Editorial Neoficial Stelele Informaticii

Decembrie 2023

Cuprins

1 Multumiri

Acest concurs a fost un concurs organizat in decembrie 2023 de elevi ai Liceului Teoretic International de Informatica Bucuresti (ICHB). Pentru mai multe detalii puteti accesa siteul oficial al concursului.

- Rares Felix Tudose, Mihai-Valeriu Voicu, Tiberiu Stefan Cozma, Theodor-Gabriel Tulbă-Lecu, autorii problemelor și laureati la concursuri nationale si internaționale de informatică;
- Mihai Valeriu Voicu, autorul editorialelor oficiale și a îmbunătățirilor la editorialele neoficiale.
- Ștefan Alecu, creatorul acestui șablon LaTeXpe care îl folosim;
- Stefan Cosmin Dascalescu, testerul rundei si autorul unora din editoriale
- Luca Valentin Muresan, Roland Petrean și Vlad Tutunaru, autorii editorialelor neoficiale
- Theodor-Gabriel Tulbă-Lecu, coordonatorul concursurilor;

2 Umplere simetrica

Autor: Rareș Felix Tudose

2.1 Soluția

2.1.1 Soluția de 56

Ne vom creea un graf în care asociem un nod fiecărei celulă din matrice. Fie $u_{i,j}$ valoarea asociată nodului (i, j). Pentru a asocia nodurile, am putea de exemplu să parcurgem matricea, și să asociem pe rând valorile, $1, 2, 3, \cdots n \cdot m$. Vom trasa muchie între două noduri, dacă acestea au obligatoriu aceeași valoare. De exemplu, dacă avem restricția $(1, 1, 1, 2)$, celulele $(1, 1)$ și $(1, 2)$ au obligatoriu aceeași valoare. Așadar, vom trasa o muchie între $u_{1,1}$ și $u_{1,2}$. Acum, observăm că pentru o componentă conexă, toate nodurile trebuie să aibă aceeași valoare. De asemenea, componentele conexe sunt complet separate, deci fiecare componentă conexă se poate rezolva în exact 2 moduri. În total, răspunsul este 2^{cc}, unde cu cc am notat numărul de componente conexe. Acum, trebuie să trasăm muchiile.
O soluție brute force ar fi următoarea: Ne fixăm un punct din submatricea de la restricție și trasăm muchie între el și oglinditele sale. Complexitate: $O((c - a) \cdot (d - b))$ pentru fiecare restricție. Astfel obținem 56 de puncte.

2.1.2 Soluția de 100

Trebuie să optimizăm modul în care trasăm muchiile. În rest, soluția se încadrează în timp. Ne vom uita la oglinditele față de axa orizontală (cele față de axa verticală se vor rezolva analog).
Dacă pe linia i și două coloane $x \leq y$ am adăugat muchie de la $u_{i,x}$ la $u_{i,y}$, de la $u_{i,x+1}$ la $u_{i,y-1}$, $u_{i,x+2}$ la $u_{i,y-2}$, ···, când va trebui din nou să adăugăm un set de restricții între coloanele $x-1, y+1$, nu are sens să adăugăm din nou muchie de la $u_{i,x}$ la $u_{i,y}$, de la $u_{i,x+1}$ la $u_{i,y-1}$ ···. Pentru a rezolva astfel de situații, am putea să ne fixăm mijlocul unui set de restricții (de la x la y, $x+1$ la $y-1$ ···) și vom calcula lungimea maximă cu care ne vom extinde. Putem actualiza lungimile în $O(n)$ pentru fiecare restricție. Trebuie acordată atenție la faptul că mijlocul se poate afla între două celule (de exemplul mijlocul lui $(1, 2)$).
Cel mai ușor mod de a rezolva această problemă este să dublăm șirul.
În final, vom avea cel mult $O(n^3)$ restricții de adăugat.

2.1.3 Cod sursă

Soluție de 100

3 Etichetare

Autor: Theodor-Gabriel Tulbă-Lecu

3.1 Soluția

In loc sa consideram sirul pozitiilor atinse, sa consideram ca facem urmatoarea: Avem doua contoare, unul de pozitie p si altul de eticheta e. La un pas, aplicam $e := e + 1$, $p := p \pm 1$ si $v_p = e$.
Sa consideram un sir de operatii care cumva ne aduce sa punem eticheta ceruta de Denis pe pozitia a. Urmatoarea mutare conteaza foarte mult: daca alegem sa mergem in stanga, nu vom putea niciodata sa mutam p in asa fel incat $p \geq a$ (pentru ca asa am reeticheta pozitia a, si nu am vrea asta intr-atat ca aceasta deja are eticheta dorita). Analog pentru dreapta. Deci, daca si in stanga si in dreapta avem pozitii neetichetate cu eticheta dorita, nu le vom putea completa pe ambele.
Mai apoi, putem spune ca astfel devine clar ca trebuie sa gasim un mijloc prin care atasam etichetele in ordine crescatoare (adica, prioritatea dupa ce punem cea mai mica eticheta necesara este sa gasim un drum de la pozitia acesteia care nu suprascrie aceasta eticheta catre pozitia care are a doua cea mai mica eticheta). Astfel, procesul de etichetare va pune o eticheta dorita ori in cea mai din dreapta pozitie neetichetata, ori in cea mai din dreapta (deci sirul de etichetari trebuie sa fie de tip munte), mereu locul geometric in care ne

miscam liber fiind un segment. Cu astea fiind spuse, consideram urmatoarea dinamica, parametrizata in paralel si de doua valori l si r, reprezentand segmentul antementionat de pozitii a caror eticheta inca nu am fixat-o, initializata cu $dp_1 = 1, l = 1, r = n$. Recurenta este:

$$dp'_j += dp_i \text{ iff } |i - j| = 1 \text{ si } l <= j <= r \,;\, dp := dp'$$

(in ideea ca daca am construit un drum pana la i, putem sa mergem in orice element vecin pentru a pune urmatoarea eticheta, atata timp cat nu reetichetam vreo pozitie)
Aplicam aceasta transformare dp-ului de min e_i ori ne va da pe pozitia acestei etichete minime numarul de drumuri posibile de la 1 la aceasta. Apoi, cum noi vrem sa consideram doar drumuri care ajung pana aici, setam toate celelalte valori din dp la 0. Acum, daca pozitia era la capatul dreapta, aplicam $r-- = 1$, altfel, $l++ = 1$. Continuam sa aplicam transformarea pana cand efectuam un numar total de transformari egale cu $\min_2 e_i$. Cand ajungem la aceasta, resetam in mod similar restul valorile si modificam l sau r in mod similar. Cand aplicam acesti pasi pentru fiecare eticheta, valoarea ce ramane este raspunsul final.
Complexitate timp: $O(N * K)$

3.1.1 Cod sursă

Soluție de 100

4 Omida

Autor: Mihai Valeriu Voicu

4.1 Soluția

Intai, este clar ca putem clasifica fiecare tinta dupa diagonala asupra caruia trebuie sa stea arcasul pentru a putea sa traga catre ea, intr-atat ca fiecare sageata pastreaza invariantul diferentei intre abscisa (coordonata X) si ordonata (coordonata Y).
Prima observatie: Singurele mutari pe care le va face arcasul vor fi in jos si la dreapta. Motivul pentru aceasta tine de faptul ca viteza unei sageti relativ fata de diagonala este mult mai mare decat cea pe care ar putea sa o compenseze arcasul. In esenta, daca arcasul se afla pe o diagonala si ar trage o sageata in directia pozitiva, iar apoi ar face un pas (plecand astfel de pe diagonala initiala pe una adiacenta), in urma unui pas facut inapoi catre diagonala initiala, arcasul a putut face doar un pas in directia diagonalei, pe cand sageata ar fi putut sa faca doua
I.e., drumul arcasului in doua secunde pentru a se intoarce pe diagonala ar putea arata ca: (X, Y), (X + 1, Y), (X + 1, Y + 1), pe cand o sageata lansata de la (X, Y) ar putea ajunge la (X + 1, Y + 1) intr-o singura unitate de timp.
Astfel, pentru o diagonala, vom avem o singura sansa sa anihilam toate tintele de pe ea. Astfel, este clar ca ne vor pune cele mai multe probleme tintele de

pe margine, intr-atat ca putem lansa oricate tinte in acelasi timp.
Solutia de acum este sa cautam binar timpul maxim in care ne propunem sa rezolvam sistemul. Fie valoarea fixata T. Pentru o diagonala de tipul (Y + δ, Y), stim ca trebuie sa avem coordonata Y cand ajungem pe diagonala respectiva in $[min - (\delta - T), min + (\delta - T)] \cap [max - (\delta - T), max + (\delta + T)]$. In plus, daca stim ca putem in urma tuturor operatiilor facute pana la o asemenea diagonala noi stim ca putem obtine orice coordonata Y dintr-un interval $[L, R]$, ulterior unor α mutari libere la dreapta/jos prin niste diagonale care nu impun asemenea restrictii (cum nu au tinte pe ele), este clar ca putem obtine orice coordonata Y din intervalul $[L - \alpha, R]$ (cum la fiecare pas liber, putem ori merge in (X, Y - 1) sau in (X, Y + 1). Cu atat mai mult, acest segment de coordonate posibile incepe de pe diagonala cu delta caracterstic 0 ca $[0, 0]$, astfel demonstrand de ce locul geometric al coordonatelor posibile in urma tuturor conditiilor ramane un interval (in urma diagonalelor libere acesta se extinde prin metoda antementionata, si in urma unei diagonale cu restrictii, luam locul geometric calculat anterior si il intersectam cu cel de care avem nevoie pentru a admite solutie).
Daca in urma acestei baleieri, locul geometric nu ajunge niciodata vid, inseamna ca sistemul admite solutie in cel mult T pasi.

4.1.1 Cod sursă

Soluție de 100

5 Toska

Autor: Mihai Valeriu Voicu

5.1 Soluția

5.1.1 Subtaskurile $1, 2$, 20 de puncte

Putem fixa doua perechi de noduri, iar apoi calcula drumul dintre ele. Apoi, putem verifica daca conditia din enunt folosind un vector de frecventa. Complexitate timp: $O(N^3)$

5.1.2 Subtaskurile $3, 4$, 40 de puncte

Putem fixa un singur nod, iar apoi putem mentine frecventa culoriilor muchiilor pana la un nod parcurgand arborele in adancime. Astfel, pentru fiecare nod, putem verifica daca frecventa fiecarei culoare adaugata in frecventa pana acum este para. Complexitate timp: $O(N^2)$

5.1.3 Subtaskul 5, 75 de puncte

Exista mai multe moduri de a rezolva acest subtask. Pentru simplitate, noi il vom aborda asemenea: fiecarei muhcii de culoare c ii vom asocia un nou cost

egal cu 2^c. Acum, daca am lua suma *xor* a oricarui lant (sa o notam pe aceasta cu $s(u, v)$ pentru doua noduri u si v) am putea face observatii asupra paritatii frecventei fiecarei culori de pe lant. In plus, daca aceasta suma este 0, atunci inseamna ca toate culoriile apar de un numar par de ori
Acum, sa inradacinam arbitrar arborele (fie in nodul R), si sa notam pentru fiecare nod u cu $d_u = s(R, u)$. Mai apoi, $d_u \oplus d_v = s(u, v)$, pentru ca, daca am nota cu L cel mai apropiat stramos comun al nodurilor u si v, atunci putem scrie $s(u, v) = (d_u \oplus d_L) \oplus (d_v \oplus d_L) = d_u \oplus d_v \oplus (d_L \oplus d_L) = d_u \oplus d_v$
Deci, putem grupa nodurile dupa d_u, iar apoi, putem stii ca orice lant cu doua noduri din aceeasi grupa formeaza un lant dorit.
Complexitate timp: $O(N \cdot log(N))$

5.1.4 Subtaskul 6, 100 de puncte

Putem aplica un rationament similar subtaskului 5, doar ca fiecarei culori ii vom asocia un cost aleatoriu (pe care il vom folosi ulterior in calculariile valoriilor d_u).
Complexitate timp: $O(N \cdot log(N))$

5.1.5 Cod sursa

Solutie de 100 de puncte

6 Florinel

Autor: Rareș Felix Tudose

6.1 Soluția oficială

Inainte de toate, putem simplifica numararea triunghiurilor intr-un graf turneu in felul urmator:
Observam ca, din punctul de vedere al gradelor de iesire, un graf turneu pe 3 noduri poate avea lista de grade ori (111), ori (210). Este clar ca triunghiurile dorite sa fie numarate sunt subgrafurile de tip (111). Astfel, putem numara in schimb subgrafurile de tip (210), si sa scadem numarul acesta din numarul total de subgrafuri pe 3 noduri.
Apoi, in particular, orice subgraf de tip (210) poate fi generat in urma selectarii nodului cu grad de iesire 2, si apoi oricare doi vecini ai acestuia. Orientarea muchiei dintre vecini este irelevanta, intr-atat ca vom stii clar ca unul din ei va avea astfel grad de iesire 1, si celelalt 0, astfel indeplinind criteriul ca subgraful sa fie de tip (210).
In cele din urma, putem spune ca numarul de triunghiuri este $\binom{n}{3} - \sum_{v=1}^{n} \binom{d_v}{2}$, unde d_v este gradul de iesire al nodului v. Vrem ca acest numar sa fie maxim, deci vrem sa minimizam $\sum_{v=1}^{n} \binom{d_v}{2}$. $\binom{x}{2}$ este functie convexa dupa x, astfel ca din inegalitatea Karamata, $\sum_{v=1}^{n} \binom{d_v}{2} \geq n * \binom{\frac{\sum_{v=1}^{n} d_v}{n}}{2}$.

Intuitiv, toate aceastea inseamna ca este optim ca toate gradele sa fie egale. https://steleleinformaticii.ichb.ro/ Intr-un final, pentru a rezolva problema, există diverse construcții care funcționează, precum table de șah sau construcții simetrice pline cu 0 și 1 pe linii și coloane; atata timp cat gradele de iesire raman egale, solutia va fi valida.

6.1.1 Cod sursă

Soluție de 100

7 Drum Luminat

Autor: Rareș Felix Tudose

7.1 Soluția oficială

Observație secundară:
Dacă înmulțim un număr cu 0, produsul va fi mereu 0. Dar, toate valorile din matrice sunt fie ≥ 0. Deci, preferăm să nu luam 0.
Observăm că matricea se împarte în niște componente conexe, delimitate de 0-uri. Dacă (x_s, y_s) și (x_f, y_f) se află în componente diferite, nu putem obține alt produs de 0. Altfel, vom rezolva problema pentru aceea componentă.
Observație principală:
Vom încerca să fixăm răspunsul. Fie x valoarea fixată.
$(L[x_1][y_1] \cdot L[x_2][y_2] \cdot L[x_3][y_3] \cdot ... L[x_k][y_k])^{1/k} \geq x \iff$
$L[x_1][y_1] \cdot L[x_2][y_2] \cdot ... \cdot L[x_k][y_k] \geq x^k \iff (\frac{L[x_1][y_1]}{x}) \cdot (\frac{L[x_2][y_2]}{x}) \cdot ... \cdot (\frac{L[x_k][y_k]}{x}) \geq 1$
Observăm că fiecare factor apare înmulțit cu $\frac{1}{x}$. Deci, putem înmulți fiecare valoare din matricea inițială cu $\frac{1}{x}$ și vrem să găsim dacă există vreun drum cu produsul ≥ 1.
O idee ar fi să căutăm două pătrate adiacente în matrice și să tot mergem din unul în altul de o infinitate de ori (nu chiar). Ca să creștem produsul, trebuie să avem produsul pătratelor adiacente strict mai mare ca 1.
$\frac{a}{v} \cdot \frac{b}{v} \geq 1 \iff a \cdot b \geq v^2 \iff v \leq \sqrt{a \cdot b}$. Deci, am putea alege de exemplu

$v = \sqrt{a \cdot b} - 10^{-100}$, sau chiar $v = \sqrt{a \cdot b}$. (merge, deoarece diferența e foarte mică). După ce am făcut un număr foarte mare de astfel de operații, putem doar să alegem orice drum care nu trece prin 0. Deci, dacă alegem o pereche de celule consecutive și luăm $v = max\sqrt{a \cdot b}$, unde a și b sunt valori din două pătrate adiacente.
Acum, să demonstrăm că aceasta valoare e cea optimă. Vom înlocui in formulă $x = v$. $\prod \frac{L[x_i][y_i]}{v} \geq 1$. Dacă am avea un drum de lungime ≥ 3 care l-am putea folosi de multe ori (cum l-am folosit pe cel de lungime 2), ar însemna ca (exemplu pentru lungime 3) $\frac{a}{v} \cdot \frac{b}{v} \cdot \frac{c}{v} \geq 1$. Vom considera $a \geq b \geq c$. Deci, $\frac{a}{v} \cdot \frac{b}{v} \geq \frac{a}{v} \cdot \frac{b}{v} \cdot \frac{c}{v} \geq 1$. Deci $\frac{a}{v} \cdot \frac{b}{v} \cdot \frac{c}{v} \geq 1$, așadar am putea alege doar a, b în loc de a, b, c.

7.1.1 Cod sursă

Soluție de 100

8 Swap and Mex

Autor: Tiberiu Stefan Cozma

8.1 Soluția

8.1.1 Solutii brute

Pentru primele subtaskuri, se poate simula procesul în diverse moduri pentru a putea obține punctajele parțiale. Pentru a obține 60 de puncte, se poate rezolva query-ul în $O(n)$ considerând minimele pe prefix și pe sufix pentru ambele șiruri, iar pentru a putea răspunde YES la un query, trebuie să verificăm ca pentru ambele șiruri, pentru toate pozițiile de la 1 la n, aceste minime pe prefixe și sufixe sunt identice.
Motivul pentru care această soluție merge este acela că dacă ar fi la un moment dat aceste minime diferite, înseamnă că există o subsecvență într-unul din șirurile date care au mex-urile diferite.

8.1.2 Solutia de 100

Pentru a rezolva problema, plecăm de la calcularea mex-urilor pe prefixe și sufixe pentru șirurile inițiale, și observăm faptul că atunci când schimbăm valorile dintr-un șir, doar două valori din șirul de prefixe și sufixe se schimbă,

pentru că dacă schimbăm valorile de la poziția x și $x + 1$, prefixele de la $x + 2$ la n rămân identice, la fel ca sufixele de la $x - 1$ până la 1.
Pentru a trata schimbările, este îndeajuns să verificăm când s-ar produce modificarea acelui mex, iar în funcție de relația dintre a_x și prefixul de la $x + 1$, preluăm fie prefixul următor, fie valoarea a_x. Analog se va proceda și în cazul sufixelor.
În mod similar cu soluția de 60 de puncte, va trebui să verificăm dacă toate mexurile de pe prefixe și sufixe sunt egale, lucru ce se poate face cu ușurință folosind vectori de frecvență sau seturi. Soluții alternative există cu diverse structuri de date, precum arborii de intervale sau arbori indexați binari.
Pentru mai multe detalii de implementare, accesați linkul expus mai jos.

8.1.3 Cod sursă

Soluție de 100

9 Haos

Autor: Rareș Felix Tudose

9.1 Soluția

Definim b_i ca fiind câți bani ar pierde / câștiga prietenul i după ce s-ar achita toate plățile inițiale.
Să considerăm graful în care fiecare nod reprezintă un prieten și o muchie între u și v înseamnă că u poate transfera bani lui v. Vom rezolva problema individual pentru fiecare componentă conexă din acest graf, considerând doar prietenii care fac parte dintr-o componentă, deoarece doi prieteni din componente diferite nu își pot influența balanța.
Pentru ca o soluție să existe, trebuie ca suma balanțelor prietenilor din componentă să fie egală cu 0 - dacă un prieten câștigă bani, trebuie să existe o altă mulțime de prieteni care pierd aceeași sumă de bani. Dacă soluția există, putem mereu să construim o mulțime de relații echivalentă de mărime $n - 1$, unde n e numărul de prieteni din componentă.
Să vedem o astfel de construcție: luăm orice arbore parțial al componentei și îl înrădăcinăm într-un nod oarecare, apoi pornim un DFS. Pentru fiecare nod u, vom adăuga o relație între el și părintele său p_u, iar în total vom avea $n - 1$ relații. De asemenea notăm $t_u = \sum b_v$ pentru fiecare v din subarborele lui u.

- Dacă $t_u < 0$, înseamnă că din subarborele lui u se pierd $-t_u$ bani, și vom adăuga relația $u \xrightarrow{-t_u} p_u$.
- Altfel, înseamnă că în subarborele lui u este nevoie de t_u bani, și vom adăuga relația $p_u \xrightarrow{t_u} u$.

Rămâne să găsim arborii parțiali pentru fiecare componentă. Folosind o metodă naivă în $O(N^2)$, putem obține 50 de puncte. Pentru a optimiza această soluție, putem să facem un DFS în care ștergem fiecare nod după ce îl vizităm, reducând complexitatea la $O(NlogN)$. Mai multe informații despre această metodă se pot găsi aici.

9.1.1 Cod sursă

Soluție de 100