Concursul de Informatică „Future for Future”
Clasele XI-XII
Descrierea soluțiilor

1. Problema Lăptika

Propusă de: Rareș Ștefan Stanciu, ICHC

Subtask 1 (10 puncte)

Rețeaua clădirilor din campus poate fi privită ca un graf neorientat cu ponderi. $K = 1$ înseamnă că numai dintr-un nod pleacă căutarea lui Andi. Costurile sunt toate 1, deci problema se reduce, de fapt, la un algoritm de tip BFS. Acesta pornește din unicul centru de examen și găsește cel mai îndepărtat nod față de el.

Complexitate: $O(N)$

Subtask 2 (20 puncte)

Față de soluția anterioară diferă numărul de centre de examen. Dacă ar fi să rulăm un BFS pentru fiecare dintre cele K centre, complexitatea ar fi de $O(KN)$, ceea ce nu se încadrează în timpul de execuție propus.

Această problemă poate fi rezolvată căutând pentru fiecare nod numai distanța optimă, nu și sursa. Astfel, se observă că structura acestei cerințe este asemănătoare cu cea a BFS-ului, singura diferență fiind sursele multiple din care se începe căutarea. Vom adăuga în coadă încă de la început toate centrele de examen și vom rula BFS-ul. Astfel, vom găsi distanța minimă, fără a ne interesa și centrul aflat la distanța respectivă.

Complexitate: $O(N)$

Subtask 3 (20 puncte)

Asemenea primului subtask, $K = 1$, însă muchiile pot avea costuri mai mari. Aceasta este o aplicație clasică a algoritmului lui Dijkstra.

Complexitate: $O(N \log N)$

Subtask 4 (20 puncte)

Pentru acest subtask, este de ajuns să rulăm un algoritm Dijkstra din fiecare sursă pentru a găsi distanța minimă până la fiecare alt nod.

Complexitate: $O(N^2 \log N)$

Subtask 5 (20 puncte)

Din cauza costurilor supraunitare, trebuie folosit algoritmul lui Dijkstra cu surse multiple, în locul BFS-ului cu surse multiple (din subtaskul 2). Asemănător, trebuie adăugate toate centrele de examen de la început în *priority queue*.

Complexitate: $O(N \log N)$

O soluție de 100p se poate găsi aici: www.pastebin.com/kqRbbXVk

2. Problema Gând

Propusă de: Andrei-Rareș Tănăsescu, CNMB

Pentru a ne începe raționamentul, reducem problema la un enunț formal, folosindu-ne de următoarele observații:

- Considerăm o axă a timpului; acțiunea numărul i se va efectua la momentul i.
- **Graful gândurilor** la momentul de timp T se va nota prin $G_T = (V, E_T)$, unde V este mulțimea nodurilor, iar E_T este mulțimea muchiilor la acel moment de timp. Nodurile reprezintă numărul de ordine al elevilor, iar muchiile bidirecționale reprezintă gândurile.
- Un **grup maximal de elevi interconectați prin gânduri** este, de fapt, o componentă conexă maximală.
- Domnul Acob minte în mod cert cu privire la gândurile unei componente conexe maximale dacă aceasta conține un ciclu de lungime impară. Un graf ce nu conține niciun ciclu de lungime impară se numește bipartit, deci știm că Domnul Acob minte în mod cert dacă componenta conexă maximală nu este bipartită.

Așadar, avem un graf neorientat, inițial nul, pe care putem face următoarele operații:

1. Adăugăm o muchie între două noduri;
2. Ștergem o muchie preexistentă dintre două noduri;
3. Ne întrebăm câte componente conexe maximale nu sunt bipartite.

Subtask 1 (20 puncte)

Restricții: Există o singură acțiune de tip 3.

Deoarece avem o singură întrebare, nu ne vor interesa acțiunile de după aceasta. Pentru a rezolva aceast subtask, ne vom ține un `set`/`unordered_set` de muchii.

Pentru a răspunde la întrebare, folosim un algoritm de tip DFS care determină componentele conexe maximale și, în același timp, dacă acestea sunt bipartite. Algoritmul folosește două culori (0 și 1). Pentru fiecare nod, dacă nu a fost colorat încă, îl colorăm cu 0 și începem DFS-ul de la el. În cadrul DFS-ului, dacă nodul curent este necolorat, îl colorăm contrar nodului anterior. Dacă acesta este deja colorat la fel ca nodul anterior, atunci componenta **nu** este bipartită.

Complexitate timp: $O(N + Q)$ **sau** $O(N + Q \log Q)$**, în funcție de implementare**.

Subtask 2 (20 de puncte)

Restricții: $1 \leq N, Q \leq 1000$.

Similar ca la subtaskul 1, dar acum aplicăm DFS pentru fiecare acțiune de tip 3.

Complexitate timp: $O((N + Q) \cdot Q)$ **sau** $O((N + Q) \cdot Q \log Q)$**, în funcție de implementare**.

Subtask 3 (40 de puncte)

Restricții: Nu există acțiuni de tip 2.

Folosim păduri de mulțimi disjuncte (`Disjoint Set Union/Union Find`). Putem să modificăm DSU-ul pentru a permite deteminarea numărului de componente conexe nebipartite, folosindu-ne de paritatea lungimii drumului în graf de la un nod la „reprezentantul” mulțimii acestuia (exemplu).

Neavând operații de ștergere a muchiilor, întreg graful poate fi adăugat la structura DSU, în care operațiile de tip 1 sunt echivalente cu reuniunea mulțimilor ce conțin nodul A și nodul B, iar în cadrul operațiilor de tip 3 determinăm numărul de mulțimi reprezentate drept grafuri nebiparite.

Complexitate timp: $O(Q \cdot \alpha(N)) \approx O(Q)$.

Soluția pentru 100 puncte

O problemă care apare în urma utilizării DSU este dificultatea ștergerii de legături dintre noduri, lucru cerut în acțiunile de tip 2. Din păcate, ne este la îndemână să ștergem la un moment de timp doar muchia ultimă adăugată (o operație „undo”). Din fericire, însă, aceste operații sunt suficiente pentru a rezolva problema într-un timp destul de bun, folosind tehnica Dynamic Connectivity.

Pe axa timpului considerăm intervalele de timp în care o anumită muchie este există și momentele de timp corespunzătoare acțiunilor de tip 3. Reprezentăm axa timpului cu un arbore de intervale.

Pentru a răspunde la întrebări, inițial ne vom crea graful vid, iar apoi vom face DFS pe arbore, începând de la rădăcină și adăugând muchiile ce există pe întregul interval de timp reprezentat de nodul curent. Înainte de a ne întoarce la părintele nodului, vom face „undo” la operațiile de adăugare a muchiilor efectuate în acest interval de timp.

Pentru a permite executarea acestor operații, trebuie să renunțăm la compresia căii din DSU. Astfel, complexitatea DSU-ului devine $O(Q \log N)$.

Complexitate timp: $O(Q \log Q \log N)$.

O implementare a acestei soluții se poate găsi aici: https://kilonova.ro/submissions/390780.