Tema 1 - Algoritmi Fundamentali

Nicoleta Ciaușu

November 19, 2020

Exercitiul 1

I. Presupunem că avem un graf neorientat aciclic conex cu n muchii. Acesta este defapt un arbore. Numarul de muchii al arborelui este $n - 1$, ceea ce înseamnă că mai trebuie sa atasez o muchie arborelui. Oricum as pozitiona aceasta muchie, o sa creez un ciclu, ceea ce contrazice presupunerea.

II. Daca graful nu este conex, atunci pentru a respecta conditia de aciclitate el trebuie sa fie compus din cel puțin 2 arbori mai mici, unde numarul de muchii prezent este $n - c$, unde c este numarul de componente conexe al grafului. Trebuie sa mai adaug muchii in graf pentru a ajunge la n muchii.

Pot avea 2 situatii:

- Muchia adaugata uneste doua componente conexe diferite (si se pastreaza aciclitatea grafului), asta pana cand ajung la o singura componenta conexa (si se intampla cazul I, deci contradictie)
- Muchia adaugata uneste doua noduri din aceeasi componenta conexa ceea ce-mi creeaza ciclu (contradictie)

Din I, II $\Longrightarrow$ orice graf neorientat cu n noduri are cel mult $n - 1$ muchii.

Exercitiul 2

1. *Care sunt mulțimile de noduri si de muchii* (V, E)*? Este graf neorientat sau orientat?*

Fie f o functie bijectiva care asociaza fiecare configuratie posibila a unui cub rubik cu un numar. Numerele acestea vor fi **nodurile** grafului.

Muchiile vor fi mutarile valide de la o configuratie la alta.

Graful este **neorientat** deoarece orice mutare este inversabila (poti obtine configuratii precedente prin aplicarea mutarilor inverse asupra configuratiei curente).

2. Descrieti un algoritm eficient care gaseste o rezolvare a cubului.

Cum algoritmul **BFS** ne ofera drumul minim dintre 2 noduri, pot sa rulez o cautare in adancime pornind din configuratia nerezolvata N si am garantia ca voi ajunge in configuratia rezolvata R in minim de mutari.

Exercitiul 3

Deoarece ma intereseaza distantele minime din 1 in restul grafului, efectuez o parcurgere a grafului folosind algoritmul **BFS**. Arborele rezultat are numarul minim de muchii necesar pentru a se pastra conditia ($|V - 1|$ muchii).

In concluzie, numarul maxim de muchii care pot fi sterse este $|E| - (|V| - 1)$ (numarul existent de muchii minus numarul de muchii ale grafului).

Exercitiul 4

Avem $d|(ax + by) \implies (ax + by) \in M_d$. Daca $x = y = d$ atunci avem $d|d(a + b)$ ceea ce inseamna ca, in cel mai rau caz, $x = y = d$. Avem garantia ca putem gasi cel putin o solutie.

1. *Care sunt mulțimile de noduri si de muchii* (V, E)*? Este graf neorientat sau orientat?*

Fie graful $G(V, E)$ unde V este multimea tuturor punctelor de forma $(a \cdot c, b \cdot d)$. Fiecare nod $(a \cdot c, b \cdot d)$ il voi conecta cu muchie orientata la $(a \cdot (c + 1), b \cdot d)$ si $(a \cdot c, b \cdot (d + 1))$ iar aceste muchii vor forma multimea E.

2. Descrieti un algoritm eficient care gaseste o rezolvare a cubului.

Ruland un BFS din punctul (a, b) avem garantia ca prima suma gasita care divide d este suma minima. In plus, stim ca nu vom merge la nesfarsit (exista numar finit de noduri) deoarece in cel mai rau caz avem solutia $(a \cdot d, b \cdot d)$.

Exercitiul 5

Stiu ca nu pot adauga muchii care sa faca legatura intre doi subarbori diferiti (ar schimba arborele in urma parcurgerii).

Practic, ce incerc eu sa fac e sa adaug back edges astfel incat sa nu stric arborele.

Rezulta graful urmator, dat prin liste de adiacenta:

$$
\begin{aligned}
A_1 &= \{3, 6, 2, 4, 5, 6, 7, 8, 9, 10, 11\} \\
A_2 &= \{1, 3, 4, 5\} \\
A_3 &= \{4, 11, 1, 2, 5, 8\} \\
A_4 &= \{5, 8, 1, 3, 2\} \\
A_5 &= \{2, 1, 3, 4\} \\
A_6 &= \{7, 1, 9, 10\} \\
A_7 &= \{10, 9, 1, 6\} \\
A_8 &= \{1, 3, 4\} \\
A_9 &= \{1, 6, 7\} \\
A_{10} &= \{1, 6, 7\} \\
A_{11} &= \{1, 3\}
\end{aligned}
$$

Bonus: Numarul maxim de muchii ale arborelui este dat de suma dintre numarul de muchii ale arborelui + muchiile rezultate prin conectarea fiecarui nod la toti stramosii sai. (Nu pot conecta doi subarbori intre ei deoarece parcurgerea DFS ar genera alt arbore.)

Exercitiul 6

G trebuie sa fie arbore.

Demonstratie. Presupun ca G nu este arbore $\Rightarrow$ am un ciclu undeva. Efectuez cele doua parcurgeri, DFS si BFS:

- **DFS** o data ce va intra in ciclu, il va parcurge pe tot, iar nodul de intrare in ciclu si nodul de iesire din ciclu (ultimul nod de dinainte de a cicla) se vor afla la niveluri diferite in arbore.
- **BFS** va pune cele doua noduri pe acelasi nivel in arbore.

Cum DFS si BFS vor genera arbori diferiti, rezulta ca G trebuie sa fie arbore pentru a se indeplini conditia ceruta.

Exercitiul 7

1. *Care sunt mulțimile de noduri si de muchii* (V, E)*? Este graf neorientat sau orientat?*

Multimea de **noduri** va fi compusa din:

- Toate capetele intervalelor ($[a, b] \Rightarrow$ am nodurile a si b).
- Nodurile S si F.

Multimea de muchii va fi formata din:

- Muchii de cost 0 intre capatul end si capatul start ale 2 intervale diferite care pot fi inlantuite si sa se respecte cerinta

- Muchii de cost $b - a + 1$ intre capatul de start si capatul end al aceluiasi interval
- De S leg toate capetele stanga ale intervalelor, de F leg toate capetele dreapta ale intervalelor.

Graful este **orientat**. Problema se rezolva prin aflarea lantului de cost maxim din S in D.

Bonus: Tin punctele (x, y) ca noduri in graf. Pun costul pe muchia de la A la B ca fiind aria dreptunghiului cu coltul stanga jos in A si coltul dreapta sus in B. Pun muchie de la S la toate punctele de stanga jos, si de la F la toate punctele dreapta sus. Aflu lantul de cost maxim din S in D.

Exercitiul 8

Deoarece raspunsul unei interogari pe o muchie (x, y) imi ofera doar informatia ca exista si muchia (y, x), dar nu ne ajuta sa aflam nimic altceva despre celelalte muchii ale grafului, suntem obligati sa incercam toate muchiile $\Longrightarrow$ avem macar C_n^2 interogari orice algoritm am folosi.

Exercitiul 9

Un traseul posibil ar fi sa parcurgem eulerian arborele APM. Vreau sa arat ca suma drumului de pe APM obtinut prin parcurgere euleriana <= distanta minima * 2, iar aceasta sa fie solutia.

Lemă. Suma costurilor de pe APM <= suma pe drumul de cost minim.

Proof. Presupun prin reducere la absurd ca costul drumului minim este mai mic decat suma costurilor de pe APM. Contradictie, deoarece APM-ul ar fi putut sa fie chiar drumul de cost minim, pentru ca si el e conex. □

Cum parcurgerea euleriana trece prin fiecare muchie de 2 ori, costul total al acestei parcurgeri este 2 * suma costurilor de pe APM. Inmultind cu 2 relatia din lema, obtinem ca

*2 * Suma costurilor de pe APM <= 2 * suma pe drumul de cost minim*

echivalent cu

*suma drumului de pe APM obtinut prin parcurgere euleriana <= distanta minima * 2*

ceea ce rezolva cerinta deoarece APM-ul se poate afla in $\mathcal{O}(n^2)$ cu algoritmul lui Prim (util deoarece graful este foarte dens, avem muchie de la fiecare nod la fiecare nod).