

Descrierea Soluțiilor

Concursul Național “InfoPro”, Runda 4

Grupa A

1 Problema Grafuri

Propunator: prof. Marius Nicoli, Colegiul Național “Frații Buzzești”, Craiova

1.1 Soluție

Soluția problemei se bazează pe următoarea proprietate: un graf neorientat este bipartit dacă și numai dacă orice ciclu are lungime pară. Astfel, pentru cazul $n=1$ problema are soluție doar dacă graful ciclul dat are număr par de noduri. Acestea se împart în cele două mulțimi pe parități. Pentru cazul în care toate grafurile ciclu sunt pătrate nu trebuie eliminată nicio muchie. Cazul în care toate grafurile ciclu date sunt triunghiuri poate fi tratat și în mod particular considerând că se elimină câte o muchie între oricare două triunghiuri de rand $2K - 1$ și $2K$. În general problema are soluție dacă și numai dacă suma numărului de noduri din toate ciclurile date este un număr par. Demonstrația este consecință a modului de construire a grafului, care la o analiză atentă se deduce că este unic determinat. Din enunț se deduce că se formează un “lanț de cicluri” și trebuie să ne asigurăm că ultimul ciclu de pe lanț este de lungime pară (ciclurile din față să nu ne mai interesează întrucât din enunț se deduce că nu mai pot fi afectate de următoarele grafuri ciclu). Dacă ultimul ciclu al lanțului de cicluri este de lungime impară suntem obligați să rupem muchia suprapusă dintre el și graful ciclu care se adaugă. În caz contrar lăsăm așa muchia suprapusă și ciclul curent se reșetează la graful ciclu care se adaugă (are forma acestui nou graf ciclu, mai puțin reetichetarea a două noduri). Trebuie acordată atenție implementării în etapa reetichetării nodurilor. La final se realizează o parcurgere a grafului obținut, marcând nodurile cu două valori, 1 și 2 (dacă nodul curent este marcat cu 1, vecinii săi vor fi marcați cu 2 și invers). Se recomandă folosirea la parcurgere a algoritmului BFS. DFS poate avea ca efect un necesar mare de memorie în stivă mai ales în cazul grafurilor ciclu cu număr mare de elemente.

2 Problema Matsum

Propunator: Tamio-Vesa Nakajima, Oxford University, Costin-Andrei Oncescu, Oxford University

2.1 Soluția în $O(N^3)$

Se fixează prima și ultima linie din submatricea selectată. Dacă a este șirul sumelor pe coloane, în liniile selectate, atunci matricile cu liniile fixate contribuie la soluție cu $\sum_{i=1}^N \sum_{j=i}^N (a_i + \dots + a_j)^2$. Cump calculăm această valoare?

Fie $f_k : \mathbb{N}^k \rightarrow \mathbb{N}^3$ o funcție care ia un șir a de lungime k de numere și calculează valorile

$$\left(\sum_{i=1}^k a_1 + \dots + a_i, \sum_{i=1}^k (a_1 + \dots + a_i)^2, \sum_{i=1}^k \sum_{j=i}^k (a_i + \dots + a_j)^2 \right)$$

Este clar că funcția f_N ne calculează valoarea necesară (este a 3-a valoare calculată). Putem defini $f_1(a_1) = (a_1, a_1^2, a_1^2)$. Apoi, observăm că

$$\begin{aligned} f_k(a_1, \dots, a_k) &= \left(\sum_{i=1}^k a_1 + \dots + a_i, \sum_{i=1}^k (a_1 + \dots + a_i)^2, \sum_{i=1}^k \sum_{j=i}^k (a_i + \dots + a_j)^2 \right) \\ &= \left(ka_1 + \sum_{i=2}^k a_2 + \dots + a_i, \right. \\ &\quad \left. ka_1^2 + 2a_1 \left(\sum_{i=2}^k a_2 + \dots + a_k \right) + \sum_{i=2}^k (a_2 + \dots + a_i)^2, \right. \\ &\quad \left. \sum_{j=1}^k (a_1 + \dots + a_j)^2 + \sum_{i=2}^k \sum_{j=i}^k (a_i + \dots + a_j)^2 \right) \end{aligned}$$

Astfel, dacă $f_{k-1}(a_2, \dots, a_k) = (x, y, z)$, rezultă că

$$f_k(a_1, \dots, a_k) = (ka_1 + x, ka_1^2 + 2a_1x + y, ka_1^2 + 2a_1x + y + z)$$

Lucru care ne arată cum să definim o funcție recursivă care calculează valoarea cerută, în $O(N)$. Atenție la modulo!

2.2 Soluția în $O(N^2)$

Pentru a obține complexitatea $\mathcal{O}(N^2)$, este nevoie de o interpretare combinatorică a conceptului de “ridicare la patrat a unei sume”. Mai exact, putem interpreta ridicarea la patrat a unei sume ca fiind suma produselor tuturor perechilor ordonate de termeni ai sumei. Putem formaliza asta astfel:

$$\begin{aligned}
& \sum_{\substack{1 \leq i_1 \leq i_2 \leq N \\ 1 \leq j_1 \leq j_2 \leq M}} \left(\sum_{\substack{1 \leq i_1 \leq i \leq i_2 \leq N \\ 1 \leq j_1 \leq j \leq j_2 \leq M}} A_{i,j} \right)^2 = \\
& \sum_{\substack{1 \leq i_1 \leq i_2 \leq N \\ 1 \leq j_1 \leq j_2 \leq M}} \left(\sum_{\substack{1 \leq i_1 \leq i \leq i_2 \leq N \\ 1 \leq j_1 \leq j \leq j_2 \leq M}} A_{i,j} \right) \cdot \left(\sum_{\substack{1 \leq i_1 \leq i \leq i_2 \leq N \\ 1 \leq j_1 \leq j \leq j_2 \leq M}} A_{i,j} \right) = \\
& \sum_{\substack{1 \leq i_1 \leq i_2 \leq N \\ 1 \leq j_1 \leq j_2 \leq M}} \sum_{\substack{1 \leq i_1 \leq i, k \leq i_2 \leq N \\ 1 \leq j_1 \leq j, p \leq j_2 \leq M}} A_{i,j} \cdot A_{k,p} = \\
& \sum_{\substack{1 \leq i, k \leq N \\ 1 \leq j, p \leq M}} A_{i,j} \cdot A_{k,p} \cdot \left(\sum_{\substack{1 \leq i_1 \leq i, k \leq i_2 \leq N \\ 1 \leq j_1 \leq j, p \leq j_2 \leq M}} 1 \right) = \\
& \sum_{\substack{1 \leq i, k \leq N \\ 1 \leq j, p \leq M}} A_{i,j} \cdot A_{k,p} \cdot \min(i, k) \cdot \min(j, p) \cdot (N - \max(i, k) + 1) \cdot (M - \max(j, p) + 1)
\end{aligned}$$

Fezabilitatea calcularii sumei de mai sus in mod eficient provine din independenta ce se poate obtine daca luam toate cazurile legate de cum se compara i cu k si j cu p . Aratam mai jos cum se trateaza cazul $i < k, j < p$. Rezultatul sumei acestor termeni poate fi inmultit cu 2 datorita simetriei si complementat cu cazul $i < k, j > p$ (trebuie tratate si cazurile cu $i = k, j \neq p, i \neq k, j = p$ si $i = j, k = p$, insa acestea sunt strict mai usor de tratat, ba chiar pot fi cu grija tratate in acelasi timp cu cazurile de mai sus). Pentru a trata cazul $i < k, j < p$, se poate folosi exact aceeasi metoda pe matricea rotita.

$$\sum_{\substack{1 \leq i < k \leq N \\ 1 \leq j < p \leq M}} A_{i,j} \cdot A_{k,p} \cdot i \cdot j \cdot (N - k + 1) \cdot (M - p + 1) = \tag{1}$$

$$\sum_{\substack{1 \leq i < N \\ 1 \leq j < M}} A_{i,j} \cdot i \cdot j \cdot \left(\sum_{\substack{i < k \leq N \\ j < p \leq M}} A_{k,p} \cdot (N - k + 1) \cdot (M - p + 1) \right) \tag{2}$$

Observati ca (2) este calculabil in timp patrat datorita independentei sumelor pe care trebuie sa le calculam (i si j apar in suma interioara doar pentru a restrictiona indicii dupa care efectuam suma la un dreptunghi) daca precaluam $B_{i,j} = \sum_{\substack{i < k \leq N \\ j < p \leq M}} A_{k,p} \cdot (N - k + 1) \cdot (M - p + 1)$.

3 Problema Atentie

Propunator: Costin Andrei Oncescu, Oxford University, Autor Extern
Editorial: Andrei Constantinescu, Oxford University

Problema se ataca, in mod posibil surprinzator, destul de liniar—adica este nevoie de o inlantuire de idei, fiecare relativ naturala si nu foarte greu de justificat de sine statatoare.

Pentru a usura explicatia, dat fiind un sir S , format din caracterele ‘U’, ‘D’, ‘L’, ‘R’, vom nota cu X_s numarul de aparitii ale literei $X \in \{\text{‘U’}, \text{‘D’}, \text{‘L’}, \text{‘R’}\}$ in S , si vom defini $A_S = U_S + D_S$ si, respectiv, $B_S = L_S + R_S$. De asemenea, data fiind o celula (i, j) din matrice, sa notam cu $M_{S,(i,j)}$ submatricea cu colturile stanga-jos/dreapta-sus in celulele $(i - D, j - L)/(i + U, j + R)$. Cand sirul S si/sau celula (i, j) sunt clare din context le vom omite din notatie.

Lemă. *Fie S sirul fixat de Nitsoc, si fie (i, j) pozitia unde Argintolac a ales sa plaseze robotul. Atunci, Bronzolac castiga jocul daca si numai daca M contine cel putin un obstacol. Observati cum M poate iesi din limitele matricei, caz in care se considera implicit ca M contine un obstacol.*

Demonstratie. Este clar ca oricum ar permuta Bronzolac sirul S toate pozitiile intermediare ale robotului se vor afla in M . Daca M nu contine nici un obstacol, atunci la fiecare moment de timp robotul nu se va lovi de niciun obstacol si nu va iesi din matrice. Altfel, M contine un obstacol (sau o pozitie din afara matricei), fie el la pozitia $(i_O, j_O) \in M$. Fara a restrange din generalitate, consideram doar cazul $i_O \geq i, j_O \geq j$ (celelalte trei cazuri se trateaza analog). Cum $i \leq i_O \leq i + U$ si $j \leq j_O \leq j + R$, Bronzolac poate permuta sirul S astfel incat sa inceapa cu $i_O - i$ litere ‘U’ si $j_O - j$ litere ‘R’, efectul fiind ca dupa cel mult $i_O - i + j_O - j$ pasi robotul se va lovi de un obstacol (de obstacolul de la pozitia (i_O, j_O) sau chiar mai devreme). Asadar, cum Bronzolac joaca optim, in acest caz el castiga alegand orice S cu proprietatea descrisa, altfel castiga Argintolac, indiferent de strategia acestuia. \square

Teoremă. *Fie S sirul fixat de Nitsoc. Atunci faptul ca Argintolac castiga sau nu este unic determinat de numerele A si B . Mai exact, Argintolac are strategie de castig daca si numai daca matricea nu contine un dreptunghi de dimensiuni $(A + 1) \times (B + 1)$ fara obstacole si care nu intersecteaza exteriorul matricei.*

Demonstratie. Cum atat Argintolac cat si Bronzolac joaca optim, ne intereseaza daca Argintolac poate alege o pozitie (i, j) din matrice in care sa plaseze robotul astfel incat Bronzolac sa nu poata sa il izbeasca ulterior de un zid (sau de bordura matricei) prin alegerea sirului S . Din Lema 3 stim ca acest lucru este posibil doar daca submatricea $M_{(i,j)}$ nu contine niciun obstacol. Observatia fundamentala este urmatoarea: Consideram doua siruri S^1 si S^2 , cu valorile caracteristice $U^i, D^i, L^i, R^i, A^i, B^i$, pentru $i \in \{1, 2\}$, atunci $M_{S^1, (i_1, j_1)}$ si $M_{S^2, (i_2, j_2)}$ sunt identice daca si numai daca $A^1 = A^2, B^1 = B^2, i_1 - D^1 = i_2 - D^2$ si $j_1 - L^1 = j_2 - D^2$. Facand abstractie de notatiile putin incalcite, ce vrea aceasta sa zica este ca pentru valori A, B fixate avem ca, indiferent de valorile U, D, L, R , multimea $T = \{M_{(i,j)} \mid M_{(i,j)} \text{ nu contine obstacole}\}$ este aceeaasi. Mai mult, avem ca T este egal cu multimea tuturor submatricelor de dimensiuni $(A + 1) \times (B + 1)$ care nu contin obstacole. Asadar, pentru valori A, B fixate, Argintolac poate gasi o pozitie (i, j) unde sa plaseze robotul astfel incat $M_{(i,j)}$ sa nu contina obstacole daca si numai daca $T \neq \emptyset$, adica daca si numai daca exista o submatrice de dimensiuni $(A + 1) \times (B + 1)$ care nu contine obstacole (si nu intersecteaza exteriorul matricei), de unde rezulta concluzia. \square

Data fiind Teorema 3, acum trebuie doar sa calculam numarul de siruri S formate din caracterele ‘A’ si ‘B’ astfel incat sa existe un dreptunghi de dimensiuni $(A + 1) \times (B + 1)$ fara obstacole in matrice. Pentru a calcula numarul final de siruri fiecare litera ‘A’ va deveni un caracter din multimea {‘U’, ‘D’} iar fiecare litera ‘B’ va deveni un caracter din multimea {‘L’, ‘R’}, deci raspunsul mai trebuie inmultit pentru fiecare sir si cu $2^{|S|}$, dar acesta este un detaliu minor pe care il lasam ca

tema cititorilor. Pentru doua numere A, B fixate, numarul de siruri S cu A litere 'A' si B litere 'B' este dat de formula $\binom{A+B}{A}$, deoarece trebuie sa alegem A dintre cele $A+B$ caractere sa fie 'A'-urile, iar restul vor fi, in mod necesar, 'B'-urile. Practic raspunsul la problema noastra acum este dat de formula:

$$\sum_{\substack{(A,B) \in K \\ A+B > 0}} 2^{A+B} \binom{A+B}{A} \quad (3)$$

Unde K este multimea perechilor (A, B) pentru care exista un dreptunghi de dimensiuni $(A+1) \times (B+1)$ in matrice care sa nu contina niciun obstacol. Pentru a obtine o complexitate finala de $O(NM)$ este suficient acum sa reusim sa determinam pentru fiecare pereche (A, B) daca $(A, B) \in K$ sau nu, pentru ca apoi putem sa calculam suma de mai sus intr-o maniera relativ brute-force, singurul amanunt fiind ca trebuie sa precaluam valorile combinatorilor folosind formula Triunghiului lui Pascal: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ (n.b. exista si alte abordari, important este doar sa o facem relativ eficient). In cele din urmare descriem determinarea multimii K in complexitate $O(NM)$.

Observam ca daca $(A, B) \in K$, atunci si $(A-1, B) \in K$, dar si $(A, B-1) \in K$ (mai putin in cazurile in care noile valori (A, B) ar deveni astfel negative). Numim procedeul prin care se pleaca de la o multime \mathcal{L} si se introduc succesiv elementele $(A, B-1)$ si $(A-1, B)$ in \mathcal{L} pentru $(A, B) \in \mathcal{L}$ pana cand nu mai apar schimbari *calculul inchiderii* (closure-ului) multimii \mathcal{L} . Inchiderea se poate calcula in $O(NM)$ reprezentand multimea \mathcal{L} ca o matrice 2D de frecventa/vizitare si parcurgand-o in ordine descrescatoare a liniilor/coloanelor, la fiecare pas setand vecinul de jos si din stanga celei curente ca vizitat daca celula curenta este vizitata. Acestea fiind date, este acum suficient sa identificam o multimea K' cu aceeasi inchidere ca K si sa ii calculam inchiderea, raspunsul final va fi apoi chiar aceasta inchidere (n.b. observati cum K este o multimea inchisa, adica egala cu inchiderea sa). Este nevoie sa calculam K' in timp $O(NM)$ pentru a avea complexitatea totala $O(NM)$.

Cele de mai sus pot suna deosebit de abstracte in prima faza, intuitia este urmatoarea: multimea K' va fi multimea perechilor (A, B) pentru care exista un dreptunghi **maximal** fara obstacole de dimensiuni $(A+1) \times (B+1)$ in matrice. Un dreptunghi (submatrice) fara obstacole se numeste maximal daca nu poate fi marit in sus/jos/stanga/dreapta nici macar cu o linie/coloana fara sa introducem obstacole in el. Desi posibila, rezolvarea cu aceasta definitie K' este usor dificila, asa ca, pentru usurinta, vom schimba definitia maximalitatii pentru a permite extinderea in **jos** a dreptunghiurilor. Cu alte cuvinte, acum numim un dreptunghi fara obstacole maximal daca nu poate fi lungit in sus/stanga/dreapta nici macar cu o linie/coloana fara sa introducem obstacole in el. Nu este greu de observat ca aceasta schimbare nu schimba inchiderea multimii K' .

Calculul dreptunghiurilor maximale acum este o problema relativ standard pentru Olimpiada Nationala:

1. Se itereaza linia de jos i a submatricei. Sa notam matricea mare cu G .
2. Se retine un sir $(h_i)_{1 \leq i \leq M}$ cu proprietatea ca $G[i, j], \dots, G[i, j - h_i + 1]$ nu contin obstacole, iar $G[i, j - h_i]$ este un obstacol. Pentru comoditate definim si $h_0 = h_{M+1} = \infty$. Practic acum un interval de coloane $[j_0, j_1]$ determina dreptunghiuri cu linia de jos i si de inaltime maxima $\min\{h_j \mid j \in [j_0, j_1]\}$. Situati la care am ajuns se mai numeste si determinarea dreptunghiurilor maximale intr-o histograma (determinata de sirul de inaltime h). Observam

ca daca se stie sirul h de la linia anterioara (i.e. $i - 1$) atunci putem imediat calcula in $O(M)$ sirul h de la linia curenta.

3. Determinarea dreptunghiurilor maxime din histograma se face astfel: Se calculeaza pentru fiecare j astfel incat $1 \leq j \leq M$ valoarea ℓ_j , care este cel mai mare indice k astfel incat $0 \leq k < j$ si $h_k > h_j$. Similar, se calculeaza pentru fiecare j astfel incat $1 \leq j \leq M$ valoarea r_j , care este cel mai mic indice k astfel incat $j < k \leq M + 1$ si $h_k > h_j$. Aceste calcule se pot face in complexitate $O(M)$ folosind doua stive de maxime, lucru pe care nu il vom explica aici. Daca aceste valori s-ar fi calculat intr-o maniera brute-force, atunci am fi obtinut complexitatea totala $O(NM^2)$, insuficienta pentru punctaj maxim.
4. Cu valorile ℓ_j, r_j precalculate, dreptunghiurile maxime se obtin ca dreptunghiuri delimitate de coloanele $\ell_j + 1, r_j - 1$ si liniile $i, i - h_j + 1$ pentru toate valorile $j \in [1, M]$. Pentru a proceda ulterior cu calculul inchiderii multimii K' , de cate ori gasim un astfel de dreptunghi vom memora dimensiunile sale (i.e. $(r_j - \ell_j - 1) \times h_j$) prin marcarea celulei de coordonate $(r_j - \ell_j - 1, h_j)$ intr-un tablou 2D de frecventa/vizitare, acelasi tablou pe care vom efectua ulterior algoritmul de inchidere.

Asa cum am amintit si mai sus, trebuie in final sa efectuam algoritmul de inchidere si sa folosim multimea K rezultata pentru calculul sumei 3. Complexitate finala $O(NM)$.

Echipa. Setul de probleme pentru această rundă a fost pregatit de:

- prof. Adrian Panaete, Colegiul “A. T. Laurian”, Botosani
- prof. Zoltan Szabó, Liceul Tehnologic “Petru Maior” Reghin / ISJ Mureş Tg. Mureş
- prof. Marius Nicoli, Colegiul Național ”Frații Buzești”, Craiova
- Andrei Constantinescu, student University of Oxford, Balliol College
- Costin-Andrei Oncescu, student University of Oxford, St. John’s College
- Bogdan Ciobanu, software engineer, Hudson River Trading
- George Chichirim, student University of Oxford, Keble College
- Tamio-Vesa Nakajima, student University of Oxford, University College
- Bogdan Sitaru, student University of Oxford, Hertford College