Tema 1 - Algoritmica Grafurilor

Lazăr Cătălina A6 —– Marin Mălina A6

06.11.2020

Exercitiul 1

Vom utiliza algoritmul DFS din curs, cu urmatoarea modificare: initializarile label si parent cu -1 pentru toate nodurile (primele 2 linii din algoritmul din curs) se vor sterge. Aceste initializari trebuie facute o singura data. Vom compara numarul de componente conexe cu k, daca numarul de componente este mai mare atunci nu se poate, deoarece nu putem adauga muchii, altfel este posibil deoarece putem izola noduri care nu sunt puncte de articulatie, treptat, pana la atingerea numarului dorit. Algoritmul final propus are complexitate O(n+m) deoarece algoritmul DFS are complexitate O(n+m).

Algorithm 1: Algoritm exercitiul 1

```
if k > n then
   return Nu ;
   else if k = n then
 └ return
   Da, fiecare nod reprezinta o componenta conexa
else
   count ← 0
   for u ∈ V do
      label[u] ← −1
      parent[u] ← −1
   end for
   for v ∈ V do
      if label[v] = −1 then
         DFS(v)
         count ← count + 1
      end if
   end for
end if
if k < count then
   return Nu ;
   else if k >= count then
 └ return
   Da ;
end if
```

Exercițiul 2

a)

$\forall u, v \in V$ exista 2 situatii:

1. $u, v \in V$ a.i. $uv \in E \Rightarrow d(u, v) = 1$

2. $u, v \in V$ a.i. $uv \notin E \Rightarrow \exists! k \in V$ a.i. $uk, kv \in E \Rightarrow d(u,v) = 2$

Astfel ca, d(G)=2
Presupunem prin reducere la absurd ca ar exista un $C_4 \in G \Rightarrow \exists u, v \in C_4$ cu 2 vecini comuni $\Rightarrow$ Contradictie, deoarece in ipoteza este specificat ca $\forall u, v \in V, \exists! k \in V$ a.i. $uk, kv \in E$.

b)

Din ipoteza $\Rightarrow \forall u, v \in V, \exists! k$ a.i. $uk, kv \in E$. Daca $\exists uv \in E \Rightarrow [A]_G = K_3, A = \{u, v, k\}$
Pentru $\forall v \in V$, daca $\exists u \in V, u \neq v$ a.i. $uv \in E \Rightarrow \exists [A]_G = K_3, A = \{u, v, k\} \Rightarrow \sum_{u \in A} d_u = 3 * 2$. Pentru $\forall u \in [\{v\} \cup N_G(V)]_G, u \neq v$ se va forma o $3-clica \Rightarrow$ vor fi $d_G(v)/2$ astfel de subgrafuri complete (deoarece fiecare vecin va presupune inca un vecin, deci vor fi perechi) $\Rightarrow 2 * |E_{[\{v\} \cup N_G(V)]_G}| = 3 * 2 * d_G(v)/2 \Rightarrow 2 * |E_{[\{v\} \cup N_G(V)]_G}| = 3 * d_G(v) \Rightarrow |E_{[\{v\} \cup N_G(V)]_G}| = 3 * d_G(v)/2$

c)

$\forall u, v \in V$ a.i. $uv \notin E, \exists! k \in V; uk, kv \in E$ si din **b)** $\Rightarrow \{u, k\}$ si $\{k, v\}$ vor forma $K_3 \Rightarrow d_{K_3}(u) = 2; d_{K_3}(v) = 2$.
$d_G(u) = 2, d_G(v) = 2$ deoarece $\forall a \in V$ a.i. $au \in E$ din ipoteza $\Rightarrow \forall u, v \in V, \exists! x$ a.i. $ux, xv \in E$ daca $x = k \Rightarrow$ atunci u si k vor avea 2 noduri in comun. daca $x \neq k \Rightarrow a$ nu ar avea o cunostinta comuna cu restul nodurilor $\Rightarrow$ Contradictie. $\Rightarrow d_G(u) = d_G(v) = 2$.

Exercitiul 3

a)

Folosim algoritmul din Cursul 4 pentru determinarea unei ordonari topologice, doar ca in loc de gradul nodului vom folosi altitudinea acelui nod. Nu vor exista cicluri deoarece vom selecta doar acele noduri aflate in relatia $alt[u] < alt[v]$. Avand in vedere ca vom face sortarea topologica dupa altitudine, nodul-casa si nodul-universitate vor apartine ordonarii. Va trebui determinat punctul de start al ordonarii (casa/universiate, va fi ales cel cu cea mai mica altitudine). Vom determina cele mai scurte drumuri din nodurile casa/universitate si dupa vom vedea care e drumul cel mai scurt care contine unul dintre nodurile comune. Complexitatea va fi $\mathcal{O}(m+n)$ deoarece ordonarea topologica are $\mathcal{O}(m+n)$, iar determinarea celor mai scurte drumuri, daca sunt implementate cu liste, va fi tot $\mathcal{O}(m+n)$.

Algorithm 2: a)Algoritm ordonare, care va primi ca parametru nodul de start si tabloul cu listele de adiacenta pentru fiecare nod va intoarce tabloul cu ordinea

$count \leftarrow 0; S \leftarrow nod_{start}$
while $S \neq \varnothing$ **do**
  $v \leftarrow pop(S); count++; ord[v] \leftarrow count;$
  for $w \in A[v]$ **do**
    if $alt[w] > alt[v]$ **then**
      $push(S, w)$
    end if
  end for
end while
return ord

b)

Pentru a rezolva problema putem aplica algoritmul lui Dijkstra adaugand coditia ca nodurile sa respecte altitudinea: $j \in V - Sandj >= j_*$. Va fi aplicat algoritmul numarul 4, dar fara a mai face ordonarea topologica si pentru Dist(), va fi Dijkstra modificat. Complexitatea va fi de O(n^2) sau chiar O(m+n)$dacaimplementamDijkstracuofamilie$

Algorithm 3: a)Algoritm de determinare a distantei care primeste ca parametri nodul de start si returneaza distantele calculate si tabloul before

$u_a \leftarrow 0; before[a] \leftarrow 0$
for $i = ord(a) + 1, n$ **do**
  $u_{ord(i)} \leftarrow \infty$
  for $j = ord(a), i - 1$ **do**
    if $u_{ord(i)} > u_{ord(j)} + a_{ji}$ **then**
      $u_{ord(i)} \leftarrow u_{ord(j)} + a_{ord(j)ord(i)}$
      $before[ord(j)] \leftarrow ord(i)$
    end if
  end for
end for
return before, u

Algorithm 4: a)Algoritmul principal care va returna drumul-o lista si lungimea

if $alt(casa) > alt(univ)$ **then**
  $ord \leftarrow Ordonare(univ, A); alt_{sup} = casa$
else
  $ord \leftarrow Ordonare(casa, A); alt_{sup} = univ$
end if
$distante_c, before_c \leftarrow Dist(a = casa)$
$distante_u, before_u \leftarrow Dist(a = univ)$
for $i = ord[alt_{sup}] + 1, n$ **do**
  if $min > distante_c(i) + distante_u(i)$ **then**
    $min = distante_c(i) + distante_u(i); nod_{comun} = i$
  end if
end for
for $j \in before[nod_{comun}...alt_{inf}]$ **do**
  $drum[n] \leftarrow j; n + +$
end for
for $j \in before[nod_{comun}...alt_{sup}]$ **do**
  $drum[n] \leftarrow j; n + +$
end for
return drum, min

Exercitiul 4

a)

Aleg o muchie $uv \in F$ a.î. $u \in A$ si $v \in B$
su - drum de cost minim in G-e $\Rightarrow$ pentru $s, u \in A \exists P_u^* = P_{us}^* \in P_{su}$
in G alegem arbitrar $k \in V \setminus \{s, u\}, V = A \cup B$, astfel ca vor fi 2 cazuri:

1. $k \in A$ a.i. $k \in P_{su}^* \Rightarrow P_{su}^* = P_{sk} + P_{ku}$
   Presupunem prin reducere la absurd ca $\exists P_{sk}^{'} < P_{sk} \Rightarrow P_{sk}^{'} + P_{ku} < P_{sk} + P_{ku}$
   $P_{sk}^{'} + P_{ku} < P_{su} \Rightarrow$ Contradictie, deoarece din ipoteza stim ca su este drum de cost minim.

2. $k \in B$ a.i. $k \in P_{su}^* \Rightarrow P_{su}^* = P_{sk} + P_{ku}$
   Din constructia SP-arborelui stim ca sx este drum de cost minim nenegativ in G $\Rightarrow$ pentru orice 2 noduri $\in$ V distanta dintre ele este nenegativa si ca este o ordine bine determinata de a traversa nodurile pentru a forma drumurul $\Rightarrow P_{sa} < P_{sb}, a \in A, b \in B$
   $s, u \in A; k \in B \Rightarrow P_{sk} > P_{su} \Rightarrow P_{sk} + P_{ku} > P_{su} \Rightarrow P_{su}$ drum minim.

Din 1,2 $\Rightarrow P_{su}^*$ este minim si in G.

b)

$v, t \in B$; $vt - drum$ de cost minim in $T_s^2 \Rightarrow \exists P_t^* = P_{vt}^* \in P_{vt}$ in T_s^2.
In G alegem k arbitrar, $k \in V \setminus \{v, t\}, V = A \cup B$ a.i. $K \in P_{vt} \Rightarrow P_{vt} = P_{vk} + P_{kt}$. Astfel, vom avea 2 cazuri:

1. $k \in A$
   Din constructia SP-arborelui stim ca sx este drum de cost minim nenegativ in G $\Rightarrow$ pentru orice 2 noduri $\in$ V distanta dintre ele este nenegativa si ca este o ordine bine determinata de a traversa nodurile pentru a forma drumurul si din $k \in A, v, t \in B \Rightarrow P_{kt} = P_{kv} + P_{kt}$
   $P_{vk} + P_{kt} = P_{vk} + P_{kv} + P_{kt}$
   $P_{vt} = P_{vk} + P_{kt} + P_{kv}$
   $P_{vt} = P_{vt} + P_{kv}$
   $P_{kv} = 0 \Rightarrow$ Contradictie

2. $k \in B$
   Presupunem prin reducere la absurd ca $\exists P_{vk}^{'} < P_{vk} \Rightarrow P_{vk}^{'} + P_{kt} < P_{vk} + P_{kt}$
   $P_{vk}^{'} + P_{kt} < P_{vt} \Rightarrow$ Contradictie, deoarece din ipoteza stim ca su este drum de cost minim in T_s^2, iar $k, v, t \in B$.

c)

1. exista minim o muchie $e \in F$
   Presupunem prin reducere la absurd ca un $st - drum$ nu contine nicio muchie $e \in F$ in $G - e \Rightarrow$ $T_s^1; T_s^2-$componente conexe <u>disjuncte</u> a $G - e \Rightarrow \nexists st - drum \Rightarrow \exists$ minim o muchie e $\in F$

2. pentru ca st sa fie drum de cost minim nu poate fi mai mult de o muchie $e \in F$
   Pp RA ca avem $e_1, e_2 \in F e_1 = uv$, iar e_2:

   (a) $e_2 = uy$
   $su - drum$ minim din G $\Rightarrow$ din u vor porni 2 muchii separate. Pentru ca cele doua muchii $e_1, e_2 \in P_{st} \Rightarrow \exists k \in P_{st}$ a.i. $P_{yt} \cap P_{vt} = k, k \in B \Rightarrow P_{st} = P_{su} + a_{uv} + P_{vk} + P_{kt} + P_{ky} + a_{yu} + P_{u..t}$
   $P_{st} = P_{st} + P_{ky} + a_{yu} + P_{u..t} \Rightarrow$ Contradictie
   Alegand o a doua muchie care trebuie sa apartina drumul se va crea un circuit sau se va parcurge de doua ori muchia pentru a ma intoarce ($P_{u..t}$).

   (b) $e_2 = xt$
   $vt - drum$ minim din G $\Rightarrow$ in v vor ajunge 2 muchii separate. Pentru ca cele doua muchii $e_1, e_2 \in P_{st} \Rightarrow \exists k \in P_{st}$ a.i. $P_{su} \cap P_{sx} = k, k \in A \Rightarrow P_{st} = P_{sk} + P_{ku} + a_{uv} + a_{vx} + P_{x..v}$ ($P_{x..v}$).

(c) $e_2 = xy$
Pentru ca cele doua muchii $e_1, e_2 \in P_{st} \Rightarrow \exists a \in P_{st}$ a.i. $P_{su} \cap P_{sx} = a, a \in A; \exists a \in P_{st}$ a.i. $P_{vt} \cap P_{yt} = b, b \in B \Rightarrow P_{st} = P_{sa} + P_{su} + a_{uv} + P_{vb} + P_{bt} + P_{by} + a_{yx} + P_{xa} + P_{a..b} \Rightarrow P_{st} = P_{st} + P_{by} + a_{yx} + P_{xa} + P_{a..b}$

In toate cele 3 cazuri alegand o a doua muchie $e_2 \in F$ care trebuie sa apartina st se va crea un circuit sau se va parcurge de doua ori muchia pentru a se intoarce spre nodul final, iar acest drum suplimentar este nenegativ, diferit de 0. Similar si daca am adauga mai multe muchii.

Din 1,2 $\Rightarrow st - drum$ de cost minim contine o singura muchie $e \in F$

d)

c)$\Rightarrow P_{st} \cap F = e; e = uv; u \in A, v \in B \Rightarrow P_{st} = a_G(s,i) + a_{uv} + a_G(v,t)$
Pentru $x, y \in A; z, w \in B; x \neq y, z \neq w$
a),b)$\Rightarrow sx, sy, zt, wt-$ drumuri de cost minim in G $\Rightarrow P_{st}^1 = a_G(s,x) + a_{xz} + a_G(z,t); P_{st}^2 = a_G(s,y) + a_{yw} + a_G(w,t)$
Totusi $P_{st}^1 \neq P_{st}^2 \Rightarrow$ trebuie sa alegem u,v astfel incat $a_g(s,u)-$minim,$a_g(v,t)-$minim, $a_g(uv)-$minim
$\Rightarrow P_{st}^* = min_{uv \in F}(a_G(s,i) + a_{uv} + a_G(v,t))$

e)

Folosesc algoritmul lui Dijkstra implementat cu un heap Fibonacci - $\mathcal{O}(m + n \log n)$, care va returna frunza spre care a gasit drumul cu costul cel mai mic si costul efectiv $\Rightarrow$ DijF(arbore, nod de plecare)

Algorithm 5: Algoritm e)

$(u, c_1) \leftarrow DijF(T_S^1, s)$
$(v, c_2) \leftarrow DijF(T_S^2, t)$
return $(c_1 + c_2 + a_{uv})$