Grafuri Tema 1

Oancea Ionut Eugen 3A5, Marius Dinu 3A5

November 2020

1.

```
rezolvare(G=(V,E), k)
if (k > | V |) then
  return "Nu"; //nu se pot crea destule partitii nici macar daca fiecare are cate un nod
nrConexe ← 0;
for (x din V) do
  if (viz[x] == 0) then
    BFS(x); //retin ce noduri au fost vizitate atunci cand fac BFS
    nrConexe++;
if (nrConexe ≤ k) then
  return "Da";
else
  return "Nu";
```

Explicatie: Numar cate componente conexe are graful. Daca sunt mai putin de k componente, le pot imparti in oricate componente conexe mai mici, deci raspunsul este afirmativ. Daca nu, partitionand graful nu pot "uni" componente conexe, deci este imposibil sa reduc numarul lor pentru a obtine k componente.

2. a. Diametrul este 2 deoarece intre oricare 2 noduri alese din graf exista un drum de lungime 2 (drumul care trece prin vecinul lor comun). Daca graful ar contine un circuit de lungime 4, atunci nodurile din acesta ar avea mai mult de un vecin comun (ar avea cate 2). Deci, este imposibil.

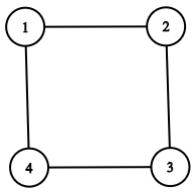

Figure 1: fiecare nod are cate 2 vecini comuni - ex.: 1 si 3 ii au pe 2 si 4

b. Conform cerintei, oricare 2 noduri au 1 vecin comun. Daca acele 2 noduri sunt vecine, inseamna ca exista muchii atat intre ele cat si intre ele si vecinul lor comun. Deci, se formeaza o 3-clica. Conform demonstratiei anterioare, fiecare nod este intr-o 3-clica impreuna cu orice vecin de-al sau. Deoarece toate nodurile din 3-clica sunt vecine, putem extinde demonstratia si afirma ca fiecare 2 vecini de-ai unui nod formeaza cu acesta o 3-clica. Alegem un nod la intamplare, v. v are d(v) vecini, iar conform demonstratiei anterioare, impreuna cu cate 2 dintre acestia formeaza 3-clici. Deci, in subgraful indus de v si vecinii sai se formeaza $d_G(v)/2$ 3-clici. Fiecare 3-clica are 3 muchii, deci numarul de muchii este $3d_G(v)/2$.

c. Alegem 2 noduri neadiacente la intamplare. Din cerinta reiese ca cele 2 noduri au un vecin comun, si din **b** reiese ca fiecare dintre cele 2 perechi de vecini fac parte din cate o 3-clica (nu pot fi toate 3 in aceeasi clica pentru ca ar trebui ca nodurile alese neadiacente sa fie vecine). Deci, cele 2 noduri alese aleator au cate 2 vecini fiecare (cel comun si vecinul din clica al fiecaruia).

3.a.

```
rezolvare(G=(V,E), altitudini[])
```

```
costuriDus[i] ← costuriIntors[i] = -1, i=1,n; //initializez costurile pentru parcurgerile din ambele sensuri
costuriDus[x] ← costuriIntors[y] = 0; //x este nodul de unde pornesc, y este nodul unde vreau sa ajung
costFinal ← ∞;
DFS_dus(G, altitudini, x, costuriDus);
DFS_intors(G, altitudini, y, costuriDus, costuriIntors, costFinal);
return costFinal;


DFS_dus(G, altitudini[], start, costuriDus[])
for (x din V) do
  if (altitudini[x] > altitudini[start] && costuriDus[x] == -1) then
    costuriDus[x] ← costuriDus[start] + α(start,x);
    DFS_dus(G, altitudini, x, costuriDus);


DFS_intors(G, altitudini[], start, costuriDus[], costuriIntors[], costFinal)
for (x din V) do
  if (altitudini[x] > altitudini[start] && costuriIntors[x] == -1) then
    costuriIntors[x] ← costuriIntors[start] + α(start,x);
    if (costuriDus[x] > -1) then
      if (costFinal > costuriDus[x] + costuriIntors[x]) then
        costFinal ← costuriDus[x] + costuriIntors[x]
    DFS_intors(G, altitudini, x, costuriDus, costuriIntors, costFinal);
```

Explicatie: Deoarece pentru a parcurge un ciclu, studentul ar trebui sa urce, apoi sa coboare (pana in nodul de start al ciclului), iar apoi sa urce din nou (pe traseul ciclului), ceea ce este imposibil deoarece dupa ce a coborat nu mai poate urca, se poate afirma ca graful practic este aciclic. Datorita acestei proprietati, o parcurgere DFS a acestuia dintr-un nod va putea calcula costurile minime pana in celelalte noduri. Fac un DFS din nodul de start spre final, apoi un DFS din nodul final spre start. Primul DFS va salva costurile minime din start pana unde se poate ajunge doar urcand, iar al doilea DFS va salva costurile minime de la final la toate nodurile unde se poate ajunge urcand (practic, costurile minime din orice nod din care se poate ajunge pana la final doar coborand). Costul minim al drumului total va fi costul minim al unei parcurgeri din start pana intr-un nod (urcand) si din acel nod pana la final (coborand). Deci, in implementarea mea, costul minim va fi minimul costurilor totale pana in si din nodurile care apar in ambele parcurgeri (acele noduri sunt posibilele varfuri ale traseului studentului). Complexitatea implementarii este $O(n+m)$ deoarece aceasta este complexitatea fiecarei pargurgeri DFS.

b. Acum, graful poate avea cicluri (noduri consecutive la aceeasi inaltime), deci nu mai exista garantia ca DFS sa calculeze costul minim. Astfel, cea mai eficienta solutie ar fi folosirea algoritmului lui Dijkstra (este posibila deoarece nu exista muchii de cost negativ), care are complexitatea $O(m+nlogn)$ (conform cursului 4).

4. a. s si u sunt in acelasi subarbore (T_s^1) in G - e, deci stergerea muchiei e nu influenteaza drumul de la s la u. Drumul minim de la s la u in G va exista si in G - e, deci cele doua drumuri sunt aceleasi si evident au acelasi cost.

b. Conform criteriului de selectie, v si t se afla in acelasi subarbore (T_s^2). Deci, exista un drum de la s la t care trece prin x_2 (radacina subarborelui). De asemenea, este garantat ca drumul de la x_2 la t (sau la s) nu trece prin e (dupa modul cum a fost construit arborele). Deci, exista un drum de la s la t care nu trece prin e care are cost mai mic decat unul care ar trece prin e. Astfel, eliminarea muchiei e nu va influenta drumul de cost minim dintre v si t - acesta va ramane neschimbat in G si in G - e.

c. Presupunem prin absurd ca exista un drum de cost minim de la s la t care trece prin mai multe muchii din F. Fie uv ultima dintre aceste muchii (din F) prin care duce drumul. Deoarece drumul selectat de mine este de cost minim, inseamna ca si d(s, u) este de cost minim. Dar, conform constructiei arborelui T_s, drumul de cost minim de la s la u (ambele fiind in A) nu trece prin nicio muchie din F (altfel ar fi si acestea in A). Deci, am ajuns la contradictie. Drumul de cost minim contine doar o muchie din F.

d. Conform **c**, drumul minim contine doar o muchie din F. Fie aceasta muchie uv. Conform **a** si

b, costurile minime ale drumurilor din s in u si din v in t in G sunt aceleasi cu cele din G - e. Deci, proprietatea este demonstrata.

e. Cum graful are muchii cu costuri strict pozitive, putem aplica algoritmul lui Dijkstra pornind din s. Conform cursului 4, acest algoritm are complexitatea $O(m + nlogn)$.