DESCRIEREA SOLUȚIILOR, OLIMPIADA JUDEȚEANĂ DE INFORMATICĂ, CLASA A10-A

COMISIA ȘTIINȚIFICĂ

Problema 1: Arhitect

Propusă de: Lect. Diac Paul, Facultatea Informatică, Universitatea Alexandru Ioan Cuza din Iași

Segmentele sunt paralele cu axele sau bisectoarele. atunci segmentele au pantele cu unghiuri de 0°, 45°, 90° sau 135° și se pot număra segmentele care sunt aliniate de la început, adică verticale deci $x_1 = x_2$ sau orizontale deci $y_1 = y_2$, fie numărul acestora a. Toate celelalte segmente se pot alinia printr-o rotație cu 45°. Deci răspunsul va fi $maxim(a, N\text{-}a)$: fie nu rotim segmentele, fie le rotim cu 45°.

Soluție $O(N^2)$ aproximativ 50 puncte. Fixăm orice segment $1 \leq i \leq N$ pentru a fi aliniat. Prin rotirea tuturor segmentelor cu unghiul necesar alinierii lui i se vor alinia și alte segmente: cele care sunt paralele și cele care sunt perpendiculare pe i. Putem evita implementarea efectivă a rotațiilor dacă doar numărăm aceste segmente printr-o altă parcurgere. Pentru a testa dacă două segmente sunt paralele putem translata un segment astfel încât segmentele să aibă un capăt comun, segmentele sunt paralele dacă triunghiul format are aria zero. Segmentele sunt perpendiculare, dacă în triunghiul format se verifică terema lui Pitagora.

Soluție $O(NlogN)$ punctaj maxim. Putem reprezenta fiecare segment printr-un singur punct dacă translatăm toate segmentele astfel ca un capăt al lor să fie $(0, 0)$. Mai departe, pentru simplitate, mutăm orice segment cu valoarea y negativă, în simetricul lui față de $(0, 0)$, adică $(x, y < 0)$ devine $(-x, -y)$ iar panta este aceeași. Acum, pentru a trata cazul segmentelor perpendiculare, orice segment cu $x < 0$ poate fi rotit cu 45° și astfel vom avea toate segmentele reprezentate de puncte cu ambele coordonate pozitive. Rotația cu 45° se poate aplica transformând punctul $(x < 0, y)$ în punctul $(y, -x)$. În sfârșit, acum este suficient să calculăm numărul maxim de segmente ce au aceeași pantă. Coordonatele fiind naturale, putem reprezenta pantele prin perechi de coordonate simplificate: (x, y) devine $\Big(x/cmmdc(x, y), y/cmmdc(x, y)\Big)$. Acum trebuie să determinăm numărul maxim de segmente *egale*. Pentru aceasta, putem sorta segmentele după pantă ori coordonate. Apoi, printr-o singură parcurgere găsim numărul maxim de segmente de pe poziții consecutive egale sau, alternativ, segmentele pot fi numărate printr-un tablou asociativ adică un *map* <>. În funcție de operațiile care se folosesc pentru calcule, trebuie folosit tipul *long long* evitând depășirile.

Problema 2: Bingo

Propusă de: Stud. Popa Sebastian, Facultatea de Matematică și Informatică, Universitatea din București

Problema ne cere să aflăm numărul minim de interschimbări de elemente adiacente ce trebuie efectuate pentru a obține o subsecvență egală cu *bingo* într-un șir dat.

Subtask 1. În acest subtask, fiecare șir primit are 5 caractere, dar cum se garantează faptul că fiecare element din mulțimea $\{b,i,n,g,o\}$ apare cel puțin o dată, rezultă că toate șirurile date sunt, de fapt, anagrame ale șirului țintă *bingo*, deci trebuie doar să rearanjăm literele date. Considerând șirul *bingo* permutarea identică de lungime 5, iar șirul **S** permutarea asociată lui ($b \longmapsto 1$, $i \longmapsto 2$ ...), rezultatul este **numărul de inversiuni** din această permutare.

Exemplu: *biong* $\longrightarrow \binom{1\,2\,3\,4\,5}{1\,2\,5\,3\,4} \longrightarrow$ 2 inversiuni $\longrightarrow$ 2 interschimbări necesare

Subtask 2. Șirurile conțin tot o singură anagramă a șirului *bingo*, dar de data aceasta pozițiile caracterelor ce o formează nu mai sunt consecutive. Am rezolvat mai sus cazul în care pozițiile sunt consecutive, deci ar fi folositor să reducem și acest caz la problema anterioară.

Pentru orice șir, există cel puțin o soluție optimă în care, mai întâi, caracterele ce duc la soluția optimă sunt aduse pe poziții consecutive (fără a se schimba ordinea relativă a lor), după care se rezolvă anagrama obținută ca la subtask-ul 1.

De asemenea, se poate demonstra că este optim să aducem caracterele lângă cel din mijloc, iar pe acesta să nu îl *mutăm*. Considerând că pozițiile celor 5 caractere formează un vector (sortat strict cescător), interschimbările dintre elemente pot fi asimilate cu adăgarea sau scăderea unui 1 la un element din șir atâta timp cât NU se interschimbă între ele două caractere dintre cele 5 selectate. Ne-am dori ca vectorul sa fie format din numere consecutive, însă este mai ușor să vedem cum facem vectorul să fie format doar din elemente egale. Așadar, adăugăm un 2 la primul element, un 1 la al doilea, scădem un 1 din penultimul și scădem un 2 din ultimul. Astfel, ținem cont de faptul că nu trebuie să aducem toate elementele pe poziția din mijloc (ci fieare pe pozițiile adiacente corespunzătoare), iar șirul rămâne sortat crescător. Acum, numărul minim de interschimbări necesare pentru a aduce cele 5 caractere pe poziții consecutive este egal cu numărul minim de incrementări/decrementări necesare pentru a face toate elementele egale. Știm că este optim să le facem egale cu valoarea mediană[1], iar șirul nostru fiind sortat, aceasta este cea din mijloc.

Exemplu: *hhnhbihog* $\longrightarrow \{3,5,6,8,9\} \longrightarrow \{5,6,6,7,7\} \longrightarrow$ 3 incrementări/decrementări necesare pentru a aduce toate elementele la aceeași valoare $(6) \longrightarrow$ 3 interschimbări necesare pentru a ajunge la *hhhnbiogh* $\longrightarrow$ continuăm cu rezolvarea anagramei ca la subtask-ul 1

Subtask 3. Întrucât singurele caractere ce ne interesează, de fapt, sunt cele din mulțimea $\{b,i,n,g,o\}$, acest subtask ne permite să verificăm pentru fiecare variantă de a alege caracterele cu care să formăm subsecvența *bingo*, câte interschimbări sunt necesare. Pentru un 5-tuplu (format din pozițiile celor 5 caractere) fixat, aplicăm raționamentul de la subtask-ul 2. Complexitatea unei astfel de abordări este de ordinul $\mathcal{O}(N^5)$, unde N este lungimea șirului de caractere.

Exemplu: *xboiing* $\longrightarrow \{2,3,4,6,7\} \longrightarrow$ subtask 2
$\longrightarrow \{2,3,5,6,7\} \longrightarrow$ subtask 2

Soluția completă. Pentru fiecare permutare (dintre cele 120 = 5!) a lui *bingo*, încercăm să aplicăm raționamentul de la subtask-ul 3 într-un mod mai eficient. Pentru o permutare fixată, ne interesează doar 5-tuplurile care păstrează ordinea relativă a caracterelor dată de permutare. Dintre acestea, majoritatea nu pot duce la soluția optimă, fiind *depășite* de 5-tuplurile ale căror termeni au pozițiile mai *apropiate* între ele. Pentru o permutare fixată, este de ajuns să considerăm doar aparițiile caracterului din mijloc (pozitia 2, indexând de la 0) și să calculăm pozițiile primului caracter valid din stânga (cel cu indicele 1 din permutare), respectiv primul valid din dreapta (indicele 3). Folosim același raționament pentru caracterele cu indicii 0 și 4.

[1] https://www.geeksforgeeks.org/make-array-elements-equal-minimum-cost/

Observație: Considerăm aparațiile caracterului din mijloc, deoarece ulterior lângă acela trebuie aduse și celelalte, deci de la început și celelalte trebuie alese cât mai aproape de acesta.

Exemplu: Dacă anagrama fixată este *nibog*, pentru fiecare apariție a lui *b* din șir, căutăm primul *i* din stânga lui *b*, respectiv primul *o* din dreapta. Analog, căutăm primul *g* din dreapta lui *o* și primul *n* din stânga lui *i*. Pentru indicii găsiți, aplicăm rezolvarea de la subtask-ul 2.

Putem precalcula aceste poziții succesor/predecesor ale caracterelor *b,i,n,g,o* pentru fiecare poziție din șir sau le putem căuta binar la fiecare pas. Pentru găsirea rezultatului pentru un șir de lungime N, complexitatea este $\mathcal{O}(N \cdot \log N)$ sau $\mathcal{O}(N)$. Trebuie menționat că avem un factor constant aproximativ egal cu 120-130.

Soluție alternativă. Vlad-Mihai Bogdan, Universitatea din București

Pentru fiecare poziție i, vom încerca să formăm șirul *bingo* pe pozițiile $i, i+1, ..., i+4$. Pentru fiecare literă l, care ar urma să fie pe poziția $i+k$ în șirul cu subsecvența *bingo* pe pozițiile $i, i+1, ..., i+4$, avem cel mult trei opțiuni:

- să luăm litera l din stânga poziției $i+k$, dacă este posibil;
- să luăm litera l din dreapta poziție $i+k$, dacă este posibil;
- să luăm litera l chiar de pe poziția $i+k$, dacă este posibil.

Astfel, pentru fiecare poziție i, avem cel mult 3^5 configurații posibile de alegeri pentru cele 5 litere. Vom calcula numărul de swap-uri necesare pentru a aduce literele pe pozițiile $i, i+1, ..., i+4$ în ordinea pozițiilor lor din șirul inițal (spre exemplu, dacă decidem să luăm litera n de pe o poziție de dinaintea poziției literei i, ordinea în care vom aduce literele este n, iar mai apoi i). La final, trebuie să adăugăm la răspuns numărul de swap-uri necesare pentru a transforma anagrama rezultată în șirul *bingo*.

După precalcularea numărului de swap-uri necesare pentru fiecare anagramă a cuvântului *bingo*, complexitatea devine $\mathcal{O}(N)$, dar cu o constantă aproximativ egală cu 243.

Problema 3: Fotbal

Propusă de: Stud. Bogdan Vlad-Mihai, Facultatea de Matematică și Informatică, Universitatea din București

Problema ne cere numărul de moduri în care putem alege K intervale, a căror intersecție nu este mulțimea vida și printre care se regăsește cel puțin un interval colorat în negru ($f_i = 1$) și cel puțîn un interval colorat în alb ($f_i = 0$), dintr-o mulțime de N intervale date.

Subtask 1. În acest subtask, este suficient să verificăm fiecare pereche de intervale. Intervalele trebuie să aibă culori diferite și intersecția nevida. Complexitatea unei astfel de soluții brut-force este $O(N^2)$.

Subtask 2. Vom avea o abordare asemănătoare cu cea de la subtask-ul anterior. Observația cheie este că trebuie să fixăm un interval cu $f_i = 0$, iar mai apoi putem trece prin restul de N intervale și să verificăm condițiile de mai sus. O astfel de soluție are complexitate $O(N \cdot cnt_0)$, unde cnt_0 este numărul de intervale cu $f_i = 0$.

Subtask 3. Pentru ca intersecția a K intervale să fie nevidă, trebuie să se respecte următoarea proprietate: $max(left_i) \leq min(right_i)$ cu i de la 1 la K, unde $left_i$ reprezintă capătul din stânga al intervalului i (dintre cele K alese), iar $right_i$ capătul din dreapta al intervalului i (dintre cele K alese).

Prin urmare, vom sorta crescător intervalele în funcție de capătul din stânga. Vom parcurge intervalele în noua ordine și vom menține două variabile, cnt_0 și cnt_1, reprezentând numărul de intervale active de culoare albă, respectiv numărul de intervale active de culoare neagră. Când vrem să trecem de la intervalul $i-1$ la intervalul i, trebuie să actualizăm cnt_0 și cnt_1, eliminând toate intervalele pentru care $right_j < left_i$. Pentru a nu număra intervale de mai multe ori, vom calcula pentru fiecare interval i, numărul de moduri în care putem alege K intervale care să respecte condițiile de mai sus, fixând cel mai din dreapta dintre intervale că fiind intervalul cu numărul i. Cu variabilele cnt_0 și cnt_1 actualizate, singură condiție de care trebuie să ținem cont este condiția că trebuie să avem cel puțin un interval din fiecare culoare.

Pentru a ține cont de această condiție vom calcula mai întâi numărul de moduri prin care putem alege restul de $K-1$ intervale, iar mai apoi vom scădea numărul de moduri în care putem alege $K-1$ intervale de aceeași culoare cu intervalul i. Acest număr este dat de formula $\binom{cnt_0+cnt_1}{K-1} - \binom{cnt_i}{K-1}$, unde cnt_i este numărul de intervale de aceeași culoare cu intervalul i.

Pentru a calcula combinările $\binom{N}{K}$, ne vom folosi de triunghiul lui Pascal și de formula recurentă $\binom{N}{K} = \binom{N-1}{K} + \binom{N-1}{K-1}$. Complexitatea unei astfel de soluții este $O(N^2)$.

Soluția completă este soluția anterioară, pentru calcul combinărilor se va folosi inversul modular și algoritmul lui Euclid extins. Această soluție are complexitate $O(N \cdot logN)$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Anton Cristiana Elena
- Stud. Apostol Ilie-Daniel
- Stud. Bogdan Vlad-Mihai
- Lect. Diac Paul
- Drd. Ioniță Alexandru
- Prof. Moț Nistor
- Prof. Pintea Rodica
- Stud. Popa Sebastian
- Stud. Popescu Stefan-Alexandru
- Stud. Râpeanu George