Descrierea Soluțiilor
Olimpiada Societății pentru Excelență și Performanță în Informatică
Etapa barajelor pentru selecția echipelor naționale
Baraj 1 Juniori

1 Problema Ross

Propunători: Pop Ioan-Cristian, Universitatea Politehnica din București
Tulbă-Lecu Theodor-Gabriel, Universitatea Politehnica din București

Observații

- Se observă că, dacă selectam jumatate din puncte (din 2 în 2), putem colora toate segmentele.
- Este ineficient să selectăm două noduri adiacente.
- Vom defini cantitatea de vopsea necesara pentru a picta un segment drept costul acestui segment.
- Costul maxim posibil al colorarii segmenteleor adiacente unei submulțimi de x puncte este strict mai mic decat costul unei submulțimi de $x + 1$ puncte pentru orice $x \leq \frac{N}{2}$ și $x = x + 1$ pentru $x \geq \frac{N}{2}$.

Subtask-ul 1 ($N \leq 15$, $K \leq 10$) – 17 puncte

Pentru fiecare valoare $V[i]$, cu $1 \leq i \leq k$, generăm toate submulțimile de puncte și căutăm numărul minim de puncte, notat cu Nr astfel încât costul maxim posibil al colorării segmentelor adiacente celor Nr puncte este strict mai mare decat $V[i]$. Astfel, răspunsul va fi $Nr - 1$.

Complexitatea temporală acestei soluții este $\mathcal{O}(K \cdot 2^N)$.

Subtask-ul 2 ($N \leq 20$, $K \leq 100\ 000$) – 13 puncte

Acest subtask se rezolvă similar cu subtask-ul 1, însă plecăm de la observația că, indiferent de valorile lui $V[i]$, costul maxim posibil asociat alegerii unui anumit numar de puncte este constant. Fie $c[i]$ costul maxim pentru i puncte. Deci, $c[i] \leq c[i + 1]$.

Așadar, putem genera toate submulțimile de puncte, calculăm costul maxim posibil pentru fiecare număr de puncte (vectorul c) și căutăm binar soluția pentru fiecare valoare din vectorul V.

Complexitatea temporală acestei soluții este $\mathcal{O}(K \cdot log(N) + 2^N)$.

Subtask-ul 3 ($N \leq 100$, $K \leq 100$) – 11 puncte

Având în vedere că toate valorile sunt mai mici decât 100, putem rezolva problema folosind programare dinamică. Vom defini următorul tablou tridimensional cu valori din mulțimea $\{0, 1\}$: $d[i][j][k]$: are valoarea 1 dacă, din submulțimea primelor i puncte, se poate obține costul j selectând k puncte, respectiv 0 în caz contrar. Această abordare este similară cu soluția problemei rucsacului.

Presupunem că suntem la pasul i. Vom nota D_1 distanța dintre punctele i și $i-1$, respectiv D_2 distanța dintre punctele i și $i+1$.

Observăm ca dacă vom selecta punctul de pe poziția i, atunci nu vom selecta punctul de pe poziția $i-1$, deci starea i va depinde de starea $i-2$. Daca nu selectăm punctul de pe poziția i, atunci nu se modifică soluția cu nimic față de pasul anterior, deci starea i, va depinde de starea $i-1$.

Obținem următoarea relație de recurență:

Dacă $d[i-2][j-D_1-D_2][k-1]$ este egal cu 1, atunci și $d[i][j][k]$ va fi egal cu 1. Dacă $d[i-1][j][k]$ este egal cu 1, atunci și $d[i][j][k]$ va fi egal cu 1.

Dupa ce parcurgem toate cele n puncte, putem afla costul maxim posibil pe care îl putem obține dacă selectăm k puncte din primele i, ultimul punct selectat fiind cel cu indexul j.

Pentru a calcula vectorul c, ne interesează doar valorile pentru care $d[i][j][k]$ are valoarea 1: $c[k] = max(j),\ \forall d[i][j][k] = 1$

Astfel, complexitatea temporală a acestei soluții este $\mathcal{O}(N^2 \cdot K)$

Subtask-ul 4 ($N \leq 200$) – 24 de puncte

În continuare vom modifica dinamica astfel încât să reținem costul în valoarea dinamicii, în loc de a îl reține în starea acesteia.

Construim dinamica: $d[i][j][k] =$ costul maxim posibil pentru a colora segmentele adiacente ale oricăror k puncte dintre primele i puncte, cu ultimul punct selectat j.

Presupunem că suntem la pasul i. Fie D_1 distanța dintre punctele i și $i-1$, respectiv D_2 distanța dintre punctele i și $i+1$

Avem două cazuri:

- dacă $i \neq j$, atunci nu am selectat punctul i, deci nu se va schimba costul față de starea $i-1$.
- dacă $i = j$, atunci am selectat punctul i, deci se va schimba costul, și astfel dinamica va depinde de starea $i-2$.

Ajungem astfel la următoarele relații de recurență:

- $d[i][j][k] = d[i-1][j][k]$;
- $d[i][i][k] = max(d[i-2][1][k-1], d[i-2][2][k-1], \ldots, d[i-2][i-2][k-1]) + D_1 + D_2$;

La final, obținem costul maxim posibil de colorare a segmentelor adiacente unui anumit numar de puncte și căutăm binar soluția pentru fiecare valoare din vectorul V.

Complexitatea temporală este $\mathcal{O}(N^3)$

Subtask-ul 5 ($N \leq 2000$) – 35 de puncte

Observăm în continuare că ultimul parametru al dinamicii nu este necesar.

Construim dinamica: $d[i][j] =$ costul maxim posibil pentru a colora segmentele adiacente oricărei submulțimi de j puncte dintre primele i puncte.

Presupunem că suntem la pasul i. Fie D_1 distanța dintre punctele i și $i-1$, respectiv D_2 distanța dintre punctele i si $i+1$

În acestă formă avem două cazuri, am selectat sau nu punctul i. Similar cu subtaskul anterior, obținem următoarea relație de recurență: $d[i][j] = max(d[i-1][j], d[i-2][j-1] + D_1 + D_2)$

La final, obținem costul maxim posibil de colorare a segmentelor adiacente unui anumit numar de puncte și căutăm binar soluția pentru fiecare valoare din vectorul V.

Complexitatea temporală este $\mathcal{O}(N^2)$

2 Problema Rodiezv

Propunător: prof. Nodea Gheorghe-Eugen, Colegiul Național "Tudor Vladimirescu", Târgu-Jiu

Subtask-ul 1 ($N \leq 300$, $M \leq 1000$, doar vocale) – 26 de puncte

Pentru acest subtask, prin parcurgerea matricei linie cu linie, determinăm coordonatele (x_1, y_1) ale caracterului # aflat în vârful rombului (primul colț). Plecând de la acesta, se simulează parcurgerea tuturor elementelor unui posibil romb valid.

Evident, vom observa că următoarele două caractere # ce alcătuiesc un romb valid se vor afla pe aceași linie, simetrice unul de altul raportat la coordonata y_1. Vom nota cele două colțuri cu (x_2, y_2), respectiv (x_2, y_3). Fiind stabilite cele două colțuri, cu ușurință se poate determina latura rombului $lat = x_2 - x_1 + 1$. Coordonatele (x_3, y_1) ale celui de-al patrulea colț se deduc ușor ($x_3 = x_1 + 2 \cdot (lat - 1)$).

Permanent vom verifica dacă există # în interiorul rombului determinat de cele 4 colțuri.

Complexitatea acestei soluții este $\mathcal{O}(N^3)$.

Subtask-ul 2 ($N \leq 300$, $M \leq 10000$) – 33 de puncte

Acest subtask se rezolvă similar cu subtaskul 1, dar suplimentar trebuie verificat dacă avem pe conturul rombului cel puțin o vocală pe fiecare latură.

Complexitatea acestei soluții este $\mathcal{O}(N^3)$.

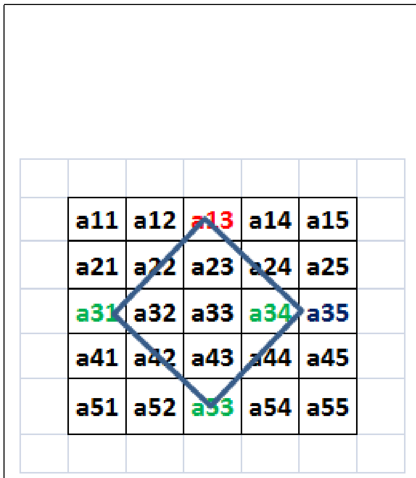
Subtask-ul 3 ($N \leq 2000$, $M \leq 50000$) – 41 de puncte

Varianta 1

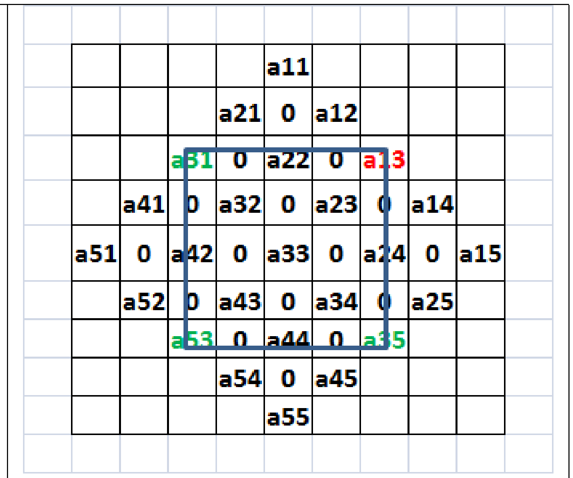
Vom construi un vector D care reține coordonatele (x, y) rotite cu 45° ale tuturor caracterelor # din matrice (M - numărul de #-uri). Vectorul obținut îl vom sorta crescător după x, iar la x-uri egale crescător după y.

De ce este necesară rotirea? Prin rotire rombul se transformă în pătrat, astfel coordonatele colțurilor pătratului fiind mai ușor de manipulat.

Pentru coordonatele inițiale (x, y) obținem coordonatele rotite: $(x + y - 1, N - x + y)$.



Romb cu latură 3



Rombul rotit cu 45°

Prin sortarea vectorului D, două #-uri aflate pe aceeași linie vor deveni candidate pentru a determina latura unui romb valid ($D[i].x = D[i+1].x$).

Să analizăm pentru exemplul de mai sus ce am obținut prin rotația cu 45° și sortarea vectorului D ($N = 5$):

- a_{31}, a_{13}, a_{53}, a_{35} - referințele din matricea inițilă:

  $D[i] = \{3, 3\} \Leftrightarrow \{3, 1\}$

  $D[i+1] = \{3, 7\} \Leftrightarrow \{1, 3\}$

  $\dots$

  $D[j] = \{7, 3\} \Leftrightarrow \{5, 3\}$

  $D[j+1] = \{7, 7\} \Leftrightarrow \{3, 5\}$

- putem spune că am determinat 2 colțuri ale rombului dacă $D[i].x = D[i+1].x$
- latura rombului $lat = \dfrac{D[i].x - D[i+1].x}{2} + 1$
- căutăm următoarea pereche pentru care:
  $D[j].x = D[j+1].x = lat$ și $D[i].y = D[j].y, D[i+1].y = D[j+1].y$.
- orice alt # întâlnit între indicii i și j care nu respectă condițiile $D[i].c \le D[j].c$ și $D[i+1].c \le D[j+1].c$ face imposibilă formarea unui romb valid.

Totodată, vom precalcula în două tablouri bidimensionale câte vocale se află pe fiecare subdiagonală paralelă cu diagonala principală a matricei, respectiv câte vocale se află pe fiecare subdiagonală paralelă cu diagonala secundară a matricei, Astfel, cunoscând coordonatele colțurilor putem afla în $\mathcal{O}(1)$ dacă există vocale pe conturul romburilor.

Observație Să remarcăm corelația între numărul total de romburi valide și latura unui romb. Astfel, un romb valid cu latură mare implică un număr mai mic de diezuri ce pot fi distribuite pentru a forma alte romburi.

Complexitatea acestei soluții este $\mathcal{O}(M \cdot log(M))$.

Varianta 2 – prof. Ionel-Vasile Piț-Rada

În timpul citirii datelor se construiesc următoarele structuri de date:

- matricea a pentru caracterele citite;
- matricea voc definită prin $voc[i][j]$ = numărul de vocale întâlnite dinspre Nord-Est;
- matricea $diez$ definită prin $diez[i][j]$ = distanța până la cel mai apropiat # dinspre Nord-Est.

Atunci când întâlnim un caracter # se verifică dacă există un romb cu vârful bazei de jos în (i, j) și se procedează astfel:

- se calculează distanța până la cel mai apropiat # dinspre Nord-Est $lat = diez[i-1][j+1]+1$
- pozițiile unde se pot afla celelalte 3 varfuri ale rombului sunt:
  - (sus) $(i - 2 \cdot lat, j)$
  - (dreapta) $(i - lat, j + lat)$
  - (stanga) $(i - lat, j - lat)$
- verificare:
  1. lungimea laturii este ≥ 3 și celelalte 3 vârfuri sunt la poziții corecte, $i - 2 \cdot lat > 0$ și $j - lat > 0$ și $j + lat \leq n$
  2. la acele poziții există #
  3. pe latura formată de vârfurile $(i - lat, j - lat)$ și $(i - 2 \cdot lat, j)$ nu există # în interiorul laturii, $diez[i - lat - 1][j - lat + 1] + 1 = lat$
  4. pe latura $(i, j) - (i - lat, j + lat)$ deja am verificat că nu există # in interior prin modul cum am calculat lat
  5. pe latura $(i, j) - (i - lat, j + lat)$ există vocale, $voc[i-1][j+1] - voc[i-lat][j+lat] > 0$
  6. pe latura $(i - lat, j - lat) - (i - 2 \cdot lat, j)$ există vocale, $voc[i - lat - 1][j - lat + 1] - voc[i - 2 \cdot lat][j] > 0$
  7. au mai rămas de verificat interiorul rombului și celelalte două laturi care se pot realiza în $\mathcal{O}(lat)$, aceasta nu va afecta complexitatea finală deoarece fiecare poziție este verificată cel mult de două ori în plus, față de parcurgerea pentru citire:
     - se parcurg pozițiile din interiorul laturii $(i, j) - (i - lat, j - lat)$ și dacă găsim # ne oprim, $diez[i_1][j_1] \leq lat$, iar dacă se întâlnesc vocale pe cele două laturi se contorizează, $voc[i_1][j_1] - voc[i_1 - 1][j_1 + 1] > 0$ și respectiv $voc[i_1 - lat][j_1 + lat] - voc[i_1 - lat - 1][j_1 + lat + 1] > 0$
     - se parcurg pozițiile din interiorul segmentului $(i-1, j) - (i - lat, j - lat + 1)$ și dacă se găsește # ne oprim, $diez[i_1][j_1] \leq lat - 1$

Complexitatea finală este $\mathcal{O}(N^2)$.

3 Problema Prosum

Propunător: prof. Mihai Bunget, Colegiul Național "Tudor Vladimirescu", Târgu-Jiu

Subtask-ul 1 ($2 \leq N \leq 5\ 000$, $0 \leq a_i \leq 10^9,\ 1 \leq M \leq 10^9$) – 23 de puncte

Se parcurg toate perechile de indici (i, j), cu $i < j$, verificând pentru fiecare pereche condiţia problemei. Complexitate $O(N^2)$.

Subtask-ul 2 ($N \leq 1000$) – 13 puncte

Se parcurg toate perechile de indici (i, j), cu $i < j$, iar cum produsul $a_i \cdot a_j$ depăşeşte valoarea maximă a tipului de date long long, se află cifrele numărului a_i, se înmulţeşte acest număr cu a_j, apoi se reconstituie produsul, modulo M. Complexitate $O(N^2 \cdot C)$. (C este numărul cifrelor numerelor a_i)

Subtask-ul 3 ($0 \leq a_i \leq 10^9,\ 1 \leq M \leq 10^9$, iar numerele a_i au în scrierea binară cel mult 3 cifre egale cu 1) – 33 de puncte

Se determină toate numerele cel mult egale cu 10^9 care au în scrierea binară cel mult 3 cifre egale cu 1, existând cel mult 4226 astfel de numere. Se determină frecvenţele acestor numere în şirul dat, apoi se parcurg perechile de indici (i, j), cu $i < j$, şi dacă e îndeplinită condiţia problemei se adună la soluţie produsul frecvenţelor celor două numere cu indicii (i, j). Complexitate $O(B^2)$. (B este numărul numerelor cu maxim 3 cifre binare egale cu 1, cel mult egale cu 10^9)

Subtask-ul 4 ($2 \leq N \leq 100\ 000$, $0 \leq a_i \leq 10^{18}$, $1 \leq M \leq 10^{17}$) – 31 de puncte

Să observăm că numărul $a_i \cdot a_j + a_i + a_j$ este divizibil cu M dacă şi numai dacă există un număr natural C astfel încât $a_i \cdot a_j + a_i + a_j = C \cdot M$, relaţie care mai poate fi scrisă şi sub forma $(a_i + 1) \cdot (a_j + 1) - M \cdot C = 1$.

Folosind algoritmul lui Euclid extins pentru numerele $(a_i + 1)$ şi M, se determină numerele x şi y astfel încât $(a_i + 1) \cdot x + M \cdot y = 1$, cu $x > 0$ şi $x \leq M$ (x este unic având aceste condiţii). Se caută apoi x printre numerele de forma $a_j + 1$, numărul apariţiilor lui x adunându-se la soluţie. Dacă $x = a_i + 1$ atunci se va scădea 1 din numărul soluţiilor. Complexitate $O(NlogM)$.

Observaţii:

1. Numerele de forma $a_i + 1$ fie se sortează, apoi x se caută binar în şirul sortat, fie se păstrează în nişte liste, un număr $a_i + 1$ fiind adăugat la lista cu indicele $(a_i + 1)\%P$, cu P fixat (căutarea lui x în aceste liste făcându-se secvenţial). De asemenea, se poate folosi şi tipul de date map din STL.

2. Există proprietatea că în algoritmul lui Euclid extins, la fiecare pas, numerele x şi y au proprietatea că $|x| \leq b$ şi $|y| \leq a$, unde x şi y verifică relaţia $a \cdot x + b \cdot y = gcd(a, b)$. Astfel, pentru cazul în care $a_i \leq 10^{18}$, pe parcursul execuţiei algoritmului operaţiile efectuate nu vor depăşi valoarea 10^{18}.

3. De asemenea, pentru existenţa soluţiilor ecuaţiei $(a_i + 1) \cdot x + M \cdot y = 1$, trebuie verificată condiţia ca numerele $a_i + 1$ şi M să fie prime între ele.

Echipa

Setul de probleme pentru această rundă a fost pregătit de:

- prof. Boian Dumitru Flavius, Colegiul Național ”Spiru Haret”, Târgu-Jiu
- prof. Bunget Mihai, Colegiul Naţional ”Tudor Vladimirescu”, Târgu-Jiu
- prof. Cheșcă Ciprian, Liceul Tehnologic ”Grigore C. Moisil”, Buzău
- prof. Costineanu Veronica Raluca, Colegiul Național ”Ștefan cel Mare”, Suceava
- prof. Dumitrascu Dan Octavian, Colegiul National Dinicu Golescu, Câmpulung
- prof. Iordaiche Cristina, Liceul Teoretic ” Grigore Moisil” Timisoara, Timișoara
- prof. Lica Daniela, Centrul Județean de Excelență, Prahova
- prof. Nicoli Marius, Colegiul Național ”Frații Buzești”, Craiova
- prof. Nodea Gheorghe-Eugen, Colegiul Național ”Tudor Vladimirescu”, Târgu-Jiu
- prof. Piț-Rada Ionel-Vasile, Colegiul Național ”Traian”, Drobeta-Turnu Severin
- student Pop Ioan-Cristian, Universitatea Politehnica București, Facultatea de automatică și calculatoare.
- prof. Pracsiu Dan, Liceul Teoretic Emil Racoviță, Vaslui
- student Tulbă-Lecu Theodor-Gabriel, Universitatea Politehnica București, Facultatea de automatică și calculatoare.