v+d}$, putem considera că reconstrucția necesară pentru $v+d$ km este identică cu cea necesară pentru v km, la care mai trebuie să adăugăm achiziționarea unui bilet de d km. Aceste reconstrucții pot fi efectuate, de exemplu, cu ajutorul unor `struct`-uri, în $\mathcal{O}(n \cdot S_{max})$ memorie – pentru fiecare posibilitate de distanță, pot exista cel mult 30 de tichete utilizate în atingerea acesteia.

Cerința 3. Folosind scrierea în baza 2 a numerelor naturale de la 0 la $2^e - 1$, putem genera toate submulțimile unei mulțimi ce are e elemente. Astfel, putem genera toate submulțimile mulțimii cu 10 elemente din problema de față:

$$\{bilet_1, bilet_2, bilet_3, \ldots, bilet_9, bilet_{10}\}$$

De exemplu, să considerăm, în contextul acestei scrieri, numărul binar 0000010111:

- bitul cel mai semnificativ se află în partea stângă;
- Aici, cel mai semnificativ bit are valoarea 0, este *nesetat*;
- Au fost selectate numerele:

  $$bilet_1, bilet_2, bilet_3 \text{ și } bilet_5$$

- Fie $S = bilet_1 + bilet_2 + bilet_3 + bilet_5$;
- La acest pas, ne-ar mai interesa să vedem dacă există o modalitate de a alege niște numere din mulțimea rămasă, adică:

  $$\{bilet_4, bilet_6, bilet_7, bilet_8, bilet_9, bilet_{10}\}$$

  astfel încât însumate să dea S. Pentru aceasta, folosim un algoritm identic cu cel folosit în cadrul cerinței 2.

Soluție stud. Bogdan-Ioan Popa. – Universitatea București

În cele ce urmează vom reprezenta o submulțime de bilete ca fiind un număr în baza 2 scris cu K biți. Dacă în configurația (submulțimea) *conf* bitul i este 1 înseamnă că biletul cu $i + 1$ km este luat în submulțimea *conf*.

Cerințele 1 și 2. Se calculează vectorul de perechi D_i = perechea (*cost*, *conf*) unde *cost* reprezintă costul minim să obținem un drum de distanță exact i, *conf* reprezintă submulțimea de bilete cu care se obține acest cost minim. Ce rămâne să găsim acum este un triplet:

$$dist_1, dist_2, dist_3 \text{ pentru care,}$$

$$dist_1 + dist_2 + dist_3 = N \text{ și}$$

$$D_{dist_1.cost} + D_{dist_2.cost} + D_{dist_3.cost} \text{ este minim.}$$

Pentru reconstrucția soluției ne vom folosi de biletele cumpărate în configurațiile $D_{dist_1.conf}$, $D_{dist_2.conf}$ respectiv $D_{dist_3.conf}$.

Complexitatea aceste soluții este $O(2K)$, unde K este numărul de tipuri de bilete posibile pe care le putem cumpăra. Pentru această problemă $K = 10$.

Cerința 3. Se calculează tabloul bidimensional:

$$A_{conf,sum\_cost} = \begin{cases} 1, & \text{dacă există o submulțime } x \text{ a submulțimii } conf \text{ astfel încât suma} \\ & \text{costurilor biletelor din submulțimea } x \text{ să fie } sum\_cost \\ 0, & \text{altfel} \end{cases}$$

Ce rămâne de făcut este să iterăm prin toate submulțimile $conf$ posibile ale celor K bilete.

Fie *comp_conf* submulțimea complementară submulțimii *conf*.

Fie s suma costurilor biletelor din submulțimea *conf*.

Dacă $A_{comp\_conf,s} = 1$ înseamnă că am găsit două submulțimi disjuncte de bilete care însumează același cost. Nu uităm însă că $A_{comp\_conf,s} = 1$ indică faptul că există o submulțime a lui *comp_conf* pentru care suma costurilor biletelor din ea să fie s și nu înseamnă neapărat că suma costurilor biletelor din *comp_conf* este egală cu s.

Complexitatea soluției este $\mathcal{O}(2^K * S_{max})$, unde S_{max} reprezintă suma totală a costurilor biletelor.

Raza

Propusă de: prof. Alin Burța

Soluție prof. Alin Burța. – Colegiul National ,,B. P Hasdeu", Buzău

Cerința 1. Încă de la citirea datelor de intrare, se verifică dacă traiectoria roverului curent se intersectează cu diagonala principală și contorizăm numărul roverelor care îndeplinesc condiția.

Avem 3 cazuri:

- Roverul curent nu intersectează diagonala, caz care nu ne interesează;
- Roverul curent intersectează diagonala într-o singură pozitie;
- Roverul curent intersectează diagonala în două poziții.

Fie (x, y) linia respectiv coloana colțului din stânga-sus al traiectoriei roverului curent, iar r lungimea laturii traiectoriei. Condiția de intersecție a traiectoriei cu diagonala este:

$$x \leq y + r - 1 \text{ și } x + r - 1 \geq y$$

Cerința 2. Concomitent cu determinarea roverelor, care intersectează diagonala principală vom construi 3 vectori, cu următoarele semnificații:

- unu_i = secunda în care roverul i intersectează prima dată diagonala în cadrul primei parcurgeri a traiectoriei;
- doi_i = secunda în care roverul i intersectează a doua oară diagonala în cadrul primei parcurgeri a traiectoriei sau 0, dacă roverul NU intersectează diagonala în două poziții;
- $prd_i = 4 \cdot r - 4$, perioada roverului i (numărul de secunde necesare parcurgerii întregii traiectorii;

Numărul de secunde S este relativ mic, deci putem calcula, pentru fiecare secundă între 1 și S, câte rovere se găsesc pe diagonala principală în acel moment. Acest calcul presupune marcarea, pentru fiecare rover, a tuturor momentelor de timp în care roverul atinge diagonala principală, folosind cei trei vectori construiți anterior. Reținem la fiecare pas numărul maxim curent și momentul de timp corespunzător.

Soluție prof. Gorea-Zamfir Claudiu-Cristian. – Școala Gimnazială Alexandru cel Bun, Iași

La citirea roverelor, identificăm 5 situații bazate pe valorile de plecare, raza atinge traiectoria astfel:

- 2 colțuri ($l = c$) – momentele de timp sunt $\{0, 2r, 4r, 6r, \dots\}$;
- 1 colț, dreapta-sus ($c + r = l$) – momentele de timp sunt $\{1r, 5r, 9r, \dots\}$;
- 1 colț, dreapta-sus ($l + r = c$) – momentele de timp sunt $\{3r, 7r, 11r, \dots\}$;
- 2 laturi nord și est ($c < l$ si $l < c + r$) – momentele de timp sunt $\{1 + l - c, 1 + l - c + 4r, 1 + l - c + 8r, \dots\}$ și $\{2r + 1 - l + c, 6r + 1 - l + c, 10r + 1 - l + c, \dots\}$;
- 2 laturi sud și vest ($l < c$ si $c < l + r$) – momentele de timp sunt $\{1 + c - l + 2r, 1 + c - l + 6r, 1 + c - l + 10r, \dots\}$ și $\{1 - c + l + 4r, 1 - c + l + 8r, 1 - c + l + 12r, \dots\}$.

În fiecare situație contorizăm faptul ca raza intersectează traiectoria acestui rover. Numărul de rovere intersectate, fiind răspunsul cerinței 1.

Pentru cerința 2, în fiecare dintre cele 5 situații, pe baza relațiilor de tip progresie aritmetică obținute din periodicitatea întâlnirii roverului cu raza, vom folosi un vector de frecvență pentru a marca la fiecare moment care rovere ar fi afectate. La final determinăm maximul din acest vector și momentul în care raza va emite.

Text

Propusă de: prof. Vlad-Laurențiu Nicu

Soluție prof. Daniela-Elena Lica. – Centrul Județean de Excelență Prahova

Cerința 1. Presupunând că șirul de caractere citit are o lungime de L caractere după eliminarea spațiilor, se poate determina numărul de linii, respectiv de coloane, într-o complexitate egală cu $\mathcal{O}(\sqrt{L})$, prin parcurgerea tuturor divizorilor lui L.

Cerința 2. Pentru ușurință, se poate implementa o funcție ce primește ca parametru un șir de caractere să zicem, de lungime v, și determină într-o complexitate pătratică $\mathcal{O}(v^2)$ care, în practică, se comportă foarte bine, care este cea mai lungă secvență palindromică a acestuia.

Pentru determinarea unei secvențe maximale palindromice, se poate alege care să fie caracterul din mijloc pentru palindroame de lungime impară, respectiv care să fie cele două caractere identice din mijloc pentru palindroame de lungime pară, și apoi se încearcă extinderea simultană a capetelor palindromului, spre stânga cât și spre dreapta.

Astfel, putem apela această funcție de $N + M$ ori, unde N = numărul de linii ale matricei obținute, M = numărul de coloane ale matricei obținute, transmițând, de fiecare dată, ca parametru al funcției, șirul obținut prin parcurgerea unei linii, respectiv a unei coloane.

Cerința 3. Problema se poate rezolva într-o complexitate de timp $\mathcal{O}(N \cdot M) = \mathcal{O}(L)$.

Astfel, încercăm să găsim răspunsul parcurgând linie cu linie matricea obținută.

Presupunem că, la pasul curent, ne aflăm pe linia i. Introducem notația:

H_j = lungimea maximă a unei subsecvențe de vocale ce se termină în celula (i, j). Astfel, H_j poate lua doar valori naturale, cuprinse între 1 și M.

În mod particular, dacă celula (i, j) conține o consoană, atunci $H_j = 0$.

Încercăm, pe rând, să fixăm fiecare coloană j ca fiind coloana cu *înălțimea* minimă dintr-un subtablou dreptunghiular ce se termină pe linia i.

Definim:

$left_j$ = cea mai din stânga poziție pentru care $H_{left_j+1} \geq H_j, H_{left_j+2} \geq H_j, \ldots, H_{j-1} \geq H_j$

$right_j$ = cea mai din dreapta poziție pentru care $H_{j+1} \geq H_j, \ldots, H_{right_j-2} \geq H_j, H_{right_j-1} \geq H_j$

Putem observa că pentru orice j subtabloul dreptunghiular în cauză are: $l = H_j$ linii și $c = right_j - 1 - (left_j + 1) + 1$ coloane, deci o arie egală cu: $l \cdot c$ elemente.

Tablourile unidimensionale *left* și *right* pot fi calculate în complexitate de timp $\mathcal{O}(M)$, cu ajutorul a două parcurgeri, una de la stânga la dreapta și una de la dreapta la stânga. Pentru a afla răspunsul, respectiv aria maximă a unui dreptunghi ce conține doar vocale, se calculează maximul tuturor numerelor $l \cdot c$ descrise anterior.

Soluție prof. Vlad-Laurențiu Nicu. – Liceul Teoretic Mihail Kogălniceanu, Vaslui

Cerința 1. Se citește textul și se elimină spațiile formându-se o matrice de caractere de dimensiune divizori ai lungimii și e afișează matricea obținută.

Cerința 2. Pentru determinarea palindromului de lungime maximă se poate folosi algoritmul lui Manacher(1).

Acest algoritm constă în modificarea șirului inițial prin inserarea unui caracter oarecare (care sigur nu apare în sir exemplu caracterul $*$) între oricare două caractere. Nu e nevoie de transformarea efectivă a șirului, se poate lucra cu indicii șirului astfel încât să considerăm ca acest caracter a fost inserat. De exemplu șirul abccba devine $*$a$*$b$*$c$*$c$*$b$*$c$*$.

Pentru a găsi cea mai lungă secvență palindrom va trebui să extindem în jurul fiecărui caracter t_i, astfel încât secvența dintre t_{i-d} și t_{i+d} să fie palindrom.

Vom păstra în vectorul L_i – cel mai lung palindrom cu centrul în i. Valoarea maximă din vectorul L va reprezenta lungimea celui mai lung palindrom.

Cerința 3. Pentru determinarea dreptunghiului de arie maximă folosim un algoritm cu histograme (o funcție in trepte).

Se fixează două linii i_1 și i_2. Pentru fiecare linie se determină numărul minim de zerouri spre dreapta pe liniile $i_1, i_1 + 1, \ldots, i_2$.

Se determină astfel aria maximă.

Soluție stud. Bogdan-Ioan Popa, stud. Tulbă-Lecu Theodor-Gabriel. – Universitatea din București, Universitatea Politehnica București

Cerința 3. Se iterează prin fiecare pereche de linii $(i_1, i_2), i_1 \leq i_2$. Pentru o pereche (i_1, i_2) fixată vom dori să calculăm care este dreptunghiul de arie maximă ce conține doar vocale care are linia de sus pe linia i_1 și linia de jos pe linia i_2. Pentru a reuși acest lucru vom considera vectorul $C_j = 1$, dacă pe coloana j între liniile i_1 și i_2 sunt doar vocale sau 0 dacă nu. Calcularea acestui vector se poate face cu sume parțiale sau folosindu-ne de vectorul C calculat anterior pentru perechea $(i_1, i_2{-}1)$.

Ce rămâne de făcut este să găsim cea mai lungă secvență de valori consecutive, fie ea de lungime K. Atunci dreptunghiul de arie maximă care conține doar vocale cu linia sus pe i_1 și linia de jos pe i_2 are aria egală cu $K * (i_2{-}i_1 + 1)$. Complexitatea timp a acestei soluții este $\mathcal{O}(M \cdot N^2) = \mathcal{O}(L \cdot \sqrt{L})$.

Soluție prof. Șerban Marinel. – Colegiul Național Emil Racoviță, Iași

Cerința 1. Citirea se face caracter cu caracter, reținându-se într-un vector caracterele diferite de spațiu. Evident, concomitent se determină numărul de caractere, apoi se calculează numărul de linii L și numărul de coloane C al matricei pentru cerința 1.

Cerința 2. Pentru a determina palindromul maxim se parcurge matricea linie cu linie, căutându-se palindroame de lungime *lung* din mulțimea $C, C-1, C-2, \ldots$.

Dacă *lung* devine mai mic decât lungimea palindromului detectat până în acel moment se trece la linia următoare. Dacă se determină un palindrom de lungime egală cu lungimea maximă detectată până în acel moment, se reține palindromul cel mai mare lexicografic. Același procedeu se repetă apoi coloană cu coloană, plecând de acestă dată de la *lung* = L, apoi $L-1, \ldots$

Cerința 3. Pentru acestă cerință construim o matrice în care am înlocuit vocalele cu 1 și consoanele cu 0 - acestă matrice poate fi calculată încă de la cerința 1. Pornind de la acestă matrice construim o *matrice de sume parțiale*. În acestă matrice am căutat dreptunghiuri de sumă maximă. Pentru datele de test foarte mari acestă metodă depășește timpul de execuție dat.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Cristina-Elena Anton
- Prof. Alin Burța
- Prof. Claudiu-Cristian Gorea-Zamfir
- Asist. Drd. Alexandru Ioniță
- Prof. Daniela-Elena Lica
- Prof. Vlad-Laurențiu Nicu
- Stud. Bogdan-Ioan Popa
- Prof. Marinel Șerban
- Stud. Theodor-Gabriel Tulbă-Lecu
- Prof. Florentina Ungureanu

Referințe

[1] articol – Algoritmul lui Manacher