

Tema 1 Algoritmi Fundamentali

Moroianu Theodor

12 noiembrie 2020

Exercițiul 1

Lema 1. Fie (A, E) un graf aciclic conex (un arbore), si $u, v \in A$.

Daca $(u, v) \notin E$, atunci $(A, E \cup (u, v))$ contine un ciclu.

Demonstrație. (A, E) este conex, deci exista un lant simplu de la u la v :

$$\exists (a_0, a_1, \dots, a_k) \text{ a.î. } (a_i, a_{i+1}) \in E, \forall i \in [1 \dots (k-1)].$$

Asadar, daca este adaugata muchia (u, v) se formeaza ciclul $(a_0, a_1 \dots a_k)$. □

Demonstrația Exercițiului. Fie $G = (V, E)$ un graf neorientat aciclic.

Fie K numarul de componente conexe ale lui G , si $M = |E|$.

Lema 2. $K + M$ este un invariant la adaugarea unei muchii care pastreaza aciclitatea grafului.

Demonstrație. Fie muchia adaugata muchia (u, v) .

Apar doua cazuri:

- u si v ambele fac parte din aceeasi componenta conexa.
Folosind lema demonstrata mai sus, inseamna ca s-a format un ciclu, ceea ce contrazice ipoteza ca nu se formeaza ciclul. □
- u si v se afla in componente contexe diferite.
Asadar, dupa adaugarea muchiei (u, v) , K scade cu 1 si M creste cu 1, pastrand suma constanta. □

Scotand toate muchiile, putem observa ca valoarea invariantului $K + M$ este $|V| + 0 = |V|$.

Numarul de componente conexe este cel putin 1, asadar M nu poate depasi $|V| - 1$ fara a adauga ciclul in graf. □

			1	2	3						
			4	U	5						
			6	7	8						
9	10	11	17	18	19	25	26	27	33	34	35
12	L	13	20	F	21	28	R	29	36	B	37
14	15	16	22	23	24	30	31	32	38	39	40
			41	42	43						
			44	D	45						
			46	47	48						

Figura 1: Reprezentarea unui cub rubik

Exercițiul 2

Observam ca orice stare a cubului rubik poate fi descrisa ca un sir s de lungime 48 cu valori in multimea $\{1, 2, 3, 4, 5, 6\}$.

Asadar, ne construim un graf $G = (V, E)$ in felul urmator:

- Pentru fiecare stare $s = (s_1, s_2, \dots, s_{48})$ a cubului rubik – indiferent daca este o stare valida sau nu ne construim un nod.
- Pentru fiecare rotatie posibila a cubului rubik (rotatia unei fete sau a intregului cub rubik) se adauga o muchie intre nodurile care reprezinta cele doua stari.

Rezolvarea cubului rubik in K pasi poate fi reprezentata printr-o lista $S_0 \dots S_K$ a starilor intermediare. In graful starilor, aceasta lista constituie un lant.

Observam ca oricare mutare este inversabila, asadar matricea de adiacenta este una simetrica, si graful G este neorientat.

Un algoritm eficient pentru rezolvarea cubului rubik ar putea fi:

- Aplicam algoritmi clasici de rezolvare a cubului rubik.
Avantajul acestor algoritmi este eficienta computationala, dar acesti algoritmi nu ne garanteaza minimalitatea numarului de pasi.
- Parcurgem graful plecand din nodul care modeleaza starea initiala a cubului, folosind algoritmi precum *BFS* sau *A** (desi nu stiu care euristici ar functiona pentru *A**).
- Folosim tehnica avansata de programare "*Meet In The Middle*", plecand de la ipoteza ca numarul minim necesar de pasi este mic.

Exercitiu 3

Pentru rezolvarea acestui exercitiu se presupune cunoscut algoritmul *BFS* si diferite rezultate ale acestuia.

Lema 3. *Numarul de muchii din G' nu poate fi mai mic de $|V| - 1$ fara a deconecta nodul 1 de cel putin un alt nod (altfel spus, nu exista arbori cu N noduri si mai putin de $N - 1$ muchii).*

Demonstrație. Adaugarea unei muchii scade cu cel mult 1 numarul de componente conexe. Asadar, adaugand mai putin de $|V| - 1$ muchii nu se poate ajunge la o singura componenta conexa. Implicit, o sa existe cel putin un nod intr-o componenta conexa diferita de cea al nodului 1. \square

Demonstratia Exercitiului. Fie E' multimea muchiilor obtinute prin rulara algoritmului de *BFS* plecand din nodul 1, si $G' = (V, E')$.

Teorema 1. *G' este subgraful cu numar minim de muchii al lui G care pastreaza distanta dintre i si 1, $\forall i \in [2 \dots |V|]$.*

Demonstrație. Stim ca:

- G' pastreaza distantele
Din functionearea algoritmului *BFS*, distanta minima dintre nodul 1 si nodul i poate fi obtinuta printr-un lant de muchii aflate in V' , $\forall i \in (2 \dots |V|)$. Asadar, G' pastreaza distantele.
- G' are numarul minim de muchii
Din functionearea algoritmului *BFS*, $|E'| = |V| - 1$. Din lema enuntata mai sus, nu este posibila obtinerea unui graf conex cu mai putine muchii. Asadar, E' este de cardinalitate minima.

Asadar, numarul maxim de muchii care pot fi scoase din G este $|E| - |V| + 1$. \square

\square

Exercitiu 4

Fie $V = \{ 0, 1, 2, \dots, d-1 \}$

Fie $E = \{ (i, (i + a) \bmod d), (i, (i + b) \bmod d) \mid i \in [0, 1, 2 \dots, d-1] \}$

Fie $G = (V, E)$.

Altfel spus, ne uitam la graful care are cate un varf pentru fiecare clasa de resturi modulo p , si muchii de la clasa de resturi x la clasa lui $(x + a)$ si $(x + b)$. Graful este orientat, desi in cazuri particulare (cum ar fi $d = 2$) poate fi neorientat.

Observam ca orice combinatie de numere naturale ale lui a si b corespunde unui lant in graf. Raspunsul este deci lantul minim de la nodul corespunzator lui $a + b$ si nodul corespunzator multiplilor lui d .

Un algoritm care rezolva problema este o parcurgere in latime, plecand de la nodul care reprezinta clasa lui $(a + b)$, adica nodul $(a + b) \bmod p$, si se opreste la nodul 0. Raspunsul este lungimea lantului de la $(a + b) \bmod d$ la 0.

Exercitiu 5

Stim ca muchiile neluate de parcurgerea DFS nu au voie sa fie intre doi subarbori diferiti. Folosind aceasta ipoteza, vom arata care este numarul maxim de muchii pe care le poate avea un graf cu parcurgerea DFS data, precum si ordinea in care trebuie procesate.

Dam muchiile prin liste de adiacenta, care contin toate muchiile mai putin cele enuntate mai sus:

- $Adiacenta_1 = \{3, 2, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $Adiacenta_2 = \{1, 3, 4, 5\}$
- $Adiacenta_3 = \{4, 1, 2, 5, 8, 11\}$
- $Adiacenta_4 = \{1, 2, 3, 5, 8\}$
- $Adiacenta_5 = \{1, 2, 3, 4\}$
- $Adiacenta_6 = \{1, 7, 9, 10\}$
- $Adiacenta_7 = \{1, 6, 10, 9\}$
- $Adiacenta_8 = \{1, 3, 4\}$
- $Adiacenta_9 = \{1, 6, 7\}$
- $Adiacenta_{10} = \{1, 6, 7\}$
- $Adiacenta_{11} = \{1, 3\}$

Prin tehnica de numarare a muchiilor putem deduce ca sunt maxim 24 de muchii. Ordinea in care trebuie parcursi vecinii unui nod este cea data de vectorii de adiacenta.

Demonstrație. Am verificat manual ca adaugarea oricarei alte muchii ne genereaza un arbore DFS diferit. □

Bonus

Pentru un arbore arbitrar, tragem muchie de la fiecare nod la tot stramosii sai. Astfel, raspunsul este suma adancimilor nodurilor.

Demonstrație. Optimalitatea solutiei decurge natural din:

- In arborele DFS nu exista muchii *cross edge*, adica muchii intre doi subarbori diferiti.

- Muchiile dintre un stramos si un descendent nu strica arborele *DFS* daca stramosul proceseaza descendentul dupa ce proceseaza fiul. Motivul este ca procesarea unui nod deja vizitat nu are side-effects.

□

Exercitiu 6

Stim ca BFS calculeaza arborele care minimizeaza adancimea fiecarui nod, si ca DFS incearca sa coboare cat poate in fiecare nod. Astfel, este evident ca:

- Daca inecam arborele *DFS* in graful original, nu o sa apara cross-edges, adica muchii care sa nu fie de tipul nod-stramos.
Asta se intampla pentru ca existenta unei muchii dintre doi subarbori diferiti ar presupune ca *DFS*-ul nu s-a dus pe o muchie posibila.
- Daca inecam arborele *BFS* in graful original, nu o sa apara muchii intre doua noduri la o diferenta de nivel mai mare ca 1.
Asta se intampla pentru ca existenta unei astfel de muchii ar implica existenta unui arbore cu adancimile nodurilor mai mici.

Astfel, pentru ca cele doua parcurgeri (*DFS* si *BFS*) sa ne dea acelasi arbore partial, trebuie ca:

1. Daca adaugam toate muchiile neluate in arborele *DFS/BFS*, nu apar muchii intre doi subarbori diferiti.
2. Daca adaugam toate muchiile neluate, nu apar nici muchii dintre noduri pe nivele cu diferenta mai mare ca 1.

Observatia cheie este ca oricare muchie neluata in arborele *DFS/BFS* este de tipul 1 sau 2, deci pentru ca cele doua parcurgeri sa coincidă, trebuie ca graful sa fie un arbore.

Observatie:

In demonstratia de mai sus, am presupus ca graful este conex.

Daca graful nu este conex, este suficient ca *componenta conexa din care face parte nodul 1* sa respecte restrictiile de mai sus.

Exercitiu 7

Fie $V = \{1 \dots n\} \cup \{S, D\}$.

Fie E multimea definita in modul urmator:

$$E = \{(S, i, 0) | 1 \leq i \leq n\} \cup \{(i, D, b_i - a_i) | 1 \leq i \leq n\} \cup \{(i, j, b_i - a_i) | 1 \leq i, j \leq n \wedge b_i \leq a_j\}$$

Altfel spus, E contine muchiile urmatoare:

- Muchiile dintre S si i , unde i este indicele unui interval, de cost 0.
- Muchiile dintre i si D de cost egal cu lungimea intervalului i .
- Muchiile dintre nodurile i si j , daca intervalul i poate fi ales in acelasi timp ca intervalul j , si se afla in stanga acestuia, de cost egal cu lungimea intervalului i .

Teorema 2. *Suma maximala a intervalelor care pot fi luate simultan fara sa se suprapuna (exceptand capetele) este lungimea celui mai lung drum din graful $G = (V, E)$ care incepe in S si se termina in D .*

Demonstratie. Este usor de verificat ca orice alegere valida (fara suprapuneri) de intervale fara $A = \{a_1 \dots a_k\}$ ii corespunde unui drum in reprezentarea noastra ca un graf.

Presupunem ca indicii din A sunt ordonati crescator (permutarea acestora nu schimba solutia). Observam asadar ca exista urmatoarele muchii:

- Muchia dintre S si a_1 , de cost 0.
- Muchia dintre a_i si a_{i+1} , $\forall 1 \leq i \leq k-1$, de cost egal cu lungimea intervalului i . Aceasta muchie exista pentru ca intervalele a_i si a_{i+1} nu se suprapun (daca s-ar suprapune nu ar putea fi in solutie).
- Muchia dintre a_k si D de cost egal cu lungimea intervalului a_k .

Asadar, vedem ca se poate forma lantul $S \rightarrow a_1 \rightarrow a_2 \dots \rightarrow a_k \rightarrow D$, care are lungimea egala cu suma lungimilor intervalelor din A .

Pe de alta parte, fie un lant $X = (S, x_1, x_2 \dots x_l, D)$. Observam ca lungimea acestui lant este suma lungimilor intervalelor x_1, x_2, \dots, x_l , si aceste intervale nu se intersecteaza. Asadar formeaza o solutie valida. \square

Bonus

Rezolvarea acestei probleme in cazul 2d este binecunoscuta ca fiind NP . Este echivalenta cu gasirea unui set independent de cost maxim intr-un graf.

Exercitiu 8

O sa rezolvam problema prin crearea unui algoritm adversarial care isi forteaza oponentul sa faca cel putin $\frac{N*(N-1)}{2}$ interogari.

Descrierea algoritmului:

1. Isi seteaza matricea $(Q_{ij})_{i,j < N}$ cu valorile *False*.
2. Cand primeste o intrebarea asupra unei muchii (u, v) , raspunde cu 0 (adica spune ca nu exista muchia), si seteaza $Q_{u,v} = Q_{v,u} = \text{True}$.

3. Daca a ramas cel putin o valoare de *False* in matricea Q si oponentul pretinde ca a gasit care este graful $G = (V, E)$, scoatem sau punem una din muchiile (u, v) cu $Q_{u,v} = \text{False}$, astfel incat sa se modifice graful, si spunem ca acesta era defapt graful la care ne gandeam.

Astfel, orice strategie are oponentul nostru, este fortat sa seteze toata matricea Q cu valori de *True*, adica sa puna cel putin $\frac{N*(N-1)}{2}$ intrebari.

O demonstratie mai formala, folosind teoria informatiei

Stim ca:

- Fiecare intrebare primeste un raspuns dintr-un spatiu de doua elemente.
- Numarul total de grafuri posibile este $2^{\frac{N*(N-1)}{2}}$

Asadar, cu mai putin de $\frac{N*(N-1)}{2}$ intrebari nu poate fi acoperit tot spatiul de posibilitati (cel putin doua grafuri o sa se afle in aceeasi frunza in arborele de decizie al algoritmului).

Exercitiu 9

Lema 4. Fie $G = (V, E)$ un graf cu costuri pozitive pe muchii.

Atunci, lungimea unui ciclu hamiltonian o sa fie cel putin la fel de mare ca suma lungimilor muchiilor unui APM al acestui graf.

Demonstratie. Presupunem prin absurd ca exista un ciclu hamiltonian de lungime mai mica ca lungimea oricarui APM.

Scoatem oricare muchie din ciclul hamiltonian, obtinand astfel un arbore partial, cu suma costurilor mai mica decat cea a unui APM. Contradictie. \square

Rezolvarea Exercitiului. Fie $G = (\{1 \dots n\}, \{(i, j, d_{ij}) \mid i, j \in \{1 \dots n\}\})$ unde $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Observam ca acest graf complet reprezinta corect problema, muchia de la u la v avand costul fix distanta dintre (x_u, y_u) si (x_v, y_v) .

Observam de asemenea ca drumul optim este un ciclu hamiltonian de cost minim al grafului.

Fie $G' = (V, E')$ un arbore partial de cost minim al lui G .

Observam ca parcurgerea in ordinea *DFS* (cu intrare / iesire din nod) al arborelui G' are lungimea totala $2 * \text{lungime}(G')$, care conform lemei de mai sus este mai mica sau egala decat lungimea oricarui ciclu hamiltonian.

Astfel, un itinerariu cu distanta totala cu cel mult de doua ori mai mare decat distanta optima este cel dat de parcurgerea in ordine *DFS* al unui APM al grafului.

Gasirea APM-ului poate fi facuta cu Prim's Algorithm in $\Theta(N^2)$, sau cu vrajeli cu Ainturi 2d intr-o complexitate pe care nu o stiu. \square