Algoritmica Grafurilor Tema 1

Iordache Iustin- Ionut **Grupa B2**
Vascan Dumitru **Grupa B2**

5 Noiembrie 2014

Problema 1

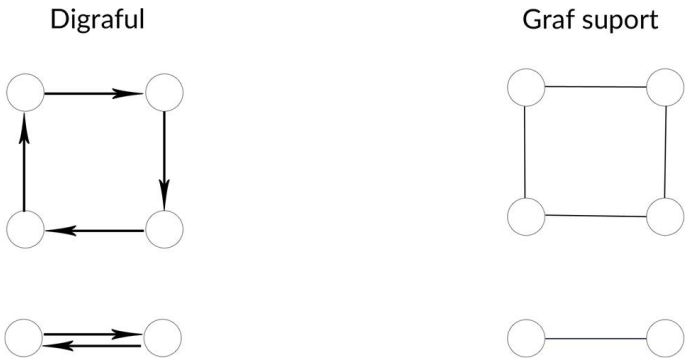
a)

Presupunem prin reducere la absurd ca M nu contine un circuit impar in D $\implies$ toate circuitele din M sunt pare, iar un circuit par contine un numar par de arce

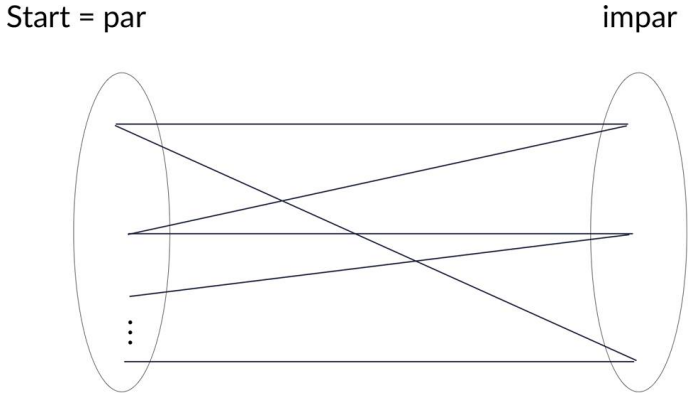
Daca mersul ar fi format din unul sau mai multe circuite pare atunci M ar fi un mers inchis par, iar daca ar exista un arc care nu apartine niciunuia dintre circuite M nu ar mai fi un mers inchis, caci arcul ar fi intr- un singur sens. Pentru un mers inchis ar trebui sa avem si o arc in sens contrar dar atunci vom forma mereu un circuit par. Deci M contine un circuit impar in D.

b)

Cum digraful este conex iar el contine doar circuite pare $\implies$ graful sau suport va fi format doar din circuite pare si din arce singure.



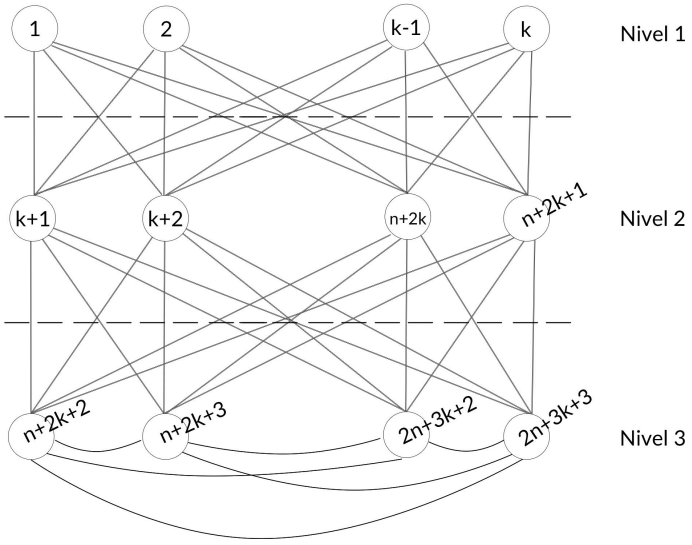
Un circuit par este format dintr-un numar par de arce $\implies$ il putem reprezenta ca un graf bipartit astfel:



Graful suport contine doar circuite pare si arce care nu apartin niciunui circuit, cum un circuit par poate fi reprezentat ca un graf bipartit, iar o arc care nu apartine niciunui circuit are obligatoriu un capat intr- o multime a bipartitiei si un capat in cealalta multime, deci aceasta arc nu creeaza nici o problema in formarea grafului bipartit $\implies$ Graful suport al unui digraf conex care nu contine ciruite impare este un graf bipartit.

Problema 2

a) Intr-un graf G algoritmul selecteaza iterativ un varf de grad minim din graful ramas si sterge acel varf si vecinii lui.Algoritmul greseste oricat de mult deoarece pentru GreedyStab(G')=1, cand G' este un subgraf complet.Algoritmul va gresi daca eliminam un intreg nivel unit de varfuri unit de G'.
Exemplu:



Pe nivelul 1 avem K varfuri , fiecare dintre ele fiind legate de toate varfuri de pe nivelul 2, iar gradul lor este n+k+1. Pe nivelul 2 avem n+k+1 varfuri , fiecare legat cu toate varfuri de pe nivelul 3, gradul lor fiind n+2k+2.Pe nivelul 3 avem un graf complet cu n+k+2 varfuri, gradul fiecaruia din ele fiind

2n+2k+2.Algoritmul alege unul dintre varfurile de pe primul nivel, si odata ce elimina varful ales si vecinii lui, gradul varfurilor de pe nivelul 1 devine egal cu 0,varfurile de pe nivelul 2 fiind eliminate deoarece sunt vecinii cu toate varfuri de pe nivelul 1,iar varfurile de pe nivelul 3 vor avea gradul n+k+1.In urmatorii k-1 pasi vor fi alese restul nodurilor de pe nivelul 1, GreedyStab(G)=k+GreedyStab(G-(nivelul 1 $\cup$ nivelul 2)).In urmatorul pas se alege oricare dintre varfuri de pe nivelul 3, iar acesta,fiind un graf complet,va returna 1.Deci, GreedyStab(G)=k+1. Deoarece nodurile de pe nivelul 1 nu sunt legate intre ele, α(G)=n+k+1. α(G)-GreedyStab(G)=n.Deci,in acest caz,algoritmul greseste oricat de mult.

b)

Vom demonstra inegalitatea prin inductie dupa numarul de varfuri. I. n=0 Cand nu avem nici un varf,nu este nici o contributie pentru multimea independenta ,si suma e tot 0. n=1 Cand avem un singur varf, GreedyStab(G)=1 si suma va fi tot 1. n=2 Avem doua cazuri :

Primul caz : cele doua varfuri sunt componente conexe , atunci GreedyStab(G)=2, si suma va fi 2.

Al doilea caz : Cele doua varfuri sunt legate printr- o muchie, GreedyStab(G)=1, si suma va fi 1 II. Presupunem ca pentru orice n¡=k , cunoastem ca inegalitatea este adevarata, si va trebuie sa demonstram aceeasi inegalitate pentru k+1. Fie x un varf de grad minim, G'=x $\cup$ N(x) ,si G" = G - G'.Algoritmul selecteaza x si elimina elementele multimii G'.Multimea G" se afla intr-unul dintre cazurile n¡=k.Folosind asta, obtinem valoarea minima determinata pentru graful G".Astfel, obtinem

$$1 + \sum_{v \in V(G")} \frac{1}{d_{G"}(v)+1} = \sum_{v \in V(G\prime)} \frac{1}{d_G(v)+1} + \sum_{v \in V(G)-G\prime} \frac{1}{d_{G"}+1}$$

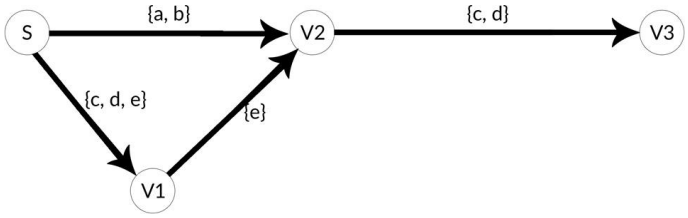
$$\geq \sum_{v \in G\prime} \frac{1}{d_G(v)+1} + \sum_{v \in V(G)-G\prime} \frac{1}{d_G(v)+1}$$

Deci, pentru n=k+1 inegalitatea este adevarata. I,II $\Longrightarrow$ inegalitatea este adevarata.

Remarcam faptul ca egalitatea se obtine cand graful G este graf complet sau grafuri complete unite printr-o muchie.Pentru aceste cazuri,GreedyStab(G) = numarul de grafuri complete,iar α(G)=numarul de grafuri complete, $\Longrightarrow$ GreedyStab(G)=α(G).

Problema 3

Algoritmul dat este o adaptare a algoritmului lui Dijkstra pentru multimi de elemente, si greseste deoarece acesta nu tine cont de elementele multimilor analizate in pasii anteriori, ci doar de cardinalii lor. La un oarecare pas, algoritmul analizeaza drumurile pana la varful ales, alegand din acestea drumul de cost minim, fara a tine cont de elemente care fac parte din celelalte drumuri, acestea putand contine elemente comune cu arcul ce duce pana la varful curent, formand astfel un drum cu cost mai mic decat cel considerat nominal de algoritm.



Algoritmul selecteaza varful S ca varful de start, seteaza cu multimea vida valoarea a acestui varf din tabloul u (u[s]= $\emptyset$), si cu 0 valoarea lui in tabloul parent(parent[s]=0), si plaseaza varful de start in multimea S(S= { s }).La urmatorul pas, algoritmul cauta varfuri care au ca drum de la ele la varful de start doar cate un singur arc, in cazul de fata aceste varfuri fiind v_1 si v_2, setand pozitiile corespunzatoare acestor varfuri din tabloul u cu multimile {c,d,e} si,respectiv, {a,b}(u[v_1]={c,d,e} si u[v_2]={a,b}).Pozitiile corespunzatoare tot acestor varfuri din tabloul parent sunt setate cu varful de start(parent[v_1]=s,parent[v_2]=s), acesta fiind varful din care porneste arcul costul careia este considerat cand se calculeaza costul drumului de la varfurile date la varful de start.Pentru toate celelate varfuri care nu au cate un singur arc care duce din ele la varful de start,in cazul de fata acest varf fiind unul singur, varful v_3, valoarea corespunzatoare acestora in tabloul u este setata cu multimea de baza X $\cup$ { # }(u[v_3]=X $\cup$ { # }), varianta alternativa valorii de infinit care se utilizeaza in algoritmul lui Dijkstra, iar in tabloul parent valorile acestora sunt setate cu -1(parent[v_3]=-1), ca la varfuri pana la care drumul inca nu este stabilit.Urmatoarea parte a algoritmului itereaza pana ce toate varfurile din V nu sunt vizitate ca potentialele surse de arce cu un cost mai mic pentru a forma drumuri mai eficiente spre alte varfuri.Algoritmul determina varful cu drum de cel mai mic cost, care in cazul de fata este varful v_2 cu costul 2(|u[v_2]|=2), il plaseaza in multimea de varfuri S(S={s,v_2 }), dupa verificand daca de la acest varf spre varful adiacent lui, v_3, nu merge un arc a.i. costul sumar al acestui arc impreune cu costul arcului care merge de la s spre v_2 nu este mai mic decat |u[v_3]| la momentul de fata.valoarea sumara a acestora este mai mica,si deci u[v_3]={a,b,c,d},iar parent[v_3]=v_2.Algoritmul alege varful v_1 ca varf cu cel mai mic cost care inca nu este in multimea S, il plaseaza in aceasta multime(S={s,v_2,v1}), verifica conditia | u[v_2] | > | u[v_1] | $\cup$ c(v_1 v_2) |,aceasta fiind falsa,si itereaza mai departe.Algoritmul selecteaza v3 ca unicul varf care nu face parte din S, il plaseaza in S,S={s,v_1,v_2,v_3 }, intra in for, insa se opreste fara a face vreo iteratie deoarece nu mai exista varfuri care nu fac parte din multimea S,se verifica conditia din while,si algoritmul isi termina rularea.valoarea finala a drumului spre varful v3 este |u[v3]|=4, cu toate ca daca drumul spre acest varf era sa fie ales ca o succesiune de varfuri v1$\rightarrow$v2$\rightarrow$v3, atunci costul drumul spre acest varf era sa fie |u[v_1] $\cup$ c(v_1 v_2) $\cup$ c(v_2 v_3)|=3. Deci, algoritmul dat nu determina drumul de cost de la un varf s spre toate celelalte varfuri in orice graf G.

Problema 4

a)

initial L={ { 1,2,3,4,5,6,7,8 } }

I. i=1;

π [1]=1;

L= { { 2,3,4,5,6,7,8 } }

L= { { 2,6 } , { 3,4,5,7,8 } }

II.i=2;

π [2]=2;

L= { { 6 } , { 3,4,5,7,8 } }

L= { $\emptyset$, { 6 } , { 3,7 } , { 4,5,8 } }

L= { { 6 } , { 3,7 } , { 4,5,8 } }

III.i=3;

π [6]=3;

L= { { 3,7 } , { 4,5,8 } }

L= { { 7 } , { 3 } , { 5 } , { 4,8 } }

IV. i=4;

π [7]=4;

L= { { 3 } , { 5 } , { 4,8 } }

L= { $\emptyset$, { 3 } , $\emptyset$, { 5 } , { 8 }, $\emptyset$, $\emptyset$, { 4 } }

L= { { 3 } , { 5 } , { 8 } , { 4 } }

V. i=5;

π [3]=5;

L= { { 5 } , { 8 } , { 4 } }

L= { $\emptyset$, { 5 } , $\emptyset$, { 8 } , { 4 } , $\emptyset$ }

L= { { 5 } , { 8 } , { 4 } }

VI. i=6;

π [5]=6;

L= { { 8 } , { 4 } }

L= { $\emptyset$, { 8 } , { 4 } , $\emptyset$ }

L= { { 8 } , { 4 } }

VII. i=7;

π [8]=7;

L= { 4 }

L= { { 4 }, $\emptyset$ }

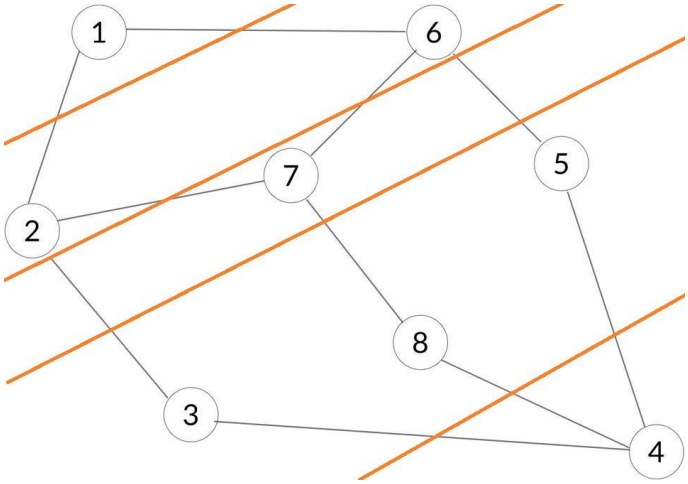
VIII. i=8;
π [4]=8;
L=$\emptyset$

π [ ]= { 1,2,6,7,3,5,8,4 }

Selectand oricare 2 elemente din tabloul π , gasim inaintea acestora in tablou un element care este legat de unu din ei doar cu o muchie, daca acestea sunt unite doar printr-o muchie, sau vid in orice alt caz.Spre exemplu : varfurile (2,6), (7,3), (5,8), etc.In tabloul π, aceste perechi de varfuri au $\emptyset$ inaintea lor.Perechile legate printr-o muchie sunt : (6,7), (5,4), (3,4), etc.Au varfurile 1, 6, si respectiv, 2 legate doar cu unul dintre ele si aflandu-se inaintea lor in tablou.

b)

Observam pe exemplu de la subpunctul a) ca algoritmul face o parcurgere BFS in functie de vecinii comuni sau necomuni ai elementelor de pe nivelul curent.



In primul pas al rularii sale, in tabloul unidimensional π de valori lexicografice este introdus varful 1 ca primul varf de la care va continua aprofundarea in graf. La urmatorul pas, in acelasi tablou sunt introduse varfuri adiacente varfului 1.Cautarea in lista dublu inlantuita a acestor varfuri necesita complexitate timp O(1). Algoritmul reia operatiunile efectuate anterior asupra varfului 1, aplicand acestea varfurilor adiacente lui,adica : insereaza varfuri de pe urmatorul nivel care sunt adiacente cu cel putin doua varfuri de pe nivelul curent, in continuare reluand operatia pentru varfuri care sunt adiacente cu doar cate un varf, pentru ambele operatiuni complexitatea timp necesara cautarilor in lista dublu inlantuita fiind O(1).Toate operatiile de mai sus sunt aplicate tuturor nivelurilor din graf, si deci, daca simulam o parcurgere din pas in pas, tragem concluzia ca algoritmul poate fi realizat in complexitate timp O(n+m), aceasta bazandu-se pe faptul ca costul dominant al algoritmului este parcurgerea BFS al grafului,complexitatea timp a careia si este O(n+m).