Algoritmica Grafurilor - Cursul 3

19 octombrie 2018

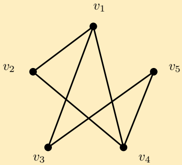
Cuprins

Fie G un graf cu $V(G) = \{v_1, \ldots, v_n\}$. **Matricea de adiacenţă** a lui G este matricea $A = (a_{ij})_{1 \leqslant i,j \leqslant n} \in \mathcal{M}_{n \times n}(\{0, 1\})$, unde

$$a_{ij} = \begin{cases} 1, & \text{dacă } v_i \text{ şi } v_j \text{ adiacente} \\ 0, & \text{altfel} \end{cases}.$$

Exemplu

Un graf şi matricea sa de adiacenţă.



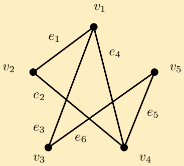
$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Fie G un graf cu $V(G) = \{v_1, \ldots, v_n\}$ şi $E(G) = \{e_1, \ldots, e_m\}$. **Matricea de incidenţă** a lui G este matricea $B = (b_{ij})_{1 \leqslant i,j \leqslant n} \in \mathcal{M}_{n \times m}(\{0,1\})$, unde

$$b_{ij} = \begin{cases} 1, & \text{dacă } e_j \text{ este incidentă cu } v_i \\ 0, & \text{altfel} \end{cases}.$$

Exemplu

Un graf şi matricea sa de incidenţă.



$$= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Pentru digrafuri, matrici similare pot fi definite cu elemente din $\{-1, 0, 1\}$ pentru a marca direcţia arcelor.

Valorile proprii, vectorii proprii şi polinomul caracteristic ale matricei de adiacenţă sunt numite **valorile proprii**, **vectorii proprii**, şi, respectiv, **polinomul caracteristic ale grafului** corespunzător. Acestea sunt obiectele de studiu ale teoriei spectrale a grafurilor.

Fie $G = (V, E)$ un (di)graf cu $V = \{1, 2, \ldots, n\}$.

- Dacă $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$ este matricea de adiacenţă a lui G atunci, reprezentând-o ca un tablou 2-dimensional, avem nevoie de $\mathcal{O}(n^2)$ timp pentru iniţializare (în funcţie de limbajul de programare).
- Astfel, orice algoritm care reprezintă G cu matrice de adiacenţă are complexitatea timp (şi spaţiu) $\Omega(n^2)$.
- Testarea dacă două noduri fixate sunt adiacente se face în $\mathcal{O}(1)$, dar parcurgerea întregii mulţimi de vecini $N_G(u)$ (sau $N_{G^+(u)}$), pentru un anume nod $u \in V$, necesită $\Omega(n)$ - prea mult pentru grafuri rare de ordin mare.

Fie $G = (V, E)$ un (di)graf cu $V = \{1, 2, \ldots, n\}$ şi $|E| = e$.

- Orice nod $u \in V$ are o listă, $A(u)$, a vecinilor săi din G:
  - când G este graf $A(u) = N_G(u)$;
  - dacă G este digraf, atunci $A(u) = N_G^+(u) = \{v \in V : uv \in E\}$;
- Dacă G este graf, atunci fiecare muchie $uv \in E$ generează două elemente în listele de adiacenţă: unul în $A(u)$ şi unul în $A(v)$; spaţiul necesar este $\mathcal{O}(n + 2e)$.
- Dacă G este digraf, atunci spaţiul necesar este $\mathcal{O}(n + e)$.
- Listele de adiacenţă pot fi implementate folosind **liste înlănţuite** sau tablouri.
- Testarea dacă un nod u este adiacent cu un altul v în G necesită $\Omega(d_G(u))$ timp, dar **parcurgerea întregii mulţimi de vecini $N_G(u)$ (sau $N_G^+(u)$)**, pentru un nod $u \in V$, se poate face în $\Omega(d_G(u))$ (şi nu în $\mathcal{O}(n)$ ca în cazul matricei de adiacenţă).

Traversarea (sistematică a) grafurilor (**graph search, graph traversal**) este o paradigmă algoritmică care specifică o metodă sistematică de a parcurge mulţimea nodurilor care pot fi accesate (atinse) pe drumuri plecând dintr-un nod fixat al (di)grafului.

Dat un (di)graf $G = (\{1, \ldots, n\}, E)$ şi $s \in V(G)$

generează "eficient" muţimea

$$S = \{u \in V(G) : \text{ există un drum de la } s \text{ la } u \text{ in } G\}.$$

G va fi reprezentat cu liste de adiacenţă, deoarece, în timpul procesului de traversare, trebuie manipulată în mod eficient muţimea de vecini a nodului curent.

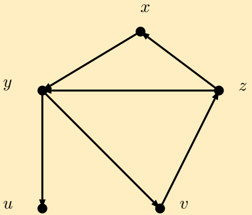
for $v \in V$ **do**
  $label(u) \leftarrow -1$; $parent(u) \leftarrow -1$;
$label(s) \leftarrow 0$; $parent(s) \leftarrow 0$;
create queue Q conţinând s;
while $Q \neq \varnothing$ **do**
  $u \leftarrow pop(Q)$;
  for $v \in A(u)$ **do**
    if $label(v) < 0$ **then**
      $label(v) \leftarrow label(u) + 1$;
      $parent(v) \leftarrow u$; $push(Q, v)$;

Proprietăţi ale BFS. Nu este dificil de arătat că:

- $S = \{u \in V \;:\; label(u) \geqslant 0\}$;
- $\forall u \in V, label(u) = d_G(s, u)$ (distanţa în G de la s la u);
- Variabila parent defineşte **arborele bfs** asociat parcurgerii din s: dacă G este graf atunci arborele bfs este un arbore parţial al componentei conexe conţinând s; dacă G este digraf atunci arborele bfs este o **arborescenţă** (arbore cu rădăcină, orientat - arcele sunt îndreptate către exteriorul rădăcinii s).
- Complexitatea timp a $BFS(s)$ este $\mathcal{O}(n_S + m_S)$, unde $n_S = |S| \leqslant |V| = n$, şi $m_S = |E([S]_G)| \leqslant |E|$ (se observă că fiecare vecin din lista de adiacenţă a unui nod din S este accesat o singură dată).

Exemplu

Două parcurgeri BFS ale unui aceluiaşi graf (începând din z şi din v):



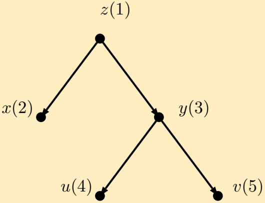
$A(x) = [y]$

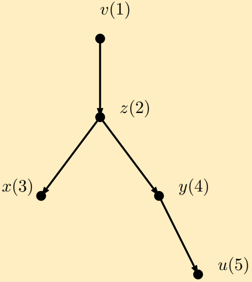
$A(y) = [u, v]$

$A(z) = [x, y]$

$A(u) = \emptyset$

$A(v) = [z]$





(Am marcat în paranteze ordinea de vizitare.)

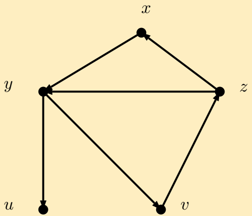
for $v \in V$ **do**
    $label(u) \leftarrow -1$; $parent(u) \leftarrow -1$;
$label(s) \leftarrow 0$; $parent(s) \leftarrow 0$;
create stack S conţinând s; $n_S \leftarrow 0$;
while $S \neq \varnothing$ **do**
    $u \leftarrow top(S)$;
    if $((v \leftarrow next[A(u)]) \neq NULL)$ **then**
        if $label(v) < 0$ **then**
            $n_S + +$; $label(v) \leftarrow n_S$;
            $parent(v) \leftarrow u$; $push(S, v)$;
    else
        $delete(S, u)$;

Proprietăţi ale DFS. Nu este dificil de arătat că:

- $\{u \in V : label(u) \geqslant 0\}$ este exact muţimea S de noduri accesibile pe drumuri din s;
- $\forall u \in V, label(u) =$ momentul vizitării lui u (s este vizitat la momentul 0);
- Variabila parent defineşte **arborele dfs** asociat parcurgerii din s
- Complexitatea timp a $DFS(s)$ este $\mathcal{O}(n_S + m_S)$, unde $n_S = |S| \leqslant |V| = n$, şi $m_S = |E([S]_G)| \leqslant |E|$ (se observă că fiecare vecin din lista de adiacenţă a unui nod din S este accesat o singură dată).

Exemplu

Două parcurgeri DFS ale unui aceluiaşi graf (începând din v şi din x):



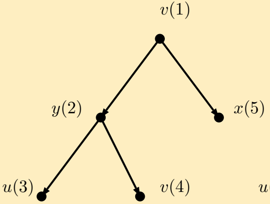
$A(x) = [y]$

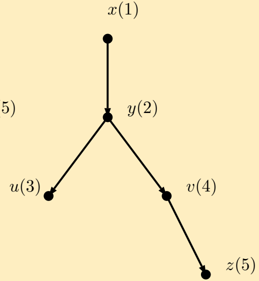
$A(y) = [u, v]$

$A(z) = [x, y]$

$A(u) = \emptyset$

$A(v) = [z]$





Fie $G = (V, E)$ un digraf, cu $V = \{1, \ldots, n\}$.

- Fiecare arc (muchie orientată) $e \in E$ are asociat un cost $a(e) \in \mathbb{R}$ (pondere, lungime etc).
- Dacă G este reprezentat cu liste de adiacenţă, atunci $a(ij)$ este un câmp al elementului din lista de adiacenţă a lui i (reprezentând un arc ij).
- Pentru uşurinţa notaţiei vom folosi reprezentarea lui G cu o matrice de cost-adiacenţă $A = (a_{ij})_{1 \leqslant i,j \leqslant n}$, unde

$$a_{ij} = \begin{cases} a(ij), & \text{dacă } ij \in E \\ \infty, & \text{altfel} \end{cases}.$$

- Aici, ∞ reprezintă un număr real foarte mare relativ la costurile muchiilor(e.g., $\infty > n \cdot \max\limits_{ij \in E} a(ij)$) şi presupunem că $\infty \pm a = \infty$, $\infty + \infty = \infty$.
- Este de asemeni posibil să folosim ∞ ca un acces nereuşit la structura de date utilizată pentru reprezentarea matricei A.
- Pentru $i, j \in V$, mulţimea tuturor drumurilor din G de la i la j este notată cu $\mathcal{P}_{ij}$:

$$\mathcal{P}_{ij} = \{P \ : \ P \text{ este un drum de la } i \text{ la } j\}.$$

- Dacă $P_{ij} \in \mathcal{P}_{ij}$, $P : (i =) v_0, v_0v_1, v_1, \ldots, v_{r-1}, v_{r-1}v_r, v_r (= j)$, atunci

$$V(P_{ij}) = \{v_0, v_1, \ldots, v_r\}, E(P_{ij}) = \{v_0v_1, \ldots, v_{r-1}v_r\}.$$

- Costul lui $P_{ij} \in \mathcal{P}_{ij}$ este

$$a(P_{ij}) = 0 + \sum_{uv \in E(P_{ij})} a_{uv}.$$

- În particular $a(P_{ii}) = 0$.

Drumul de cost minim între două noduri.

P1 Date $G = (V, E)$ digraf; $a : E \to \mathbb{R}$; $s, t \in V$, $s \neq t$.
Găsiţi $P^*_{st} \in \mathcal{P}_{st}$, aşa încât $a(P^*_{st}) = \min\{a(P_{st}) \;:\; P_{st} \in \mathcal{P}_{st}\}$.

Drumurile de cost minim dintre un nod şi toate celelalte noduri

P2 Date $G = (V, E)$ digraf; $a : E \to \mathbb{R}$; $s \in V$.
Găsiţi $P^*_{si} \in \mathcal{P}_{si}, \forall i \in V$, a. î. $a(P^*_{si}) = \min\{a(P_{si}) \;:\; P_{si} \in \mathcal{P}_{si}\}$

Toate drumurile de cost minim.

P3 Date $G = (V, E)$ digraf; $a : E \to \mathbb{R}$.
Găsiţi $P^*_{ij} \in \mathcal{P}_{ij}, \forall i, j \in V$, a. î. $a(P^*_{ij}) = \min\{a(P_{ij}) \;:\; P_{ij} \in \mathcal{P}_{ij}\}$

Remarci 1

- Reprezentarea cu matrice de cost-adiacenţă a perechii (G, a) implică $\mathcal{P}_{ij} \neq \varnothing$, $\forall i, j \in V(G)$: dacă $a(P_{ij}) < \infty$, atunci P_{ij} este un drum real în G, iar dacă $a(P_{ij}) = \infty$, atunci P_{ij} nu este un drum în G dar este un drum în digraful complet simetric obţinut din G prin adăugarea tuturor arcelor lipsă (cu costuri ∞).
- Urmează că toate mulţimile din care se determină un element (drum) de cost minim în problemele **P1** - **P3** sunt nevide şi finite şi drumurile minime cerute sunt bine definite.
- Algoritmii pentru rezolvarea problemei **P1** se pot obţine din cei pentru rezolvarea problemei **P2** prin adăugarea unei condiţii (evidente) de oprire.
- Problema **P3** poate fi rezolvată prin iterarea oricărui algoritm pentru problema **P2**. Vom vedea că sunt şi soluţii mai eficiente.

1. Reţele de comunicare (Communication Networks). Digraful $G = (V, E)$ reprezintă o reţea de comunicaţii între nodurile din V, iar E modelează mulţimea legăturilor orientate dintre noduri.

- Dacă $a(e) \geqslant 0$ $(\forall e \in E)$ reprezintă lungimea conexiunii directe dintre extremităţile lui e, atunci problemele **P1 - P3** sunt probleme de drumuri de cost minim naturale.
- Dacă $a(e) \geqslant 0$ $(\forall e \in E)$ reprezintă timpul necesar pentru parcurgerea conexiunii directe dintre extremităţile lui e, atunci problemele **P1 - P3** probleme pentru determinarea drumurilor celor mai rapide.
- Dacă $a(e) \in (0, 1]$ $(\forall e \in E)$ reprezintă probabilitatea funcţionării corecte a conexiunii directe dintre extremităţile lui e şi dacă presupunem că muchiile funcţionează independent una de cealaltă, atunci problemele **P1 - P3** sunt probleme pentru determinarea drumurilor celor mai sigure:

Dacă $P_{ij} \in \mathcal{P}_{ij}$ pentru o pereche $i, j \in V$, atunci probabilitatea ca acest drum să funcţioneze corect este (datorită independenţei)

$$Prob(P_{ij}) = \prod_{e \in E(P_{ij})} a(e).$$

Luând $a'(e) = -\log a(e)$,

$$\log Prob(P_{ij}) = \log \left(\prod_{e \in E(P_{ij})} a(e) \right) = -\sum_{e \in E(P_{ij})} a'(e).$$

Funcţia log fiind monotonă, problemele **P1 - P3** cu costurile a', oferă drumurile cele mai fiabile în reţele de comunicare.

2. Reţele PERT - Metoda drumului critic (Critical path method - CPM). PERT (**P**ert **E**valuation and **R**eview **T**echnique) este o metodă de a analiza (îndeosebi) îndeplinirea fiecărei sarcini dintr-un proiect mai complex.

- Fie $P = \{A_1, A_2, \ldots, A_n\}$ activităţile atomice ale unui mare proiect P (n este foarte mare). $(P, <)$ este o mulţime parţial ordonată, unde $A_I < A_j$ dacă $i \neq j$ şi activitatea A_i poate începe înainte ca activitatea A_j să se fi terminat.
- Pentru fiecare activitate A_i, timpul necesar finalizării t_i este cunoscut (sau doar estimat).

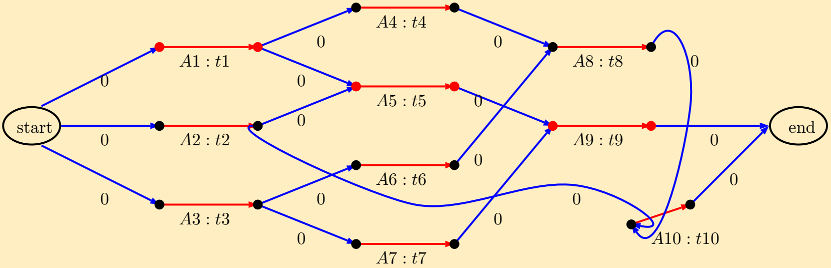
Găsiţi o planificare a activităţilor acestui proiect care să minimizeze durata totală până la finalizare.

Asociem acestei problem un digraf aciclic astfel:

- pentru fiecare activitate A_p ($p \in \{1, \ldots, n\}$) adăugăm un arc $i_p j_p$ de cost $a(i_p j_p) = t_p$;
- nodul i_p corespunde startului activităţii A_p iar nodul j_p este asociat finalizării ei;
- dacă activitatea A_k poate porni imediat după activitatea A_p adăugăm arcul $j_p i_k$ (o activitate fictivă - dummy activity).

- Construcţia digrafului este încheiată după adăugarea unui nod s corespunzând momentului iniţial al proiectului, legat prin arce si_p pentru fiecare activitate A_p în care nu intră vreun arc şi a unui nod t corespunzând finalizării proiectului, legat prin arce $j_p t$ pentru fiecare activitate A_p din care nu iese vreun arc.
- În digraful obţinut **costul maxim al unui drum de la s la t este egal cu timpul minim necesar îndeplinirii în întregime a proiectului**.
- Un drum de cost este numit a drum critic deoarece orice întârziere a unei activităţi de pe acest drum implică o întârziere a întregului proiect.

Exemplu


start
end
A1 : t1
A2 : t2
A3 : t3
A4 : t4
A5 : t5
A6 : t6
A7 : t7
A8 : t8
A9 : t9
A10 : t10
0

3. Problema rucsacului $(0-1)$. Avem un rucsac de dimensiune $b \in \mathbb{N}$, şi n obiecte de dimensiuni $a_1, \ldots, a_n \in \mathbb{N}$. Se cunoaşte şi profitul $p_i \in \mathbb{N}$ al adăugării obiectului i ($i \in \{1, \ldots, n\}$) în rucsac. Se cere să se găsească o completare a rucsacului care să maximizeze profitul total. Fie X_i, pentru $i \in \{1, \ldots, n\}$, o variabilă booleană cu semnificaţia $x_i = 1$ dacă şi numai dacă obiectul i este pus în rucsac. Problema rucsacului poate fi descrisă astfel

$$\max \{\sum_{i=1}^{n} p_i x_i \; : \; \sum_{i=1}^{n} a_i x_i \leqslant b, x_i \in \{0, 1\}, \forall i = \overline{1, n}\}.$$

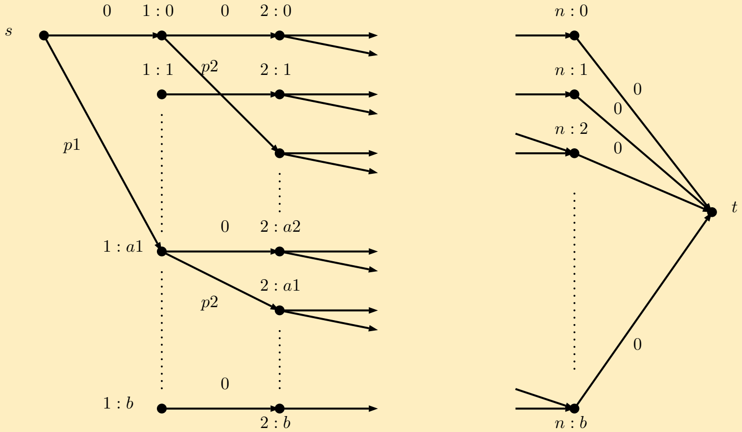
- Fie $G = (V, E)$ un digraf cu $V = \{s\} \cup V_1 \cup \ldots \cup V_n \cup \{t\}$, unde $V_i = \{i^0, i^1, \ldots, i^b\}$ este asociat obiectului i, $i = \overline{1, n}$.
- Arcele lui G şi costurile sunt:
  - $s1^0$ şi $s1^{a_1}$ cu $a(s1^0) = 0, a(s1^{a_1}) = p_1$ (obiectul 1 este adăugat rucsacului cu profitul p_1 şi nivelul de umplere a_1 , sau nu esteadăugat, cu profitul şi nivelul de umplere 0).
  - $(i-1)^j i^j$ cu $a((i-1)^j i^j) = 0$, $\forall i = \overline{2, n}, \forall j = \overline{0, b}$ (obiectul i nu este adăugat rucsacului: după competarea cu primele $i-1$ obiecte şi nivelul de umplere j se trece la umplerea cu primele i objecte, fără obiectul i; nivelul de umplere rămâne neschimbat j iar profitul adăugat este 0).
  - Dacă $j - a_i \geqslant 0$, atunci avem şi arcul $(i-1)^{j-a_i} i^j$ cu $a((i-1)^{j-a_i} i^j) = p_i$ (se poate ajunge la nivelul de umplere j prin adăugarea obiectului i la o umplere cu primele $i-1$ obiecte, cu nivelul de umplere $j - a_i$).

- $n^j t$ cu $a(n^j t) = 0, \forall j = \overline{0, b}$.

Remarci 2

- Fiecare drum de la s la t în G corespunde unei submulţimi de obiecte cu nivelul de umplere $\leqslant b$ şi cu profitul total egal cu costul drumului. Deoarece, reciproc, fiecărei umpleri a rucsacului îi corespunde un drum dela s la t în G, urmează că problema rucsacului poate fi rezolvată prin determinarea unui drum de cost maxim în digraful aciclic G.
- Descrierea statică dată mai sus pentru G poate fi transformată într-una procedurală, folosind programarea dinamică. Problema aceasta este **NP**-hard (ordinul lui G poate fi exponenţial în dimensiunea problemei).

Exemplu



P2 Dat $G = (V, E)$ digraf; $a : E \to \mathbb{R}$; $s \in V$.
Găsiţi $P^*_{si} \in \mathcal{P}_{si}, \forall i \in V$, a. î. $a(P^*_{si}) = \min\{a(P_{si}) \; : \; P_{si} \in \mathcal{P}_{si}\}$

Teorema 1

Fie G un digraf, $s \in V(G) = \{1, \ldots, n\}$ şi $a : E(G) \to \mathbb{R}$, a. î.

(I) $\quad a(C) > 0$, *pentru toate circuitele C din G.*

Atunci $(u_1, \ldots, u_n)$ este o soluţie a sistemului de ecuaţii

(B) $$\begin{cases} u_s = & 0 \\ u_i = & \min\limits_{j \neq i}(u_j + a_{ij}) \end{cases}$$ *dacă şi numai dacă*

$\forall i \in V(G)$, $\exists P^*_{si} \in \mathcal{P}_{si}$ a. î. $u_i = a(P^*_{si}) = \min\{a(P) \; : \; P \in \mathcal{P}_{si}\}$.

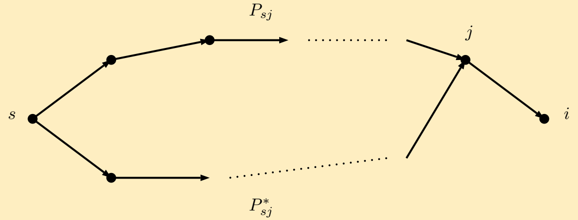
Proof:

"$\Leftarrow$". Fie P^*_{si} o soluţie optimă a problemei **P2** şi $u_i = a(P^*_{si})$. Ipoteza (I) implică $u_s = 0$, i.e., prima ecuaţie a sistemului (B) este satisfăcută. Pentru $i \neq s$, drumul P^*_{si} are penultimul nod j. Dacă P_{sj} este drumul de la s la j determinat pe P^*_{si} de j, avem

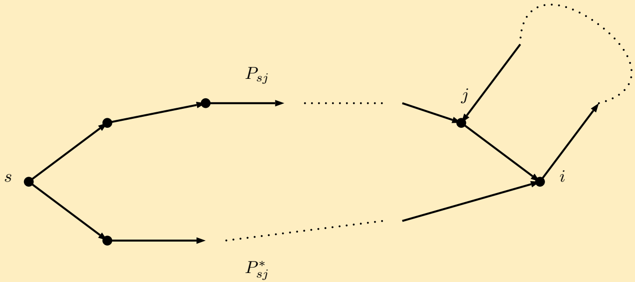
$$u_i = a(P^*_{si}) = a(P_{sj}) + a_{ji} \geqslant a(P^*_{sj}) + a_{ji} = u_j + a_{ji}.$$

Arătăm că $u_i = u_j + a_{ji}$. Presupunem că $u_i > u_j + a_{ji}$, i. e., $a(P_{sj}) > a(P^*_{sj})$.

Cazul 1. $i \notin V(P^*_{sj})$. Atunci $P^1 = P^*_{sj} \circ (j, ji, i) \in \mathcal{P}_{si}$ şi $a(P^1) = a(P^*_{sj}) + a_{ji} < a(P_{sj}) + a_{ji} = a(P^*_{si})$, contradicţie ($P^*_{si}$ este un drum de cost minim).



Cazul 2. $i \in V(P^*_{sj})$. Fie $P^*_{sj} = P_{si} \circ P_{ij}$ cele două drumuri determinate de nodul i on P^*_{sj}. Atunci costul circuitului $C = P_{ij} \circ (j, ji, i)$ este $a(C) = a(P_{ij}) + a_{ji} = a(P^*_{sj}) - a(P_{si}) + a_{ji} = u_j + a_{ji} - a(P_{si})$ care este este $\leqslant u_j + a_{ji} - a(P^*_{si}) = u_j + a_{ji} - u_i < 0$, contradicţie (ipoteza (I) este încălcată).



Astfel partea "$\Leftarrow$" este demonstrată.

Remarcă 1

Am demonstrat mai sus că dacă j este nodul dinaintea lui i pe un drum de cost minim de la s la i, atunci drumul de la s la j determinat de j on acest drum este un drum de cost minim de la s la j. Inductiv, urmează: **Principiul de optimalitate al lui Bellman: Dacă P^*_{si} este un drum de cost minim de la s la i, atunci $\forall j \in V(P^*_{si})$, dacă $P^*_{si} = P_{sj} \circ Pji$, atunci P_{sj} (respectiv P_{ji}) este un drum de cost minim de la s la j (respectiv de la j la i).**

"$\Rightarrow$". Arătăm că dacă $(u_1, \ldots, u_n)$ este o soluţie a sistemului (B), atunci

(a) $\exists P_{si} \in \mathcal{P}_{si}$ aşa încât $u_i = a(P_{si})$, $\forall i \in V$.

(b) $\forall i \in V$, $u_i = \min\{a(P) : P \in \mathcal{P}_{si}\}(= a(P_{si}))$.

(a) Dacă $i = s$, atunci $u_s = 0$ şi drumul P_{ss} satisface $a(P_{ss}) = 0 = u_s$. Dacă $i \neq s$, considerăm următorul algoritm

$v \leftarrow i$; $k \leftarrow 0$;
while $v \neq s$ **do**
  find w a. î. $u_v = u_w + a_{wv}$; // $\exists w$ deoarece u_v stasface (B)
  $i_k \leftarrow v$; $k + +$; $v \leftarrow w$;

Algoritmul determină drumul $P : (s =)i_{k+1}, i_{k+1}i_k, i_k, \ldots, i_1, i_1 i_0, i_0(= i)$ cu $P \in \mathcal{P}_{si}$ şi

$$a(P) = a(i_{k+1} i_k) + \cdots + a(i_1 i_0) =$$

$$(u_{i_k} - u_{i_{k+1}}) + (u_{i_{k-1}} - u_{i_k}) + \cdots + (u_{i_0} - u_{i_1}) =$$

$$u_{i_0} - u_{i_{k+1}} = u_i - u_s = u_i,$$

(altfel obţinem un circuit de cost 0, în contradicţie cu ipoteza (I)).
Cu notaţiile din algoritmul de mai sus avem $u_i = u_{i_1} + a_{i_1 i}$.
(b) Fie $\overline{u}_i = a(P^*_{si})$, $\forall i \in V$. Din demonstraţia anterioară, $\overline{u}_i$, $i = \overline{1, n}$ satisface sistemul (B).
Presupunem că $u = (u_1, \ldots, u_n) \neq (\overline{u}_1, \ldots, \overline{u}_n) = \overline{u}$.
Deoarece $u_s = \overline{u}_s = 0$, urmează că există un $i \neq s$ aşa încât $u_i \neq \overline{u}_i$ şi $\forall j \in V(P_{si})$. $j \neq i$, $u_j = \overline{u}_j$, unde P_{si} este drumul construit la (a) pentru $\overline{u}_i$.

Atunci $u_i > \overline{u}_i = \overline{u}_{i_1} + a_{i_1 i} = u_{i_1} + a_{i_1 i} \geqslant u_i$ (prima inegalitate are loc datorită modului de alegere a lui i, a doua inegalitate are loc deoarece u_i satisface (B)).

Contradicţia găsită arată că $u = \overline{u}$, i. e., elementele lui u sunt costuri minime.

□

Remarci 3

- Din demonstraţia de mai sus urmează că pentru a rezolva **P2** este suficient să se găsească o soluţie a sistemului de ecuaţii (B). Drumurile de cost minim corespunzătoare pot fi obţinute ca la (a) din demonstraţia de mai sus: dacă avem $u_i = u_k + a_{ki}$ atunci k este nodul dinaintea lui i pe drumul de cost minim de la s la i (de cost u_i). În algoritmul care rezolva sistemul (B) menţinem un tablou $before[1..n]$ cu elemente din $V \cup \{0\}$ cu semnificaţia finală $before[i] =$ nodul dinaintea lui i pe un drum de cost minim de la s la i. Nodurile de pe acest drum pot fi determinate în $\mathcal{O}(n)$ prin construcţia secvenţei $i, before[i], before[before[i]], \ldots, s$.

Remarci 4

- Dacă algoritmii care rezolva sistemul de ecuaţii (B) ocolesc (prin întreţinerea tabloului *before*) circuitele de cost 0, atunci problema **P2** este rezolvată, chiar dacă se pierde unicitatea soluţiei. Astfel, aceşti algoritmi vor rezolva **P2** în ipoteza

(I') $\qquad a(C) \geqslant 0,$ pentru toate circuitele C din G.

Remarci 5

- Dacă, în problemele **P1 - P3**, G este un graf şi nu un digraf, putem folosi algoritmii pentru digrafuri înlocuind fiecare muchie a lui G cu o pereche simetrică de arce, fiecare cu acelaşi cost ca muchia. Această abordare funcţionează doar pentru muchii de cost nenegativ (dacă o muchie are cost negativ, atunci 2-circuitul corespunzător format cu cele două arce simetrice are cost negativ, deci ipoteza (I') este încălcată).

Remarci 6

- Deoarece muţimile P_{ij} sunt finite (şi nevide), putem considera probleme similare cu **P1 - P3** înlocuind *min* cu *max*.
- Relaţia evidentă $\max_{x \in A} x = -\min_{x \in A}(-x)$, prin înlocuirea costurilor a_{ij} cu $-a_{ij}$, se poate folosi doar în digrafuri în care, pentru fiecare circuit C, avem $a(C) \leqslant 0$ (în particular, această abordare merge pentru digrafuri fără circuite). **Dacă digraful are circuite, problema celui mai lung drum este, în general, NP - hard**.

Exerciţiul 1.

Spunem că un graf $G = (V, E)$ este rar dacă $m \leqslant cn^2/\log n$ ($n = |V|, m = |E|$). Motivul este acela că putem reprezenta matricea de adiacenţă A a lui G folosind doar $\mathcal{O}(n^2/\log n)$ spaţiu de memorie aşa încât răspunsul la întrebarea "$a(i, j) = 1$?" să poată fi dat în $\mathcal{O}(1)$. Descrieţi o astfel de reprezentare.

Exerciţiul 2.

Arătaţi că nu există o ordonare $e_1, e_2, \ldots, e_{10}$ a muchiilor unui graf K_5, aşa încât: e_{10} şi e_1 nu sunt adiacente şi e_i şi e_{i+1} nu sunt adiacente pentru orice $1 \leqslant i \leqslant 9$.

Exerciţiul 3. Fie $G = (V, E)$ un graf de ordin n şi dimensiune m cu matricea de adiacenţă A. Din mulţimea celor 2^m orientări posibile ale tuturor muchiilor sale alegem una şi considerăm matricea de incidenţă arc-nod $Q \in \mathcal{M}_{n\times n}(\{-1, 0, 1\})$.

$$q_{ve} = \begin{cases} -1, & \text{dacă } v \text{ este extremitatea iniţială a arcului } e \\ 1, & \text{dacă } v \text{ este extremitatea finală a arcului } e \\ 0, & \text{dacă } e \text{ nu este incident cu } v. \end{cases}$$

Arătaţi că $A + QQ^T$ este matrice diagonală şi aflaţi interpretarea combinatorială a elementelor sale diagonale.

Exerciţiul 4. Fie $D = (V, E)$ un digraf cu $V = \{v_1, v_2, \ldots, v_n\}$ şi $E = \{e_1, e_2, \ldots, e_m\}$. Fie $B = (b_{ij}) \in \mathcal{M}_{n\times m}(\{-1, 0, 1\})$ matricea de incidenţă a lui D, unde

$$b_{ij} = \begin{cases} 1, & \text{dacă } e_j \text{ este incident dinspre } v_i \\ -1, & \text{dacă } e_j \text{ este incident către } v_i \\ 0, & \text{altfel} \end{cases}.$$

Arătaţi că $\det(M) \in \{-1, 0, 1\}$ pentru orice submatrice pătratică, M, a lui B (i. e., B este o matrice total unimodulară).

Exerciţiul 5. Fie $G = (S, T; E)$ un graf bipartit cu $V = S \cup T = \{v_1, v_2, \ldots, v_n\}$ şi $E = \{e_1, e_2, \ldots, e_m\}$. Fie $B = (b_{ij}) \in \mathcal{M}_{n\times m}(\{0, 1\})$ matricea de incidenţă a lui G, unde

$$b_{ij} = \begin{cases} 1, & \text{dacă } e_j \text{ este incidentă cu } v_i \\ 0, & \text{altfel} \end{cases} .$$

Arătaţi că $\det(M) \in \{-1, 0, 1\}$ pentru orice submatrice pătratică a lui B (i. e., B este o matrice total unimodulară).

Exerciţiul 6. Arătaţi că un digraf are o singură ordonare topologică dacă şi numai dacă are un drum Hamiltonian iar toate celelate arce sunt orientate înainte relativ la direcţia de parcurgere a acestui drum.

Exerciţiul 7. Diametrul unui graf G este cea mai mare distanţa dintre any două noduri din G. Două noduri formează o pereche diametrală de noduri dacă distanţa dintre ele este cât diametrul. Arătaţi că următorul algoritm determină pereche diametrală de noduri într-un arbore dat T:

1. plecând dintr-un nod al lui T, parcurge *bfs* (Breadth First Search) arborele T; fie u ultimul nod vizitat.
2. mai parcurge o data *bfs* arborele T plecând din nodul u; fie v ultimul nod vizitat.
3. returnează perechea (u, v).

Exerciţiul 8. Arătaţi că o parcurgere DFS poate fi utilizată pentru a determina un circuit par într-un graf 3-regulat în $\mathcal{O}(n)$.

Exercise 9.

(a) Arătaţi că pentru un graf bipartit cu n noduri şi m muchii avem $4m \leqslant n^2$.

(b) Descrieţi un algoritm de complexitate timp $\mathcal{O}(n+m)$ care să decidă dacă un graf dat (cu n noduri şi m muchii) este complementul unui graf bipartit.

Exercise 10. Demonstraţi că un graf G este bipartit dacă şi numai dacă orice subgraf indus H al lui G satisface inegalitatea: $2\alpha(H) \geqslant |H|$.

Exercise 11. Fie $G = (S, T; E)$ un graf bipartit şi $X \in \{S, T\}$. G se numeşte *X-lanţ* dacă nodurile lui X pot fi ordonate: $x_1, x_2, \ldots x_k$ ($|X| = k$) astfel ca

$$N_G(x_1) \supseteq N_G(x_2) \supseteq \ldots \supseteq N_G(x_k)$$

(a) Arătaţi că G este S-lanţ dacă şi numai dacă este T-lanţ.

(b) Să presupunem că G (care este bipartit) are ordinul n, dimensiunea m şi este reprezentat folosind liste de adiacenţă. Descrieţi un algoritm de recunoaştere a unui S-lanţ de complexitate timp $\mathcal{O}(n+m)$.

Exercise 12. Fie G un graf; notăm cu $b(G)$ graful obţinut din G prin inserarea simultană a câte unui nou nod pe fiecare muchie a lui G.

a) Arătaţi că $b(G)$ este graf bipartit.

b) Demonstraţi că $G \simeq H$ dacă şi numai dacă $b(G) \simeq b(H)$. De aici deduceţi că testarea izomorfismului între două grafuri poate fi redusă în timp polinomial la testarea izomorfismului între două grafuri bipartite.