Simionesei Loredana-Marinela
Prisacaru Elena-Cătălina
grupa A5 anul 2

Tema 2
Algoritmica Grafurilor

<u>Problema 3</u>

a)

1. for i,j $\in$ {1,...,n},i<j do compute d(P_i,P_j)

În acest for se generează distanţa dintre oricare două puncte din U. Numărul acestor distanţe d(P_i,P_j)=n(n-1)/2, echivalent cu numărul de muchii într-un graf complet.

2. sort the n(n-1)/2 elements e=(P_i,P_j ,d(P_i,P_j)) increasing by key P_i,P_j

Această secvenţă sortează crescător elementele generate de prima linie. Este nevoie de această sortare pentru a obţine o soluţie optimă (se parcurg muchiile în ordine crescătoare ca şi la Algoritmul lui Kruskal).

3. for i=1,n do Make-set($P_{i)}$

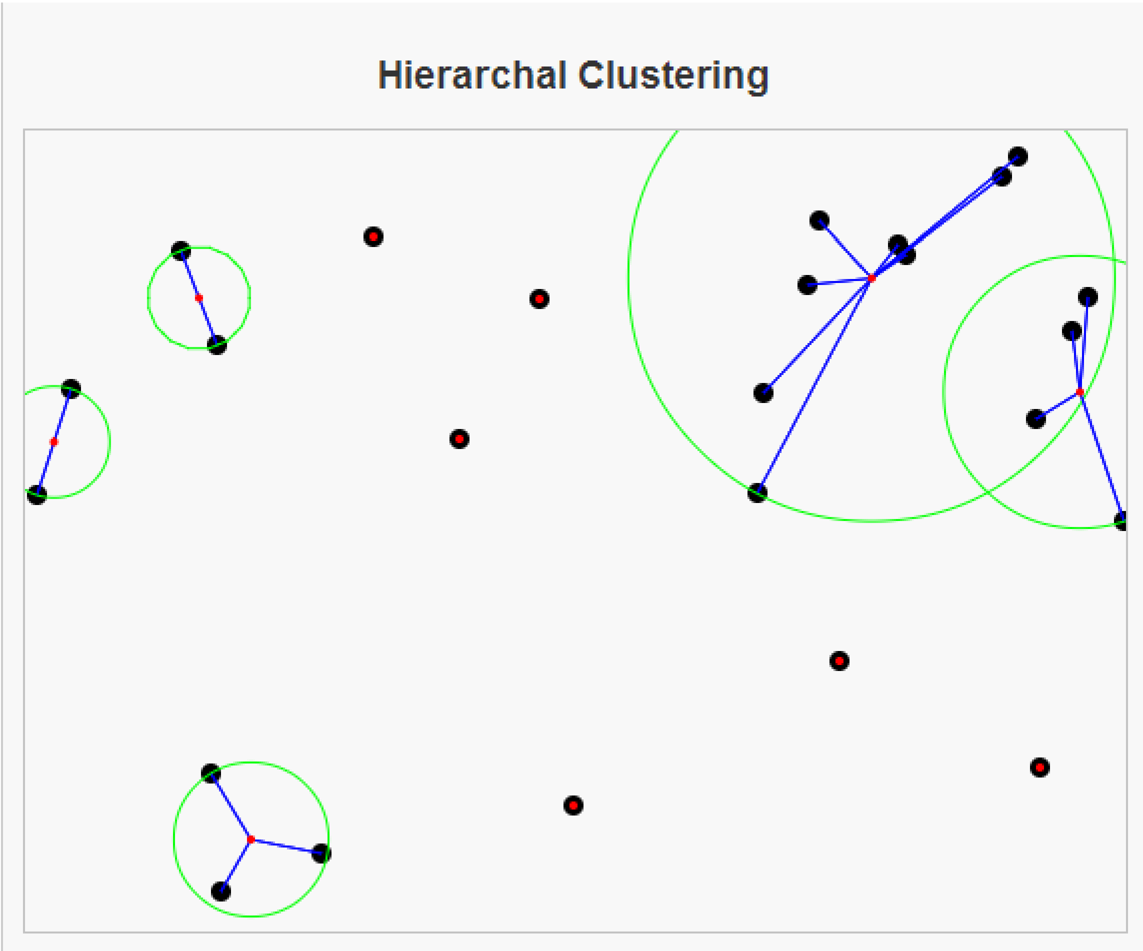
În linia 3 se construieşte partiţia lui U cu n clase, ({P_1}, {P_2},... , {P_n}), iniţializând o structură de date pentru utilizarea procedurilor union-find.

Iniţial cele n clase sunt vide.(din linia 4. added=0, index=0)

5. while added$\leq$ n-k do
6.   index=index+1; e_{index}=(P,Q,d)
7.   if Find(P) $\neq$ Find(Q) then
8.   Union(P,Q)
9.   added=added+1

Liniile 5-9 formează k clase (seturi) de puncte între care distanţa este minimă. Se ia fiecare element e=(P,Q ,d(P,Q)) generat de linia 1 şi sortat de linia 2. Pentru oricare două puncte P,Q şi distanţa dintre ele d(P,Q) care formează un astfel de element e_{index} se va verifica dacă până la momentul curent fac parte din clase diferite . Dacă se întâmplă acest lucru se va realiza uniunea dintre P şi Q i se va trece la următoare pereche de puncte.

Linia 10 va returna cele k seturi (clase) obţinute, unde în fiecare set este alcătuit de puncte din U={$P_{1,}$ e_2,... , P_n}, acestea au proprietatea că între fiecare două puncte din acelaşi set, distanţa este minimă. În fiecare clasă se formează un arbore de cost minim. Aceştia alcătuiesc o pădure. De asemenea, algoritmul se opreşte atunci când se găseşte un arbore de cost minim.



b)

Timpul de execuţie al algoritmului Kruskal-clustering pentru o mulţime de puncte U={P_1, P_2,... , P_n}, este obţinut prin însumarea tututor complexităţilor care sunt întâlnite în funcţie şi se ia complexitatea celei mai mari valori. Această sumă este obţinută din complexitatea timp pentru Union O(1), complexitatea timp pentru Find O(logn), care la rândul lor sunt înmulţite cu n^2 deoarece în bucla while se fac n^2 paşi, complexitatea timp pentru sortare O(n^2logn)=> complexitatea timp al algoritmului este O(n^2logn) deoarece sortarea este cea mai costisitoare.

Problema 1

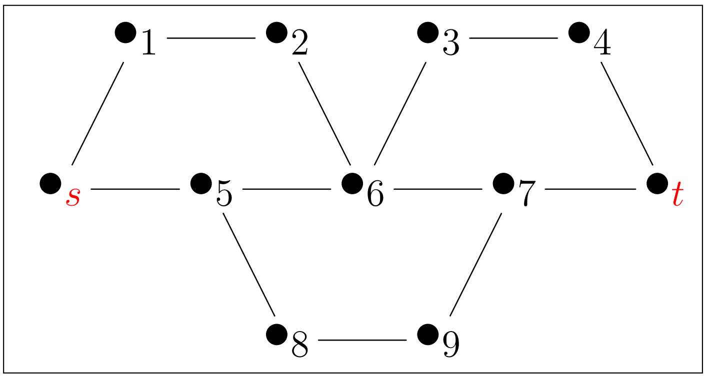
a) G=(V,E)

s,t-două vârfuri neadiacente în graful G
p_l(s,t;G)-numărul maxim de drumuri intern disjuncte (ca vârfuri) de la s la t în graful G, de lungime cel mult l(l ∈ {1,2,...,16})
k_l(s,t;G)-cardinalul minim al unei mulţimi de vârfuri, diferite de s şi t, prin îndepărtarea din graf nu mai există drumuri de la s la t de lungime cel mult l.

Vom folosi teorema lui Menger.
Notăm cu S(s,t;G) {Z|Z st-separatoare în G} (o mulţime Z ⊆ V astfel încât ∀ D ∈ D_l(s,t;G)=>V(D) ∩Z≠∅, unde D_l(s,t;G) este mulţimea de lungime cel mult l al tuturor st-drumurilor în G, adică mulţimea drumurilor de lungime cel mult l de la nodul s la nodul t).

G=(V,E) graf, vârfurile s,t $\in$ V, atunci:
Dacă $p=p_l(s,t;G)$ şi D_1, $n_2,\ldots,D_p$sunt st-drumuri intern disjuncte în G de lungime cel mult l, atunci $\forall$ Z $\in$ S(s,t;G) avem $Z \cap V(D_i) \neq \emptyset$ şi cum D_i , i=1,n ,sunt disjuncte=> $|Z|>| Z\cap \cup_{i=1}{}^{p}V(D_i)|= \sum_{i=1}^{n} |Z\cap V(D_i)|\geq \sum_{i=1}^{n} 1 = p$
Deci $\forall$ Z $\in$ S(s,t;G) $|Z|\geq$ p;
În particular: $k_l(s,t;G)\geq p_l(s,t;G)$

b)



Pentru l=4
Drumul de lungime cel mult 4 este D=(s,5,6,7,t), există un singur astfel de drum =>$p_4(s,t;G)=1$ şi $k_4(s,t;G)=1$, adică pentru a nu mai exista în graful G un drum de lungime cel mult 4, se elimină un singur nod (nodul G).
$p_4(s,t;G)=1$; $k_4(s,t;G)=1$ => inegalitatea (*) $k_l(s,t;G)\geq p_l(s,t;G)$ de la punctul a) nu este strictă

Pentru l=5
În acest caz există 4 drumuri de lungime cel mult 5:
D_1=(s,5,6,7,t)
D_2=(s,5,6,3,4,t)
D_3=(s,1,2,6,7,t)
D_4=(s,5,9,8,7,t)
Pentru că drumurile nu sunt intern disjuncte 2 câte 2 (adică $\forall$ D_i, D_j, i,j $\in$ {1,2,3,4}, D_i i $D_j$nu sunt intern disjuncte) $p_5(s,t;G)=1$ (**)
Încercăm să construim mulţimea separatoare S pe cardinal 1:
S={1} - adică eliminăm din graf nodul 1 şi toate muchiile adiacente cu acesta; dar chiar dacă eliminăm nodul 1 şi muchiile cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_4)=>S={1} nu este mulţime separatoare de cardinal 1.
S={2} - adică eliminăm din graf nodul 2 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 2 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_4)=>S={2} nu este mulţime separatoare de cardinal 1.
S={3} - adică eliminăm din graf nodul 3 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 3 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_4)=>S={3} nu este mulţime separatoare de cardinal 1.
S={4} - adică eliminăm din graf nodul 4 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 4 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_4)=>n={4} nu este mulţime separatoare de cardinal 1.
S={5} - adică eliminăm din graf nodul 5 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 5 şi muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D_3)=>n={5} nu este mulţime separatoare de cardinal 1.
S={6} - adică eliminăm din graf nodul 6 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 6 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_4)=>S={6} nu este mulţime separatoare de cardinal 1.
S={7} - adică eliminăt din graf nodul 7 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 7 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_3)=>S={7} nu este mulţime separatoare de cardinal 1.
S={8} - adică eliminăm din graf nodul 8 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 8 ţi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_3)=>S={8} nu este mulţime separatoare de cardinal 1.
S={9} - adică eliminăm din graf nodul 9 şi toate muchiile incidente cu acesta; dar chiar dacă eliminăm nodul 9 şi muchiile incidente cu acesta, rămâne drum de la s la t de lungime 5 (rămâne D_3)=>S={9} nu este mulţime separatoare de cardinal 1.
Conform celor de mai sus observăm că nu se poate construi o mulţime separatoare de cardinalul 1. Dar există mulţimi separatoare de cardinal 2 : de exemplu mulţimea S={1,5} - dacă se elimină nodurile 1 şi 5 din graf şi toate muchiile incidente cu aceste două noduri, în graful nostru nu va mai exista drum de lungime 5 de la vârful s la vârful t. Şi mulţimile de noduri {4,7}, {5,7}, {5,6} şi {6,7} sunt tot mulţimi separatoare de gradul 2=> k_5(s,t;G)=2.
Din (**) ştim că p_5(s,t;G)=1. în plus, tocmai am demonstrat că k_5(s,t;G)=2 =>inegalitatea (*) de la punctul a) este strictă, adică p_l(s,t;G)< k_l(s,t;G).

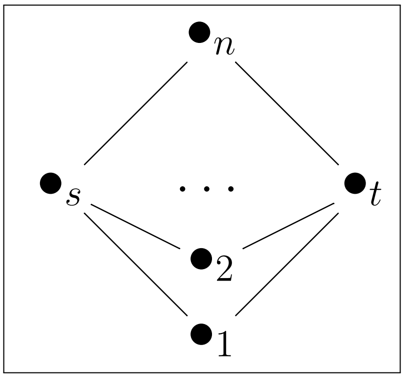
c) Pentru l=2

Presupunem iniţial că există un singur drum de lungime p_2(s,t;G)=2 între oricare două vârfuri s şi t distincte. Într-un graf G, un drum de lungime 2 între oricare două vârfuri s şi t ale grafului G arată ca în figura de mai jos.

$$\bullet_s \text{ ——— } \bullet_x \text{ ——— } \bullet_t$$

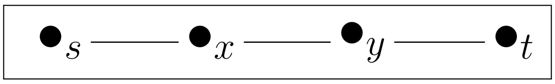
Eliminând nodul x şi muchiile incidente cu acesta din graful G, nu va mai exista un drum de lungime 2 între vârfurile s şi t=> k_2(s,t;G)=1.
Din p_2(s,t;G)=1 şi t_2(s,k;G)=1 => p_2(s,t;G)= k_2(s,t;G).
<u>Generalizare:</u>

Gândindu-ne acum că într-un graf G pot exista mai multe drumuri de lungime 2 între oricare două vârfuri s şi t din G (ca în figura de mai jos), observăm că p_2(s,t;G)=n şi k_2(s,t;G)=n=> p_2(s,t;G)= k_2(s,t;G), nefiind semnificativ numărul de drumuri de lungime 2 între oricare 2 vârfuri s şi t din G.



Pentru l=3

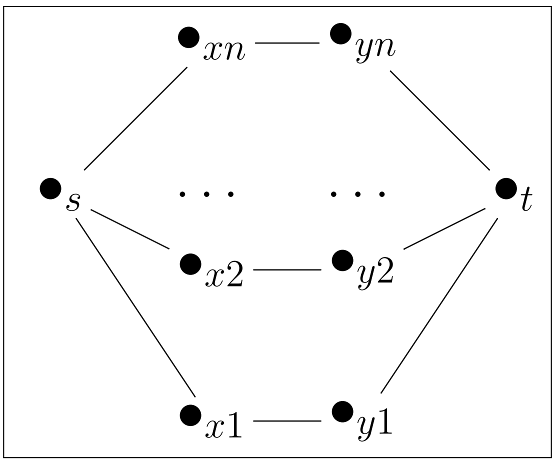
Presupunem iniţial că există un singur drum de lungime 3 între oricare două vârfuri s şi t din G=> p_3(s,t;G). Într-un graf G, un drum de lungime 3 între oricare două vârfuri s şi t ale grafului G arată ca figura de mai jos.



Eliminând nodul x sau nodul y şi muchiile incidente cu nodul eliminat din graful G, nu va mai exista un drum de lungime 3 între vârfurile s şi t=>{x} şi {y} sunt mulţimi separatoare de cardinal 1=> k_3(s,t;G)=1. Din p_3(s,t;G)=1 şi k_3(s,t;d))1 => p_3(s,t;G)= k_3(s,t;G).

Generalizare:

Într-un graf G pot exista mai multe dumuri de lungime 3 între oricare 2 vârfuri s şi t din G (ca în figura de mai jos); observăm că p_3(s,t;G)=n şi k_3(s,t;G)=n=> p_3(s,t;G)= k_3(s,t;G). Egalitatea are loc nefiind semnificativ numărul de drumuri de lungime 3 între oricare 2 vârfuri s şi t din G.



În plus, avem că:
Dacă G=(V,E) este un graf şi s, t $\in$ V, s $\neq$ t, st nu $\in$ E atunci, notând p(s,t)=numărul maxim de st-drumuri cu mulţimile de vârfuri disjuncte (cu excepţia extremităţilor), c(s,t)=cardinalul minim al unei mulţimi de vârfuri st-separatoare, avem din teorema lui Menger, p(s,t)=c(s,t).

Problema 2

a) Un cuplaj (o mulţime independentă de muchii) în graful G este o mulţime de muchii neadiacente două câte două. Cardinalul maxim al unei mulţimi independente de muchii în G se numeşte numărul de muchii independente al grafului G şi se notează cu v(G).

G=(V,E) un graf $K_{1,3}$ - free şi conex=>în G nu există niciun vârf din care să plece mai mult de două muchii, dar pleacă cel puţin o muchie; nu există între oricare 4 vârfuri din G o structură ca în Figura 1.

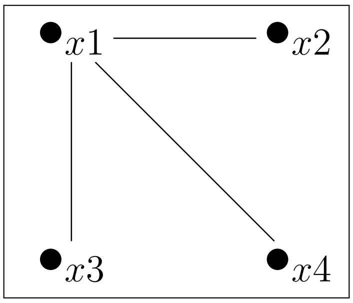
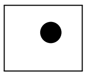


Figura 1

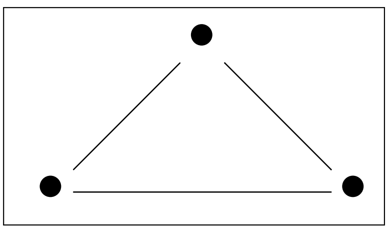
Un nod este expus în cuplaj dacă nu apare în capătul nici unei muchii din cuplaj. De fapt, graful G=(V,E) formează un arc sau o linie, adică poate avea maxim un ciclu, dar poate să nu aibă cicluri.
Pentru a construi cuplajul M de cardinal maxim în G avem 5 cazuri:
Cazul 1:

G=(V,E), |v|=1, |E|=0 (adică G este format doar dintr-un nod).

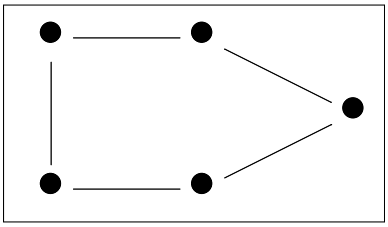


Generalizare:
(*1) În cazul 1, cuplajul M este vid, |M|=V(G)=0 şi singurul nod din graf este un nod expus relativ la cuplaj.
Cazul 2:
G=(V,E), |V|=2k+1, $\forall$ k $\in$ $\mathbb{N}^*$, |E|=|V| (adică G este format dintr-un arc cu număr impar de noduri).

1. k=1=>|V|=3, |E|=3 – În acest graf nu există un cuplaj.

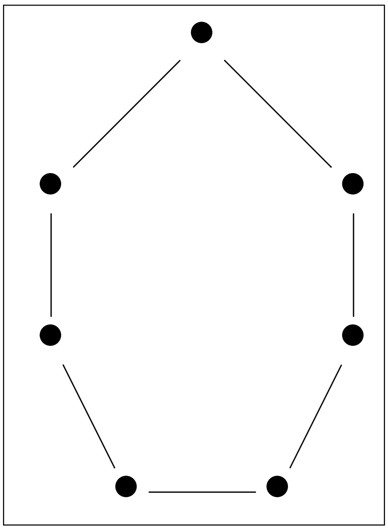


2. k=2=>|V|=5, |E|=5



   Obserăm eă există un cuplaj M dc cardinal maxim, |M|=v(G)=2 i există un singur nod expus relativ la cuplaj.

3. k=3=>|V|=7, |E|=7

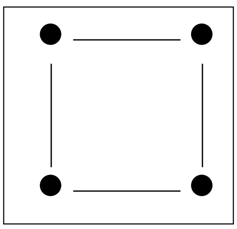


Observăm că există un cuplaj M de cardinal maxim, |M|=v(G)=3 şi există un singur nod expus relativ la cuplaj.
Generalizare:
(*2) Deci pentru cazul 2, atunci când există un cuplaj M de cardinal maxim în graful G=(V,E) va exista un singur vârf expus relativ la cuplajul M.

Cazul 3:
G=(V,E), |V|=2k, ∀ k ∈ $\mathbb{N}^*$, |E|=|V| (adică graful G este format dintr-un cerc cu număr par de noduri).

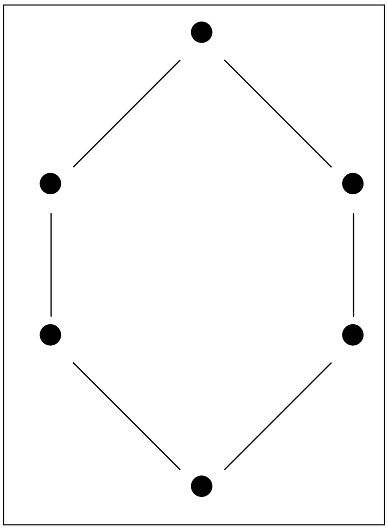
1. k=1=>|V|=2, |E|=2 – Acest graf nu există. Între oricare două noduri ale unui graf există o singură muchie.(Dacă era digraf avea sens acest caz.).

2. k=2=>|V|=4, |E|=4



Obserăm ca există un cuplaj M de cardinal maxim, |M|=v(G)=2 şi nu există niciun nod expus relativ la cuplaj (toate nodurile apar în capătul unei muchii din cuplaj).

3. k=3=>|V|=6, |E|=6



Observăm că există un cuplaj M de cardinal maxim, |j|=v(G)=3 şi nu există niciun nod expus relativ la cuplaj (toate nodurile apar în capătul unei muchii din cuplaj).

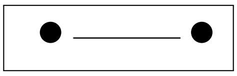
Generalizare:

(*3) Deci pentru cazul 3, atunci când există un cuplaj M de cardinal maxim în graful G=(V,E) nu va exista niciun vârf expus relativ la cuplajul M.

Cazul 4:

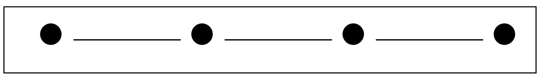
G=(V,E), |V|=2k, ∀ k ∈ ℕ*, |E|=|V|-1 (adică graful G este format dintr-o linie cu număr par de noduri).

1. k=1=>|V|=2, |E|=1



Observăm că există un cuplaj M de cardinal maxim, |M|=v(G)=1 şi nu există niciun nod expus relativ la cuplaj (toate nodurile apar în capătul unei muchii din cuplaj).

2. k=2=>|V|=4, |E|=3



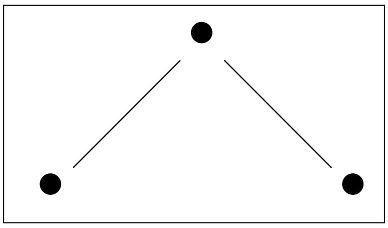
Observăm că există un cuplaj M de cardinal maxim, |M|=v(G)=2 şi nu există niciun nod expus relativ ta cuplaj (toate nodurile apar în capătul unei muchii din cuplaj).

Generalizare:

(*4) Deci, pentru cazul 4, atunci când există un cuplaj M de cardinal maxim în graful G=(V,E) nu va exista niciun vârf expus relativ la cuplajul M.

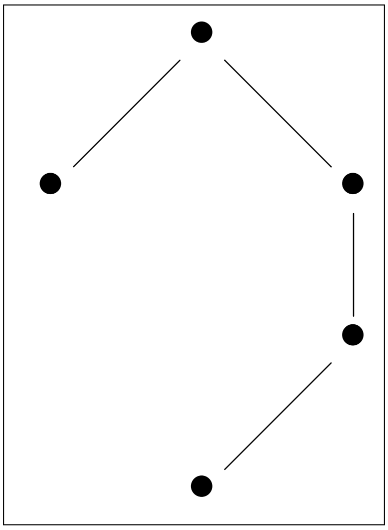
Cazul 5:
G=(V,E), |V|=2k+1, $\forall$ k $\in$ $\mathbb{N}^*$, |E|=|V|-1 (adică graful G este format dintr-o linie cu număr impar de noduri).

1. k=1=>|V|=3, |E|=2



Observăm că există un cuplaj M de cardinal maxim, |M|=v(G)=1 şi există un singur nod expus relativ la cuplaj

2. k=2=>|V|=5, |E|=4



Observăm că există un cuplaj M de cardinal maxim, |M|=v(G)=2 şi există un singur nod expus relativ la cuplaj.

Generalizare:
(*5) Deci pentru cazul 5, atunci când există un cuplaj M de cardinal maxim, în graful G=(V,E) va exista un singur vârf expus relativ la cuplajul M.

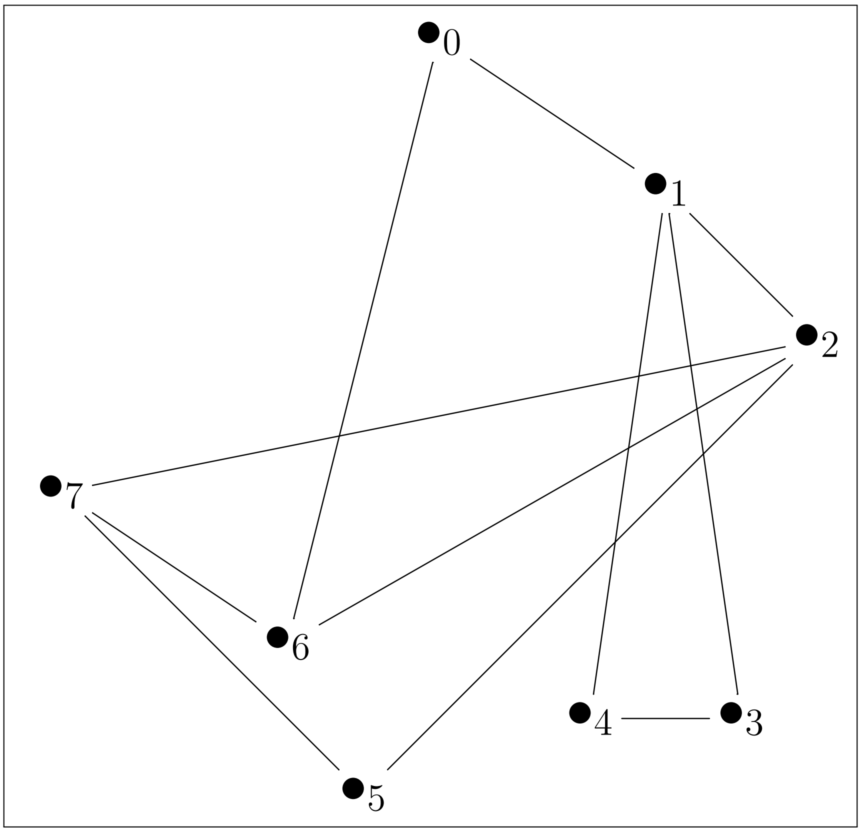
Din (*1),(*2),(*3),(*4),(*5) => dacă M este un cuplaj de cardinal maxim în G atunci există cel mult un vârf expus relativ la cuplaj.

b) Graful G este în $K_{1,3}$ de unde distingem următoarele cazuri:

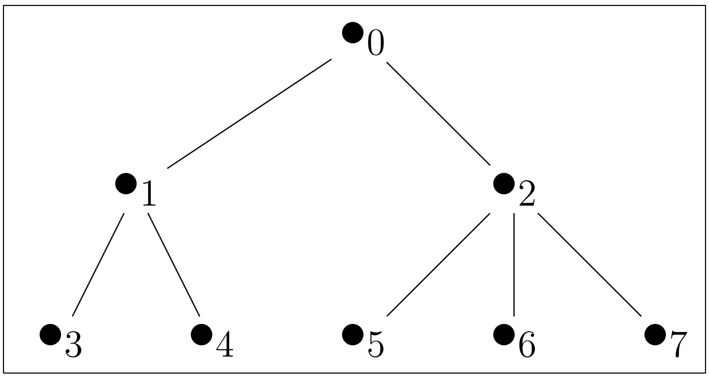
Cazul când G este linie: Parcurgând arborele DFS(depth-first search), fiecare nod poate avea 0, respectiv 1 descendent.
Cazul când G este ciclu: Parcurgând arborele DFS(depth-first search), fiecare nod poate avea 0,1, respectiv 2 descendenţi.
Pentru ambele cazuri avem arbori binari deoarece în urma parcurgerii DFS(depth-first search) avem 0,1, respectiv 2 descendenţi.

c)

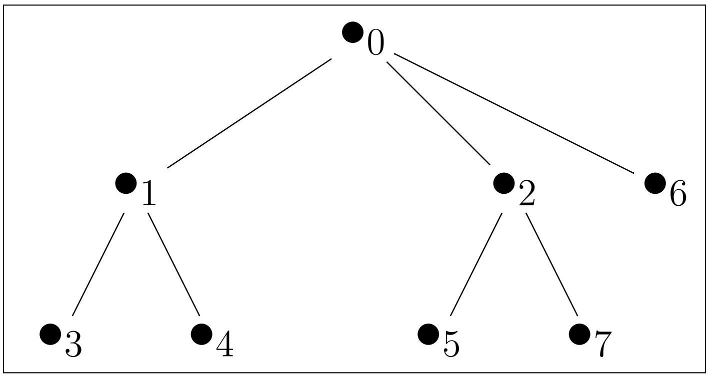
Pentru a verifica dacă un nod din T' se află în T trebuie mai întâi să construim arborele cu ajutorul parcurgerii DFS(depth-first search) în graful G. După ce s-a realizat această parcurgere se obţine un arbore. Acest arbore obţinut este notat cu T, iar T' este un arbore parţial obţinut din T şi graful G. Pentru a verifica dacă există un descendent imediat în T' al oricărui vârf diferit de la rădăcina lui T' trebuie să ne alegem un anumit nod şi să verificăm dacă nodul vecin parent(v)<-w este rădăcină în graful G, iar în caz că primul vecin al nodului nu este rădăcină, se verifică dacă există un drum de la nodul curent până la rădăcină în graful G.
De exemplu:
parent[1] = 0 deoarece 0 este unicul nod care se duce spre 1
parent[t3] = 1 deoarece avem muchie directa de la 1 la 3 în graful G
parent[3] $\neq$ 0 deoarece nu avem muchie directa de la 0 la 3 în graful G



Graful G



Arborele T



Arborele T'

d)

1. nr_de_muchii=|E|;
2. nr_de_muchii=|V|;
3. struct muchie{
4. int nod1;
5. int nod2;} e[nr_de_muchiii], M[nr_de_vârfuri/2];
6. muchie construieşte_cuplaj (int nr_de_muchii, int nr_de_vârfuri) {
7. int i, j=1;
8. for ∀ (u,v) ∈ E(G), u,v ∈ V(G){
9. e[j++].nod1=u;
10. e[j++].nod2=v;}
11. int viz[nr_de_vârfuri];
12. for (i; i<=nr_de_vârfuri;i++)
13. viz[i]=0;
14. int k=0;
15. for (j=1;j<=nr_de_muchii;j++){
16. if (viz[e[j].nod1]==0&&viz[e[j].nod2]==0&&k<=nr_de_vârfuri/2)
17. M[++k]=e[j];}
18. return M;}

Liniile 3-5 definesc o structură pentru fiecare muchie din graful G, dar şi pentru fiecare muchie din cuplajul M pe care trebuie să îl construim. (adică orice muchie este de tipul ”struct muchie”, are două câmpuri, în nod1 se va memora pentru orice muchie o extremitate, respectiv cealaltă extremitate a muchiei în câmpul nod2).
Liniile 6-18 definesc funcţia pe care o folosim pentru a construi cuplajul M. În această funcţie primul for (liniile 8-10) compune vectorul de muchii (vectorul e[nr_de_muchii]). Vom folosi un vector viz[nr_de_vârfuri] pe care îl vom iniţializa cu 0 (liniile 11-13). Liniile 14-17 construiesc cuplajul M de cardinal $[\frac{|V|}{2}]$. Vom parcurge vectorul de muchii e[nr_de_muchii] (linia 15). Apoi, vom verifica pentru fiecare muchie parcursă dacă extremităţile sale au fost vizitate; în acelaşi timp verificăm dacă cardinalul cuplajului este nr_de_vârturi/2 (linia 16). Dacă extremităţile muchiei la care s-a ajuns în momentul curent nu au fost vizitate (adică viz[e[j].nod1]==0&&viz[e[j].nod2]==0) şi dacă cardinalul cuplajului nu este nr_de_vârfuri/2, muchia la care s-a ajuns în momentul curent va fi adăugată în cuplapul M (linia 17). (Observaţie ! Trebuie ca ambele extremităţi ale unei muchii să nu fie vizitate pentru ca muchia respectivă să fie adăugată în cuplaiul M deoarece într-un cuplaj oricare două muchii sunt neadiacente, adică nu au extremităţi comune). La final se returnează cuplajul M construit în funcţia construieşte_cuplaj.
Complexitatea acestui algoritm este dată de:

- for-ul din liniile 8-10 are complexitatea O(nr_de_muchie), adică O(|E|);
- for-ul din liniile 12-13 are complexitatea O(nr_de_vârfuri), adică O(|V|);
- for-ul din liniile 15-17 are complexitatea O(nr_de_muchii), adică O(|E|);

Însumând aceste 3 complexităţi vom obţine complexitatea algoritmului:
O(|E|) + O(|V|) + O(|E|) = 2*O(|E|) + O(|V|) = O(|E|) + O(|V|) = O(|E| + |V|)