Tema 1 Algoritmi Fundamentali

Moroianu Theodor

17 noiembrie 2020

Exerciţiul 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.

Demonstraţie. (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 $(a_0, a_1 \dots a_k)$. □

Demonstraţia Exercitiului. 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.*

Demonstraţie. 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 ciclii.

- 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 ciclii in graf. □

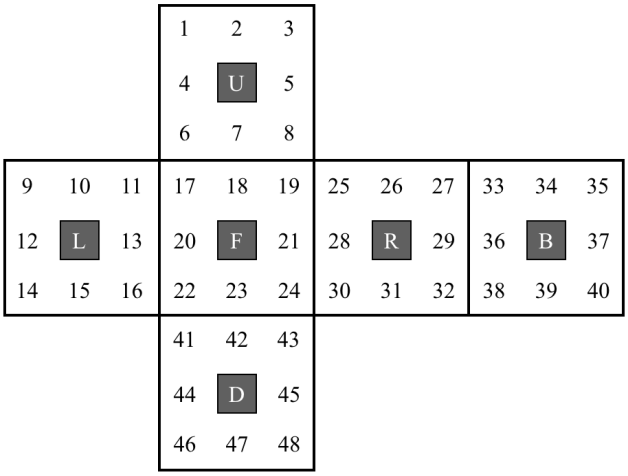

Figura 1: Reprezentarea unui cub rubik

Exerciţiul 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 $s = (s_1, s_2, \ldots, s_{48})$ a cubului rubik – indiferent daca este o stare valida sau nu ne construim un nod.
- Pentru fiecare rotatie posibila a cubului rubic (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 $S_0 \ \ldots \ S_K$ 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 rubic ar putea fi:

- Aplicam algorimi 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, folosing 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.

Cum numarul de stari din cubul rubit este marginit superior de 6^{48}, oricare din abordarile enuntate mai sus are o complexitate teoretica de $\Theta(1)$.

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).*

Demonstraţie. 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. □

Demonstratia Exercitiului. Fie E' multimea muchiilor obtinute prin rularea 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, $\forall\ i\ \in\ [2 \dots |V|]$.*

Demonstraţie. 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 functionarea 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 $|E| - |V| + 1$. □

□

Exercitiu 4

Fie $V = \{\ 0, 1,\ 2,\ \dots\ d - 1\ \}$
Fie $E = \{\ (i,\ (i\ +\ a)\ mod\ d),\ (i, (i\ +\ b)\ mod\ d)\ \ |\ i\ \in [0, 1, 2\ \dots\ d - 1]\ \}$
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 $(a + b)$ mod p, si se opreste la nodul 0. Raspunsul este lungimea lantului de la $(a + b)$ mod d la 0.

Complexitatea algoritmului este $O(|V| + |E|) = O(d)$.

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.

Demonstraţie. 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.

Demonstraţie. 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 coincida, 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 $V = \{\ S,\ D,\ b_1,\ b_2,\ \ldots\ b_n\ \}$.
Fie B lista formata din capetele dreapta ale intervalelor, in ordine crescatoare:
$B_i\ =\ b_{\sigma_i},$ unde σ este o permutare, si $B_i < B_{i+1}$.
Fie $A\ =\ \{\ (B_i,\ B_{i+1},\ 0)\ \}\ \cup\ \{\ (S,\ B_1,\ 0),\ (B_N,\ D,\ 0)\ \},$
$B\ =\ \{\ (max\_left(st_i),\ dr_i,\ dr_i\ -\ st_i)\ \},$ unde $max\_left(i)$ este cel mai mare element b_j

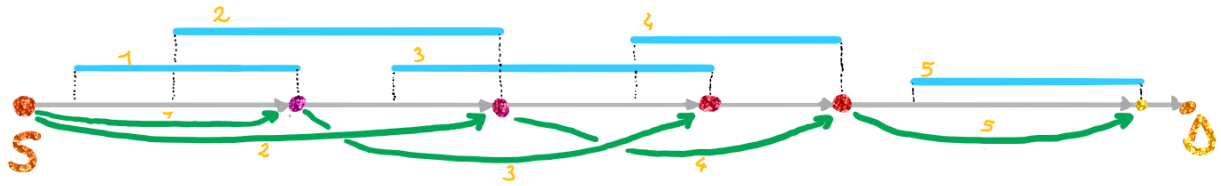

Figura 2: Exemplu de Reprezentare ca Graf

mai mic sau egal decat a_i, sau S daca nu exista niciunul.

Ne uitam acum la graful: $G = (V,\ A \cup B)$.

In desenul alaturat se poate vedea reprezentarea grafului pentru setul de muchii albastru. Muchiile in gris au costul egal cu 0, iar cele in verde au costul egal cu lungimea segmentului pe care il reprezinta.

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

Demonstraţie. Este usor de verificat ca oricarei alegeri valide (fara suprapuneri) de intervale ii corespunde unui drum in reprezentarea noastra ca un graf.
Fie multimea data in ordine crescatoare dupa capatul stanga de intervale alese $X = \{x_1,\ \ldots\ x_k\ \}$. Observam asadar ca exista urmatoarele lanturi si muchii:

- Lantul dintre S si $max\_left(a_{x_1})$, de cost 0.
- Muchia dintre $max\_left(a_{x_i})$ si b_{x_i}, $\forall 1 \leq i \leq k$, de cost egal cu lungimea intervaluli x_i.
- Lantul dintre b_{x_i} si $max\_left(a_{x_{i+1}}$ de cost 0.
- Lantul dintre b_{x_k} si D de cost egal cu 0.

Asadar, vedem ca se poate forma un lant care are lungimea egala cu suma lungimilor intervalelor din X.

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

Complexitatea algoritmului este $O(|V| + |E|) = O(N)$, dar gasirea grafului (mai precis a multimii B) implica sortarea intervalelor, deci fara sa presupunem nimic despre intervalele initiale avem o complexitate de $\Theta(N * log(N))$.

Bonus

Rezolvarea acestei probleme in cazul 2d este NP completa, asa cum explica articolul "Maximum Independent Set of Rectangles" scris de Parinya Chalermsook si Julia Chuzhoy.

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 $Q_{u,v} = Q_{v,u} = True$.
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 $Q_{u,v} = False$, 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.

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.

Demonstraţie. 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. □

Rezolvarea Exercitiului. Fie $G = (\{1 \dots n\}), \{(i, j, d_{ij}) \quad | \quad i, j \in \{1 \dots n\})$
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 G' este chiar un arbore Euclidean, reprezentand arborele minim care uneste N puncte in plan.
Observam ca parcurgerea in ordinea DFS (cu intrare / iesire din nod) al arborelui G' are lungimea totala $2 * lungime(G')$, 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 algoritmi specifici arborilor Euclideeni, cum ar fi triangulizarea Delaunay, care are o complexitate de timp de $\Theta(N * log(N))$.

□