**Problema 3**

1. 1. for i,j ∈ {1,…,n},i<j do compute d(Pi,Pj)

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

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

Această secvență sortează crescător elementele generate de prima linie. Este nevoie de această sortare pentru a obține o soluție optimă (se parcurg muchiile în ordine crescătoare ca și la Algoritmul lui Kruskal).

3. for i=1,n do Make-set(Pi)

În linia 3 se construiește partiția lui U cu n clase, ({P1}, {P2},... , {Pn}), inițializând o structură de date pentru utilizarea procedurilor union-find.

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

5. while added≤n-k do

6. index=index+1; eindex=(P,Q,d)

7. if Find(P) ≠ Find(Q) then

8. Union(P,Q)

9. added=added+1

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

Linia 10 va returna cele k seturi (clase) obținute, unde în fiecare set este alcătuit de puncte din U={P1, P2,... , Pn}, acestea au proprietatea că între oricare două puncte din același set, distanța este minimă. În fiecare clasă se formează un arbore de cost minim. Aceștia alcătuiesc o pădure. De asemenea, algoritmul se oprește atunci când se găsește unarbore de cost minim.

b) Timpul de execuție al algoritmului Kruskal-clustering pentru o mulțime de puncte U={P1, P2,... , Pn}, este obținut prin însumarea tuturor complexităților care sunt întâlnite in funcție și se ia complexitatea celei mai mari valori. Această suma este obținută din complexitatea timp pentru Union O(1), complexitatea timp pentru Find O(logn), complexitatea timp pentru sortare O(n2 long n)=> complexitatea timp al algoritmului este O(n2 log n) deoarece sortarea este cea mai costisitoare.

**Problema 1**

1. G=(V,E)

S,t-două vârfuri neadiacente în graful G

pl(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}

kl(s,t;G)-cardinalul minim al unei mulțimi de vârfuri, diferite de s și 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 in G} (o mulțime z ⊆V astfel încât ∀ D ∈ Dl(s,t;G)=>V(D) ∩z≠Ø, unde Dl(s,t;G) este mulțimea de lungime cel mult l al tuturor drumurilor-st în G, adică mulțimea drumurilor de lungime cel mult l de la nodul s la nodul t).

G=(V,E) graf, vârfurile s,t ∈ V, atunci:

Dacă p=pl(s,t;G) și D1, D2,…,Dp sunt st-drumuri intern disjuncte în G de lungime cel mult l, atunci ∀ z ∈ S(s,t;G) avem z∩V(Di)≠Ø și cum Di , i=1,n ,sunt disjuncte=>|z|>| z∩ ∪i=1pV(Di) |z∩V(Di)|≥

Deci ∀ z ∈ S(s,t;G) |z|≥p;

În particular: kl(s,t;G)≥pl(s,t;G)

1. Pentru l=4:

Drumul de lungime cel mult 4 este D=(s,5,6,7,t), există un singur astfel de drum =>p4(s,t;G)=1 și k4(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).

p4(s,t;G)=1

k4(s,t;G)=1 => inegalitatea kl(s,t;G)≥pl(s,t;G) de la punctul a) nu este strictă (\*)

Pentru l=5

În acest caz există 4 drumuri de lungime cel mult 5:

D1=(s,5,6,7,t)

D2=(s,5,6,3,4,t)

D3=(s,1,2,6,7,t)

D4=(s,5,9,8,7,t)

Pentru că drumurile nu sunt intern disjuncte 2 câte 2 (adică ∀ Di, Dj, i,j ∈{1,2,3,4}, Di și Dj nu sun intern disjuncte) p5(s,t;G)=1 (\*\*)

Încercăm să construim mulțimea separatoare S de cardinal 1:

S={1} – adică eliminăm din graf nodul 1 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 1 și muchiile cu acesa, rămâne drum de la s la t de lungime 5 (rămâne D4)=>S={1} nu este mulțime separatoare de cardinal 1.

S={2} – adică eliminăm din graf nodul 2 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 2 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D4)=>S={2} nu este mulțime separatoare de cardinal 1.

S={3} – adică eliminăm din graf nodul 3 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 3 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D4)=>S={3} nu este muțime separatoare de cardinal 1.

S={4} – adică eliminăm din graf nodul 4 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 4 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D4)=>S={4} nu este muțime separatoare de cardinal 1.

S={5} – adică eliminăm din graf nodul 5 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 5 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D3)=>S={5} nu este muțime separatoare de cardinal 1.

S={6} – adică eliminăm din graf nodul 6 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 6 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D4)=>S={6} nu este muțime separatoare de cardinal 1.

S={7} – adică eliminăm din graf nodul 7 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 7 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D3)=>S={7} nu este muțime separatoare de cardinal 1.

S={8} – adică eliminăm din graf nodul 8 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 8 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D3)=>S={8} nu este muțime separatoare de cardinal 1.

S={9} – adică eliminăm din graf nodul 9 și toate muchiile incidente cu acesta; dar chiar

dacă eliminăm nodul 9 și muchiile incidente cu acesta, rămâne drum de la s la t

de lungime 5 (rămâne D3)=>S={9} nu este muțime separatoare de cardinal 1.

Conform celor de mai sus observăm că nu se poate construi o mulțime separatoare de cardinalul 1. Dar există mulțimi separatoare de cardinal 2 : de exemplu mulțimea S={1,5} – dacă se elimină nodurile 1 și 5 din graf și toate muchiile incidente cu aceste două noduri, în graful nostrum nu va mai exista drum de lungime 5 de la vârful s la vârful t. Și mulțimile de noduri {4,7}, {5,7}, {5,6} și {6,7} sunt tot mulțimi separatoare de gradul 2=> k5(s,t;G)=2.

Din (\*) știm că p5(s,t;G)=1 => inegalitatea (\*) de la punctul a) este strictă, adică kl(s,t;G)≥pl(s,t;G).

1. Pentru l=2

Presupunem inițial că există un singur drum de lungime p2(s,t;G) 2 între oricare două vârfuri s și t distincte. Într-un graf G, un drum de lungime 2 între oricare două vârfuri s și t ale grafului G arată ca în figura de mai jos.

Eliminând nodul x și muchiile incidente cu acesta din graful G, nu va mai exista un drum de lungime 2 între vârfurile s și t=> k2(s,t;G)=1.

Din p2(s,t;G)=1 și k2(s,t;G)=1 => p2(s,t;G)= k2(s,t;G).

Generalizare:

Gândindu-ne acum că într-un graf G pot exista mai multe drumuri de lungime 2 înre oricare două vârfuri s și t din G (ca în figura de mai jos), observăm că p2(s,t;G))n și k2(s,t;G)=n=> p2(s,t;G)= k2(s,t;G), nefiind semnificativ numărul de drumuri de lungime 2 între oricare 2 vârfuri s și t din G.

Pentru l=3

Presupunem inițial că există un singur drum de lungime 3 între oricare două vârfuri s și t din G=> p3(s,t;G). Într-un graf G, un drum de lungime 3 între oricare două vârfuri s și t ale grafului G arată ca figura de mai jos.

Eliminând nodul x sau nodul y și muchiișe incidente cu nodul eliminat din graful G, nu va mai exista un drum de lungime 3 între vârfurile s și t=>{x} și {y} sunt mulțimi separatoare de cardinal 1=> k3(s,t;G)=1. Din p3(s,t;G)=1 și k3(s,t;G))1 => p3(s,t;G)= k3(s,t;G).

Generalizare:

Într-un graf G pot exista mai multe dumuri de lungime 3 între oricare 2 vârfuri s și t din G (ca în figura de mai jos); observăm că p3(s,t;G)=n și k3(s,t;G)=n=> p3(s,t;G)= k3(s,t;G). Egalitatea are loc nefiind semnificativ numărul de drumuri de lungime 3 între oricare 2 vârfuri s și t din G.

În plus, avem ca:

Dacă G=(V,E) este un graf și s, t ∈ V, s ≠ t, st nu ∈ E atunci, notând p(s,t)=numărul maxim de st-drumuri cu mulțimile de vârfuri disjuncte (cu excepția extremităților), c(s,t)=cardinalul minim al unei mulțimi de vârfuri st-separatoare, avem din teorema lui Menger, p(s,t)=c(s,t).

**Problema 2**

1. Un cuplaj ( o mulțime independentă de muchii) în graful G este p mulțime de muchii neadiacente două câte două. Cardinalul maxim al unei mulțimi independente de muchii în G se numește numărul de muchii independente al grafului G si se notează cu V(G).

G=(V,E) un graf K1,3 – free și conex=>în G nu există niciun vârf din care să plece mai mult de două muchii, dar pleacă cel puțin o muchie; nu există între oricare 4 vârfuri din G o structură ca în Figura 1.

Un nod este expus la 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 constru 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).

Cazul 2:

G=(V,E), |V|=2k+1, ∀ k ∈ 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 că există un cuplaj M de cardinal maxim, |M|=V(G)=2 și există un singur nod expus relativ la cuplaj.

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

Observăm că există un cuplaj M de cardinal maxim, |M|=V(G)=3 și există un singur nod expus relativ la cuplaj.

Cazul 3:

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

1. k=1=>|V|=2, |E|=2 – În acest graf nu există un cuplaj
2. k=2=>|V|=4, |E|=4

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

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

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

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 și nu există niciun nod expus relativ la cuplaj (toate nodurile apar în capătul unei muchii din cuplaj).

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

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

Cazul 5:

G=(V,E), |V|=2k+1, ∀ k ∈ 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 și există un singur nod expus relativ la cuplaj

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

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

Generalizare:

(\*1) În cazul 1, cuplajul M este vid, |M|=V(G)=0 și singurul nod din graf este un nod expus relativ la cuplaj.

(\*2) În 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.

(\*3) În 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.

(\*4) În 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.

(\*5) În 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.

1. Graful G este in K1,3 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 poae avea 0,1, respectiv 2 descendenți.

Pentru ambele cazuri avem arbori binari deoarece în urma parcurgerii DFS(depth-first search) avem 0,1, respectiv 2 descendenti.

1. Pentru a verifica dacă un nod din T’ se află în T trebuie mai întâi să contruim arborele cu ajutorul parcurgerii DFS(depth-first search) în graful G. După ce s-a realizat această parcurgere se obține un arbore. Acest arbore obținut este notat cu T, iar T’ este un arbore parțial obținut din T și 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 și să verificăm dacă nodul vecin parent(v)<-w este rădăcină în graful G, iar în caz că primul vecin al nodului nu este rădăcina, 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[3] = 1 deoarece avem muchie de la 1 la 3 în graful G

parent[3] ≠ 0 deoarece nu avem muchie de la 0 la 3 în graful G

1. nr\_de\_muchii=|E|;
2. nr\_de\_muchii=|V|;
3. struct muchie{
4. int nod1;
5. int nod2;} e[nr\_de\_muchii], M[nr\_de\_vârfuri/2];
6. muchie construiește\_cuplaj (int nr\_muchii, int nr\_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\_vârfuri/2)
17. M[++k]=e[j];}
18. return M;}

Liniile 3-5 definesc o structură pentru fiecare muchie din graful G, dar și 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 funcția pe care o folosim pentru a construi cuplajul M. În această funcție 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 inițiaza cu 0 (liniile 11-13). Liniile 14-17 construiesc cuplajul M de cardinal [. Vom parcurge vectorul de muchii e[nr\_de\_muchii] (linia 15). Apoi, vom verifica pentru fiecare muchie parcursă dacă extremitățile sale au fost vizitate; în același timp verificăm dacă cardinalul cuplajului este nr\_vârfuri/2 (linia 16). Dacă extremitățile muchiei la care s-a ajuns în momentul curent nu au fost vizitate (adică viz[e[j].nod1]==0&&viz[e[j].nod2]==0) și dacă cardinalul cuplajului nu este nr\_vârfuri/2, muchia la care s-a ajuns în momentul curent va fi adăugată în cuplajul M (linia 17). (Observație ! Trebuie ca ambele extremități ale unei muchii să nu fie vizitate pentru ca muchia respectivă să fie adăugată în cuplajul M deoarece într-un cuplaj oricare două muchii sunt neadiacente, adică nu au extremități comune). La final se returnează cuplajul M construit în funcția construiește\_cuplaj.

Complexitatea acestui algoritm este dată de:

* for-ul din liniile 8-10 are complexitatea O(nr\_de\_muchii), 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 complexitaea O(nr\_de\_muchii), adică O(|E|);

Însumând aceste 3 complexități vom obține complexitatea algoritmului:

O(|E|) + O(|V|) + O(|E|) = 2\*O(|E|) + O(|V|) = O(|E|) + O(|V|) = O(|E + V|)