Tema 2

Renghiuc Bianca Elena, Chichirău Claudiu-Constantin 2A2

18.11.2022

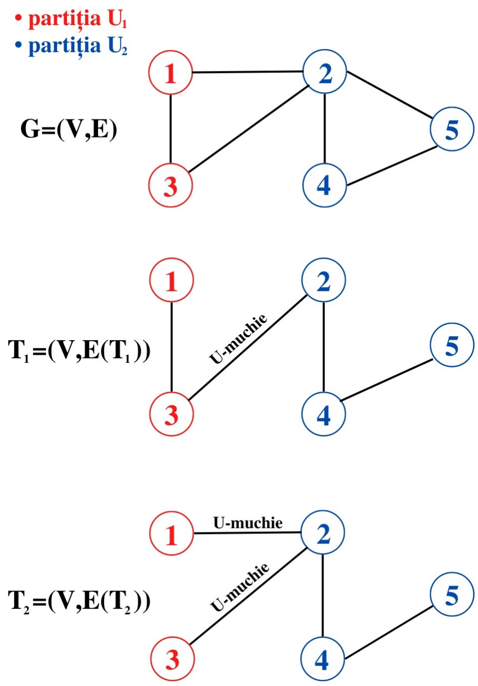
1 Execiţiul 1

a) Fie graful conex $G = (V, E)$ și $T = (V, E(T))$ un arbore parțial al grafului G. O $U - muchie$ este o muchie $uv \in E$ astfel încât $u \in U_i$, $v \in U_j$, $i \neq j$, unde $U = (U_1, U_2, ..., U_p)$ este o partiție de cardinal p a lui V.

Presupunem că graful G are o partiție $U = (U_1, U_2, ..., U_p)$ de cardinal p ce conține cel puțin $(p - 1)$ $U - muchii$ și demonstrăm prin inducție că există o partiție $U = (U_1, U_2, ..., U_p, U_{p+1}))$ de cardinal $(p + 1)$ care conține cel puțin p $U - muchii$.

Notăm cu $U(p)$ o partiție de cardinal p ce conține cel puțin $(p-1)$ $U-muchii$.

Etapa de verificare:
n=2 $\Rightarrow$ Avem 2 partiții U_1 și U_2 ale grafului G. Datorită faptului că graful G este conex și T este un arbore parțial (un graf conex aciclic) va exista cel puțin o $U - muchie$ între cele două partiții U_1 și U_2 ale arborelui parțial $T = (V, E(T))$.
$\Rightarrow U(2)$ *este adevărată.*



Etapa de demonstrație:
Presupunem că $U(p)$ este adevărată:
U(p): Fie un arbore parțial $T = (V, E(T))$ al grafului $G = (V, E)$ și $U = (U_1, U_2, ..., U_p)$ o partiție de cardinal p ce conține cel puțin $(p-1)$ $U-muchii$. Datorită faptului că graful G este conex și T este un arbore parțial (un graf conex aciclic) vor exista cel puțin $(p-1)$ $U-muchii$ ce vor lega cele p partiții ale arborelui parțial $T = (V, E(T))$. **(1)**

Și demonstrăm că $U(p+1)$ este adevărată:
U(p+1): Fie un arbore parțial $T' = (V, E(T))$ al grafului $G = (V, E)$ și$U = (U_1, U_2, ..., U_p, U_{p+1}))$ o partiție de cardinal $(p+1)$ ce conține cel puțin (p) $U-muchii$.
(1) $\Rightarrow$ Arborele parțial T cu p partiții conține cel puțin $(p-1)$ $U-muchii \Rightarrow$ Un arbore parțial T' cu $(p+1)$ partiții va conține cel puțin (p) $U-muchii$.

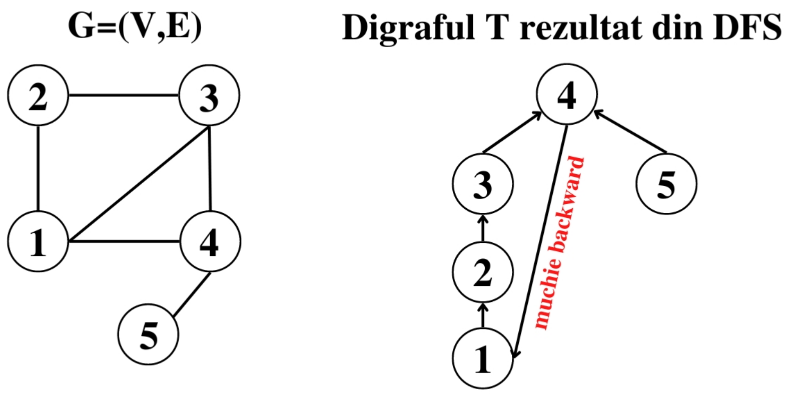
b) Fie $G = (V, E)$ un graf conex, $U = (U_1, U_2, ..., U_p)$ o partiție de cardinal p a lui V și s arbori parțiali mutual dijuncți pe muchii. După cum am demonstrat la subpunctul anterior un arbore parțial $T = (V, E(T))$ va conține cel puțin $(p-1)$ $U-muchii$. Astfel, 2 arbori parțiali disjuncți pe muchii $T_1 = (V, E(T_1))$ și $T_2 = (V, E(T_2))$ vor conține fiecare un număr de $(p-1)$ $U-muchii$. Ceea ce înseamnă că în graful G care are 2 arbori parțiali vor exista cel puțin $2*(p-1)$ $U-muchii$.

Generalizând, ajungem că într-un graf G cu s arbori parțiali dijuncți mutuali pe muchii vom avea cel puțin $s*(p-1)$ $U-muchii$.

2 Exercițiul 2

a) Știm că în digraful T există drumuri de la orice nod descendent u la un nod strămoș v (proprietatea muchiilor forward de la punctul 'i').

Fie $\{vu\}$ o muchie backward de la orice strămoș v la un descendent u. Din faptul că exista drum de la orice nod descent la toate nodurile strămoș ale sale, putem deduce că exista drum de la u la v. Prin adăugarea arcului $\{vu\}$ în digraful T se va forma astfel un circuit unic C_{vu}.



În exemplul de mai sus există drumul $1 \to 2 \to 3 \to 4$. Adăugând muchia backward $4 \to 1$ se poate forma circuitul C_{41}. De asemenea, tot în acest exemplu, putem observa drumul $1 \to 2 \to 3$, cărui i se poate adăuga doar muchia backward $3 \to 1$ pentru a forma circuitul C_{31}.

b)
```
procedure NEVIZITARE (u)
begin
        vizitat <- false
        for (w apartine B(u))
                delete w from B(u)
                if (vizitat(w)!=0) then
                    NEVIZITARE(w)
end NEVIZITARE


procedure CIRCUIT (v)
begin
        f <- false
        vizitat(v) <- true
        for (w apartine A(v))
                if (w=v) then
            //am format circuit
                        f <- true
                else
                    if (vizitat(w) = 0) then
                            if (CIRCUIT(w) != 0) then
                                f <- true
            if (f = 1) then
                NEVIZITARE(v)
            else
                for (w apartine A(v))
                        if (v nu apartine B(w)) then
                            put v on B(w)
end CIRCUIT

j<-1
s<-vertex[1]
while (j <= n)
        if (A[s] != multimea vida ) then
                for (i apartine V)
                        vizitat(i) <- false
                        B(i) <- multimea vida
                CIRCUIT(s)
        if (j!=n)
            s=vertex[j++]
```

Algoritmul acceptă un graf $G = (V, E)$ reprezentat printr-o listă de adiacentă A(v) pentru fiecare $v \in V$. Lista $A(v)$ conține nodul $\{u\}$ dacă și numai dacă există muchia $\{vu\} \in E$. Cunoaștem o parcurgere DFS din algoritmul dat (păstrată în vectorul vertex), de care ne vom folosi în algoritm pentru a forma circuite ce incep dintr-un nod curent din vectorul vertex.

Pentru a evita duplicarea circuitelor, un nod v este vizitat atunci când este adăugat la un drum elementar care începe din s. Acesta rămâne vizitat atât timp cât fiecare drum de la v la s intersectează drumul elementar curent într-un nod, altul decât s.

Algoritmul continuă prin construirea drumurilor elementare din v. Nodurile din drumul elementar curent sunt păstrate într-o stivă. Un nod este adăugat la un drum elementar printr-un apel al procedurii 'CIRCUIT' și este șters la întoarcerea din acest apel.

Când un nod v este adăugat la un drum, acesta este vizitat prin setarea $vizitat(v) = true$, astfel încât v nu poate fi folosit de două ori pentru acelasi drum. Oricare două aplelari ale functiei 'NEVIZITAT' cu parametrul v sunt separate fie printr-o ieșire a unui circuit nou, fie printr-o revenire la procedura principală.

c) Fie digraful T rezultat în urma parcugerii DFS și a orientării muchiilor conform cerinței date. Algoritmul caută primul nod din care pleacă o muchie backward. Dacă din nodul v pornește o muchie backward $\{vu\}$, vom încerca să formăm un ciclu, parcurgând nodururile din ciclul C_{vu} dar, ne oprim în momentul în care găsim primul nod p vizitat deja. Dacă nodul p este chiar nodul v din care am plecat atunci am gasit ca legătură ciclul C_{vu}. În caz contrar, am gasit drumul $v \to p$. Astfel, pornind dintr-un nod v ce are o muchie backward până la primul nod vizitat se va forma de fiecare dată un drum sau un circuit, numit legătură.

d) Luăm nodurile în ordinea parcurgerii DFS. În cazul în care primul nod din parcurgerea DFS nu este incident cu o muchie backward nu putem forma nici o legătură, fie ea drum sau ciclu. Astfel, algoritmul caută primul nod din parcurgerea DFS care să fie incident cu o muchie backward. În momentul când găsim nodul $\{v\}$ ce este incident cu o muchie backward $\{vu\}$, vom începe parcurgerea circuitlui C_{vu} până când vom găsi primul nod deja vizitat. Știm că nodul $\{v\}$ este primul nod incident cu o muchie backward, ceea ce înseamnă că toate nodurile de pe circuitul C_{vu} sunt încă nevizitate. Dat fiind acest caz, primul nod deja vizitat pe care îl găsim pe circuitul C_{vu} este chiar nodul $\{v\}$ din care am plecat. Astfel, prima legătură din digraful T poate nu poate fi decât un circuit.

e) Graful G=(V,E) este $2-conex$ dacă și numai dacă respectă una dintre cele două condiții de mai jos:

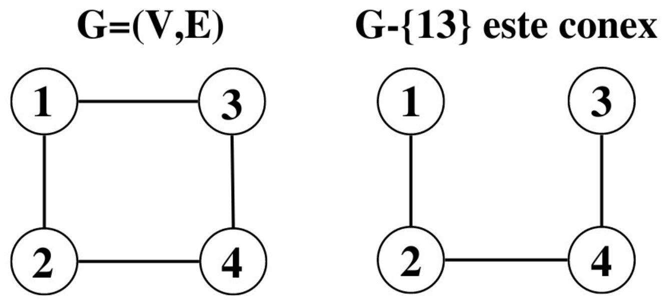
1) $|V| = p$ *și* $G = K_p$
2) $|V| \geq p+1$ și G nu are o mulțime de articulație de cardinal $< 2 \Rightarrow$ G nu are o mulțime de articulație de cardinal $= 1$.

(1) $\Rightarrow$ Presupunem că Graful $G = (V, E)$ ar fi 2-conex. $\Rightarrow |V| = 2$. Noi știm că $\delta(G) \geq 2$, ceea ce este imposibil pentru că un graf cu două noduri nu poate avea niciodată $\delta(G) \geq 2$. (Contradicție)

(2) $\Rightarrow$ Presupunem prin RA că graful $G = (V, E)$ nu este 2-conex, adică $|V| < 3$ sau G are o mulțime de articulație de cardinal < 2 (G poate fi deconectat prin ștergerea unui singur nod).

Dacă $|V| < 3 \Rightarrow |V| = 2$ *sau* $|V| = 1$, nu se respectă proprietatea $\delta(G) \geq 2$.

În schimb, pentru a vedea dacă graful G poate fi deconectat prin eliminarea unui singur nod știm că $\delta(G) \geq 2 \Rightarrow \forall u \in V, G - \{u\}$ este conex. (Contradicție)
$\Rightarrow$ Graful $G = (V, E)$ este 2-conex știind că $\delta(G) \geq 2$ și că descompunerea grafului conține doar o singură legătură care este circuit.



3 Exercițiul 3

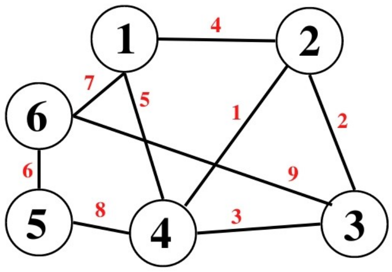
a) Fie graful $G = (V, E)$, $T_s = (V_s, E_s)$ un arbore obținut după un număr de iterații ale algoritmului lui Kruskal și $e_1 = \{u_1 v_1\}$ și $e_2 = \{u_2 v_2\}$ doua muchii din $E \backslash E_s$ astfel încât u_1, $u_2 \in V_s$ și v_1, $v_2 \notin V_s$.

La fiecare pas al buclei 'while' din algoritmul lui Kruskal vom alege muchia de cost minim, astfel încât aceasta să nu formeze un ciclu în arborele format. Funcția de cost c este injectivă, deci algorimul lui Kruskal va alege muchiile de cost strict crescător.

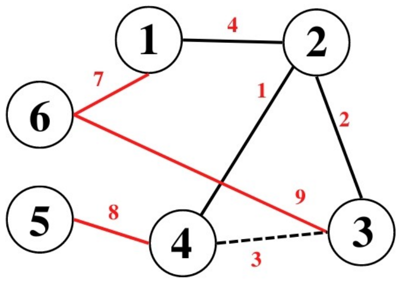
În acest caz singurele muchii care nu vor fi alese cu costuri mai mici decât muchia de cost maxim din arbore sunt acele muchii $\{xy\}$ care ar fi format cicluri, deci nodurile $x, y \in V_s$. De unde rezultă că, orice muchie $\{v_i u_i\}$, cu $v_i \in V_s$ și $u_i \notin V_s$, va avea costul mai mare decât muchia de cost maxim din arbore, muchia e, (deci implicit mai mare decât muchia de cost maxim de pe drumul de la u_1 la u_2, unde $u_1, u_2 \in T_s$).

$\Rightarrow c(e) < c(e1)$ *și* $c(e) < c(e2)$, unde e $= u_1 v_1$, $e_2 = u_2 v_2$, u_1, $u_2 \in V_s$, v_1, $v_2 \notin V_s$.

G=(V,E)



Arborele T_s=(V_s, E_s)



În exemplul de mai sus, se poate observa că arborele format are nodurile $\{1, 2, 3, 4\}$. Singura muchie peste care am sărit din cauza că ar fi format ciclu este muchia $\{34\}$ (de cost 3) care are cost mai mic decat muchia de cost maxim din arbore (cost 4). Restul muchiilor care au un nod incident care nu apratine aborelui (nodurile 6 și 5) formează împreună cu noduri din arbore muchiile $\{61\}$(cost 7), $\{63\}$(cost 9), $\{54\}$(cost 8) ce au cost mai mare decat muchia de cost maxim din T_s.

b) Fie graful $G = (V, E)$, $T_s = (V_s, E_s)$ arborele cel mai mare din pădurea obținută după un număr de $(|V| - 2)$ iterații ale algoritmului lui Prim și $e_1 = \{u_1v_1\}$ și $e_2 = \{u_2v_2\}$ doua muchii din $E \backslash E_s$ astfel încât u_1, $u_2 \in V_s$ și v_1, $v_2 \notin V_s$.

Plecăm cu algoritmul lui Prim dintr-un nod x și vom alege muchia de cost minim incidentă cu el. La fiecare pas al buclei 'while' vom alege muchia de cost minim incidentă cu unul dintre nodurile adăugate până acum în arbore.

Presupunem prin RA că $c(e) > c(e1)$. În acest caz, algoritmul lui Prim ar fi putut alege muchia de cost minim e_1 în locul muchiei e ce are un cost mai mare. Dar cum $e_1 \notin T_s$, ajungem la contradictie.
$\Rightarrow c(e) < c(e1)$
Analog si pentru muchia e2.
Astfel, muchia $\{e\}$ trebuie să aibă un cost mai mic decât măcar una din muchiile e_1 și e_2.
$\Rightarrow c(e) < c(e1)$ sau $c(e) < c(e2)$

4 Exercițiul 4

a) Fie digraful aciclic $G = (V, E)$. Pentru a forma acoperirea prietenească a digrafului aciclic G căutăm un drum direct de la un nod x la un nod y, $x, y \in V$, iar în cazul în care există acest drum, vom trasa arcul $\{xy\}$. Digraful inițial, fiind aciclic, poate admite o ordonare topologică. Astfel, în momentul în care vom trasa arcul $\{xy\}$ nu se va modifica gradul intern al lui x, digraful nou obținut păstrându-și proprietatea de aciclicitate.

Pentru a demonstra general proprietatea vom face acelaşi procedeu pentru toate drumurile existente în digraful G. De fiecare dată gradul intern al nodului cu care este incident arcul trasat nu se modifică. În acest caz, la fiecare pas se păstrează proprietatea de ordonare topologică, deci putem ajunge la concluzia că noul digraf G' este aciclic.

b) O multime S $\subseteq$ în V este stabilă în G' dacă și numai dacă nu există niciun arc direct de la un nod spre alt nod din mulțimea S. Cunoaștem faptul că o acoperire prietenească introduce în digraf un nou arc între nodurile ce nu sunt adversare. Cum mulțimea stabilă S nu are arce directe între nodurile sale iar acoperirea prietenească G' are arce între toate nodurile ce nu sunt adversare (între nodurile adversare nu există arce). Deducem faptul că submulțimea S al lui V este mulțime stabilă în digraful G' dacă și numai dacă aceasta este o adversitate.

c) Știm din cerință că acoperirea prietenească creează arce între oricare două noduri u și v dacă există un drum direct de la u la v. Construim în digraful G drumul format din nodurile x_1, c_1, x_2, ..., x_i, c_i, x_{i+1}, ..., c_{p-1}, x_p, unde c_k este mulțimea vidă sau un drum direct simplu de la x_k la x_{k+1} (dar fară capete). Prin formarea acoperirii prietenești fiecare $c_k \neq \emptyset$ va fi înlocuit cu $\emptyset$ formându-se astfel un arc direct de la x_k la x_{k+1}. Așadar, în G' se formează drumul x_1, x_2, ..., x_i, x_{i+1}, ..., x_p.