Tema1

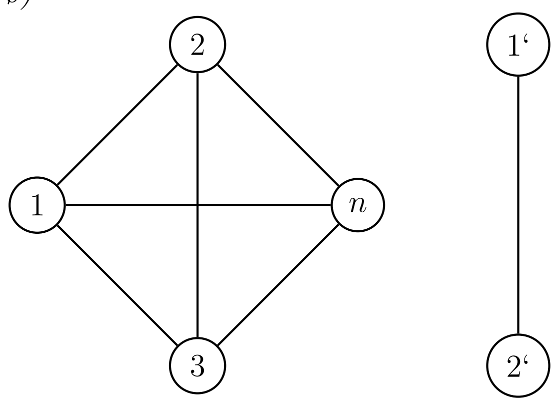
Tiganus Ionel, Grigore Valerian, grupa A3

November 2022

Exercitiul 1

a) Avem 2 cazuri extreme:

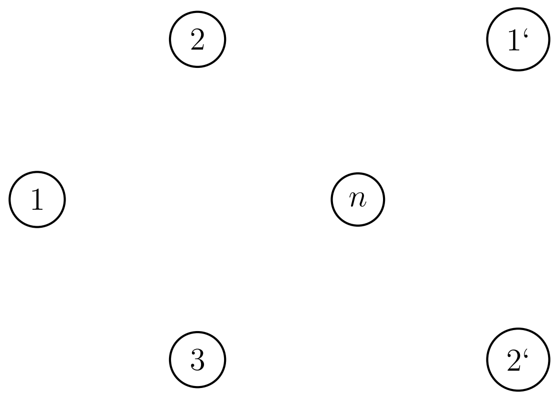
cazul minim: cand toate tramvaiele au statia i in comun si intervalele de asteptare se suprapun $((start_i, end_i) \cap (start_j, end_j) \neq \emptyset)$



In desenul alaturat prima componenta cu nodurile *1,2,3,n* este reprezentata de tramvaiele *1,2,3,...,n* care sunt in statia 1 iar cea dea doua componenta conexa este data de tramvaiele *1',2'* care sunt in statia n (asta e o reprezentare minimala a cazului acestuia).

Deci cazul minim cuprinde n grupari (n dat de cele n statii) de cate m tramvaie, adica n componente conexe.

cazul maxim: cand toate tramvaiele au statia i in comun si intervalele de asteptare nu se suprapun $((start_i, end_i) \cap (start_j, end_j) = \emptyset)$



Deci cazul maxim cuprinde n*m grupari, adica n*m componente conexe.

Deci numarul de componente conexe ale lui G este cel putin numarul de statii de tramvai.

b) Conform teoriei: *O submultime de k noduri a unui graf G care induce un graf complet este numita k-clica, numarul de clica al lui G este w(G) = max* $|Q|$ *unde Q este clica in G*].
Daca ne uitam la enuntul problemei si la subpunctul *a)* (mai ales cazul minim) observam ca tramvaiele cu statia comuna care se regasesc in acelasi timp sunt adiacente intre ele unde k din definitie e data de numarul maxim de tramvaie si aceasta reprezinta *w(G)*.

c) Stim ca fiecare componenta conexa a grafului G este un graf complet. Daca vom lua macar o componenta conexa cu cel putin 3 noduri din G vom avea mereu un circuit indus de lungime 3 (fiecare graf complet are un circuit indus de 3 noduri) deci G nu va avea circuit indus $\geq$*4*.

d)

for $(v \in V)$ **do**
  $visitedNeighbors[v] \leftarrow 0;$
end for
$Q \leftarrow 0;$
for $(v \in V)$ **do**
  $countV \leftarrow 0;$
  $S \leftarrow \emptyset;$
  if $(visitedNeighbors[v] = 0)$ **then**
    $countV \leftarrow 1;$
    $visitedNeighbors[v] \leftarrow 1;$
    $S \leftarrow S \cup \{v\};$

while $(S \neq \emptyset)$ **do**
$v \leftarrow S[0]; //primulElementDinCoada$
$m \leftarrow v;$
$S \leftarrow S \backslash \{v\};$
for $n \in NG(m)$ **do**
if $(visitedNeighbors[n] = 0)$ **then**
$visitedNeighbors[n] \leftarrow 1;$
$countV \leftarrow countV + 1;$
$S \leftarrow S \cup \{n\};$
end if
end for
end while
end if
$Q \leftarrow max(countV, Q);$
end for

Complexitatea algoritmului este de *O(n + m)* data de algoritmul *BFS* iar algoritmul functioneaza astfel:

-vede daca este deja vizitata componenta conexa

-daca nu este vizitata va lua nodul respectiv din lista de noduri si va lua toti vecinii acestuia (totodata in *countV* se va aduna cu 1 de fiecare data cand dam de un vecin)

-se va repeta procesul si cu vecinii vecinilor pana cand toata componenta conexa este deja vizitata (cand lista S ramane vida)

-cand trecem de la o componenta la alta numarul de noduri din componentele conexe se vor trece in Q (in cazul in care este mai mare valoarea acesteia, evident)

Operatia *O(1)* este data de *countV* care va memora cate noduri avem in componenta conexa.

Exercitiul 2

a) Va trebui pentru $\forall$ Kn (o permutare a unui Kn), $n \in N$ sa avem un arc e dintre u si v, $u,v \in V$ si $e \in A$ astfel incat sa nu depinda directia acestuia si sa pastreze proprietatiile lui *Kn*, e fiind arcul din P unde $e \in A' \setminus A''$ iar $\vec{Kn}' = (V, A')$, $\vec{Kn}'' = (V, A'')$. Un exemplu foarte bun e daca pentru *K3* un nod poate avea 2 arce orientate spre el astfel celalalt arc poate avea orice directie, un *K4* se poate forma din acelasi *K3* cu un nod care poate avea 2 arce orientat spre el etc. pentru celalalte *Kn*.

b) Stim ca functia *reverse* schimba directia arcelor astfel pentru orice permutare de Kn putem apela de repetate ori la *reverse* pentru a aduce doua digrafuri la stare identica.

c) Orientarea aciclica a unui graf G este un graf orientat $\vec{G} = (V, A)$ care contine circuite directe si al carui graf suport este graful G. (1)

Kn este un graf complet. (2)

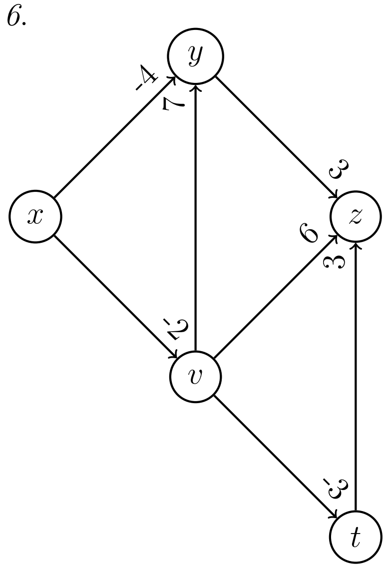
Din (1),(2) rezulta ca $\vec{Kn} = (V, A)$ este o orientare aciclica a lui Kn (graf orientat fara circuite).Daca din $\vec{Kn} = (V, A \cup Ao)$ vom elimina niste muchii la alegere vom obtine un graf $\vec{G} = (V, A)$ care este o orientare aciclica a lui G = *(V,E)* iar oricare multime de muchii eliminate *(E(Kn)* $\setminus$ *E)* reprezinta o orientare aciclica.

Deci pentru orice orientare aciclica $\vec{G}$ a unui graf G, cu n noduri poate fi extinsa la orientare aciclica a lui Kn.

d)Din subpunctul b si c.

Exercitiul 3

Fie G= (V,E) , unde $V = \{x,y,z,t,v\}$ iar $A = \{xy,yz,xv,vy,vt,tz,vz\}$ si cu functia de cost , unde $a : A \rightarrow R$, cu urmatoarele costuri: *a(xy) = -4, a(yz) = 3, a(xv) = -2, a(vy) = 7, a(vt) = -3, a(tz) = 3, a(vz) = 6.*



Observam din desen ca drumul de cost minim dintre nodurile x si z este urmatorul: $x \rightarrow v \rightarrow t \rightarrow z$, cu costul -2.

Acum daca luam toate costurile arcelor si le adunam cu constanta c, c fiind $|min(a)|$ adica modulul minimului functiei a (stim exact ca **exista** macar un arc cu cost minim), drumul initial de cost minim $x \to v \to t \to z$ va deveni de cost 10 (-2 + (-3) + 3 + 4 + 4 + 4), dar in digraf exista si drumul $x \to y \to z$ care va deveni de cost 7 iar acesta initial era de cost -1 (adica nu era in digraf initial un drum de cost minim). Deci contraexemplul functioneaza si astfel algoritmul nu este valid pentru problema aceasta.

Exercitiul 4

Fie G= (V,E), un $s \in V$, cu o functie de cost $a : A \to R$ astfel incat

$$a(s, i) = \begin{cases} < 0, \text{ daca i este vecin cu s} \\ \geq 0, \text{ altfel} \end{cases}$$

Fie $s \to j$ un drum de cost minim in care exista un $k \in V$, $k \in N_G(s)$. Stim ca drumul de cost minim $k \to j$ este un drum de cost minim din corectitudinea algoritmului Dijkstra (orice arc este de cost positiv). (*)

Pentru a demonstra ca drumul $s \to k$ este de cost minim vom alege o constanta c *este* $|min(a(s,i))|$ *pentru* $\forall i \in N_G(s)$ (vom aduce toate arcele la cost positiv). Intrucat doar costurile arcelor care ies din s s-au schimbat drumul $s \to k$ tot ramane acelasi deci algoritmului Dijkstra este corect si pentru drumul $s \to k$. (**)

Din (*) si (**): Algoritmului Dijkstra este corect astfel drumul $s \to j$ *pentru* $\forall j \in V$ este cel de cost minim.