

# Suport seminar algoritmica grafurilor

## II. Algoritm pentru drumuri de cost minim de la un vârf la toate vârfurile

Pentru o problemă de drum de cost minim, **se dă** un graf un graf orientat și ponderat  $G = (V, E)$  cu funcția de pondere  $w : E \rightarrow \mathbb{R}$ . **Ponderea**  $w(p)$  drumului  $p = \langle v_0, v_1, \dots, v_k \rangle$  este suma ponderilor arcelor ce formează drumul:

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i).$$

Se definește costul drumului minim  $\delta(u, v)$  de la  $u$  la  $v$ :

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{dacă există un drum de la } u \text{ la } v, \\ \infty & \text{altfel.} \end{cases}$$

Astfel, un **drum de cost minim** de la vârful  $u$  la vârful  $v$  în graful  $G = (V, E)$  este definit ca orice drum  $p$  cu ponderea  $w(p) = \delta(u, v)$ .

### 2.1 Cost minim în grafuri neorientate

Un algoritm care determină drumul de cost minim pentru grafuri neorientate este algoritmul lui Moore.

```
MOORE( $G, u$ )
1.   $l(u) := 0$ 
2.  for toate vârfurile  $v \in V(G)$ ,  $v \neq u$  do
3.       $l(v) := \infty$ 
4.   $Q = \emptyset$ 
5.   $u \rightarrow Q$ 
6.  while  $Q \neq \emptyset$  do
7.       $Q \rightarrow x$ 
8.      for toți vecinii  $y \in N(x)$  do
9.          if  $l(y) = \infty$  then
10.              $p(y) := x$ 
11.              $l(y) := l(x) + 1$ 
12.              $y \rightarrow Q$ 
13. return  $l, p$ 
```

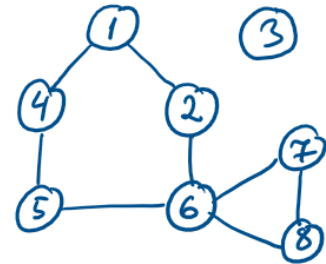
notații:

- $u$  - nod sursă;
- $l(v)$  - lungimea drumului până la vârful  $v$ ;
- $p(v)$  - părintele vârfului  $v$ ;
- $Q$  - structură de date de tip coadă.

**Problema 1.** Pentru graful din figura alăturată să se aplice algoritmul lui Moore și să se determine valorile  $l$  și  $p$  pentru fiecare vârf dacă vârful de pornire este 1.

Soluție

	1	2	3	4	5	6	7	8
l	0	1	$\infty$	1	2	2	3	3
p	-	1	-	1	4	2	6	6



## 2.2 Cost minim pentru grafuri orientate

**Algoritmul Bellman-Ford** Rezolvă problema drumului de cost minim pentru cazul general (ponderile pot lua valori negative). Pentru un graf  $G = (V, E)$  orientat și ponderat cu vârful sursă  $s$  și funcția de pondere  $W : E \rightarrow \mathbb{R}$ , algoritmul Bellman-Ford întoarce o valoare booleană care arată dacă există un circuit de pondere negativă accesibil din  $s$ . Dacă există un astfel de circuit de pondere negativ algoritmul indică faptul că nu există soluție.

Pentru a reprezenta drumul de cost minim (pentru a cunoaște vârfurile care compun drumul de cost minim) se reține pentru fiecare vârf  $v \in V$  din graful  $G = (V, E)$  **predecesorul** acestuia, notat cu  $v.\pi$ , predecesorul poate fi un alt vârf din graf sau **NIL**.

Pentru fiecare vârf  $v \in V$  din graful  $G = (V, E)$  se mai reține atributul  $v.d$  care reprezintă limita superioară a ponderii drumului de la vârful  $s$  la vârful  $v$ . Atributul  $v.d$  se mai numește **estimarea drumului de cost minim**.

**Bellman\_Ford**( $G, w, s$ )

```

1: INITIALIZARE_S( $G, s$ )
2: for  $i = 1$  la  $|V| - 1$  do
3:   for fiecare arc  $(u, v) \in E$  do
4:     RELAX( $u, v, w$ )
5: for fiecare arc  $(u, v) \in E$  do
6:   if  $v.d > u.d + w(u, v)$  then
7:     return FALSE
8: return TRUE

```

Inițializarea atributelor se face astfel:

**INITIALIZARE\_S**( $G, s$ )

```

1: for  $v \in V$  do
2:    $v.d = \infty$ 
3:    $v.\pi = NIL$ 
4:  $s.d = 0$ 

```

După inițializare:  $v.\pi = NIL \forall v \in V$ ,  $s.d = 0$  și  $v.d = \infty \forall v \in V - \{s\}$ . Cu ajutorul procedurii de relaxare se determină/testează dacă se poate îmbunătăți drumul de cost minim de la  $s$  la  $v$  dacă drumul ar trece prin nodul  $u$ , se actualizează  $v.d$  și  $v.\pi$  dacă drumul se îmbunătățește.

**RELAX**( $u, v, w$ )

- 1: **if**  $v.d > u.d + w(u, v)$  **then**
- 2:      $v.d = u.d + w(u, v)$
- 3:      $v.\pi = u$

Figura de mai jos prezintă două exemple de relaxare a unui arc. Deasupra fiecărui vârf se reprezintă estimarea drumului de cost minim ( $v.d$ ).

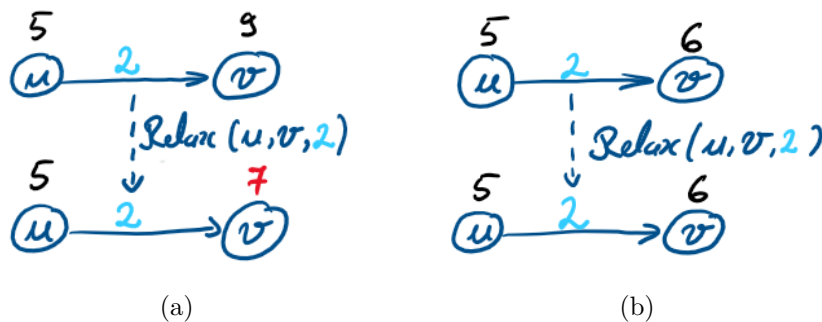


Figura 1: Pentru figura (a), deoarece  $v.d > u.d + w(u, v)$ , valoarea lui  $v.d$  scade (drumul are un cost mai mic dacă trece prin vârful  $u$ ). Pentru figura (b)  $v.d \leq u.d + w(u, v)$  și valoarea  $v.d$  rămâne neschimbată.

**Problema 2.** Pentru graful din figura alăturată să se aplice algoritmul lui Bellman-Ford. Vârful sursă este vârful  $s$ .

**Rezolvare** Vârful sursă este vârful  $s$ . Atributul  $d$  este reprezentat deasupra fiecărui vârf (culoarea roșie indică schimbare în bucla respectivă). Arcele colorate în roșu arată predecesorul unui vârf, astfel dacă arcul  $(u, v)$  are culoarea roșie atunci  $v.\pi = u$ . Pentru acest exemplu se presupune că ordinea de parcurgere a arcelor din graf (liniile 3-4 din algoritm) este:  $(t, x)$ ,  $(t, y)$ ,  $(t, z)$ ,  $(x, t)$ ,  $(y, x)$ ,  $(y, z)$ ,  $(z, x)$ ,  $(z, s)$ ,  $(s, t)$ ,  $(s, y)$ . Figura 2 prezintă execuția algoritmului Bellman-Ford.

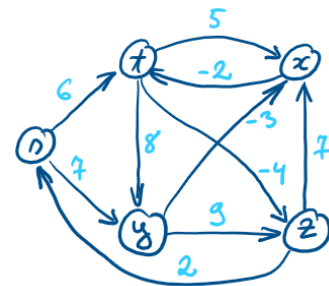


Tabela 1: Valoarea atributelor  $d$  și  $\pi$  la finalul execuției algoritmului pentru graful dat la problema 2.

$v \in V$	s	t	x	y	z
$v.d$	0	2	4	7	-2
$v.\pi$	NIL	x	y	s	t

**Algoritmul Dijkstra** Algoritmul lui Dijkstra rezolvă problema drumului de cost minim pentru un graf  $G = (V, E)$  orientat și ponderat pentru care toate ponderile nu sunt negative. Pentru fiecare arc  $(u, v) \in E$  ponderile  $w(u, v) \geq 0$ .

Algoritmul menține un set  $S$  de vârfuri pentru care s-a determinat ponderea drumului minim de la vârful sursă  $s$ . Algoritmul alege vârful  $u \in V - S$  pentru care estimarea costului drumului

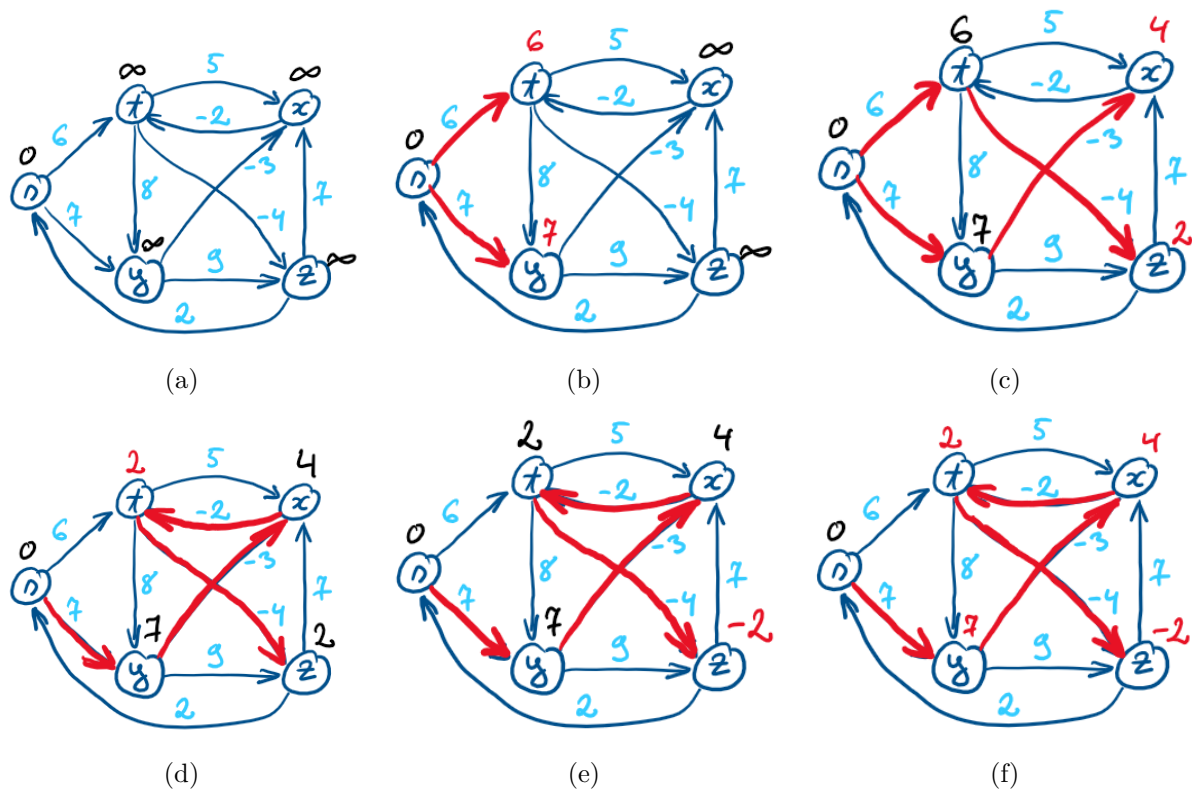


Figura 2: Execuția algoritmului Bellman-Ford. Figura (a) reprezintă graful dat cu valoarea atributului  $d$  pentru fiecare vârf după inițializare. Figurile (b)-(e) prezintă attributele  $d$  și  $\pi$  pentru fiecare vârf după fiecare iterație a buclei for din algoritm (linia 2 algoritm Bellman-Ford). Figura (f) prezintă graful după terminarea execuției algoritmului. Pentru acest exemplu algoritmul întoarce *TRUE*.

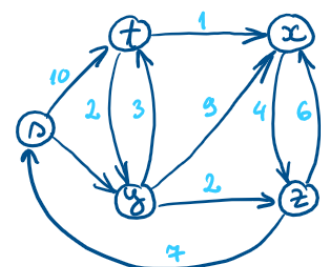
este minimă ( $\min_{u \in V-S} u.d$ ), adaugă  $u$  în  $S$  și relaxează toate arcele care pleacă din  $u$ . Pentru implementare se folosește o coadă minimă de priorități în care elementele sunt ordonate după atributul  $d$ .

### Dijkstra( $G, w, s$ )

```

1: INITIALIZARE_S( $G, s$ )
2:  $S = \emptyset$ 
3:  $Q = V$ 
4: while  $Q \neq \emptyset$  do
5:    $u = \text{EXTRACT\_MIN}(Q)$ 
6:    $S = S \cup \{u\}$ 
7:   for  $v \in G.Adj[u]$  do
8:     RELAX( $u, v, w$ )
    
```

**Problema 3.** Pentru graful alăturat să se ruleze algoritmul lui Dijkstra și să se găsească drumul de cost minim începând din vârful  $s$ .



**Rezolvare** Vârful sursă este vârful  $s$ . Atributul  $d$  este reprezentat deasupra fiecărui vârf (culoarea roșie indică schimbare în bucla respectivă). Arcele colorate în roșu arată predecesorul unui vârf, astfel dacă arcul  $(u, v)$  are culoarea roșie atunci  $v.\pi = u$ . Culoarea verde indică vârful extras din  $Q$ , un vârf colorat în gri aparține setului  $S$ . Figura 3 prezintă execuția algoritmului Dijkstra.

Tabela 2: Valoarea atributelor  $d$  și  $\pi$  la finalul execuției algoritmului pentru graful dat la problema 3.

$v \in V$	s	t	x	y	z
$v.d$	0	8	9	5	7
$v.\pi$	NIL	y	t	s	y

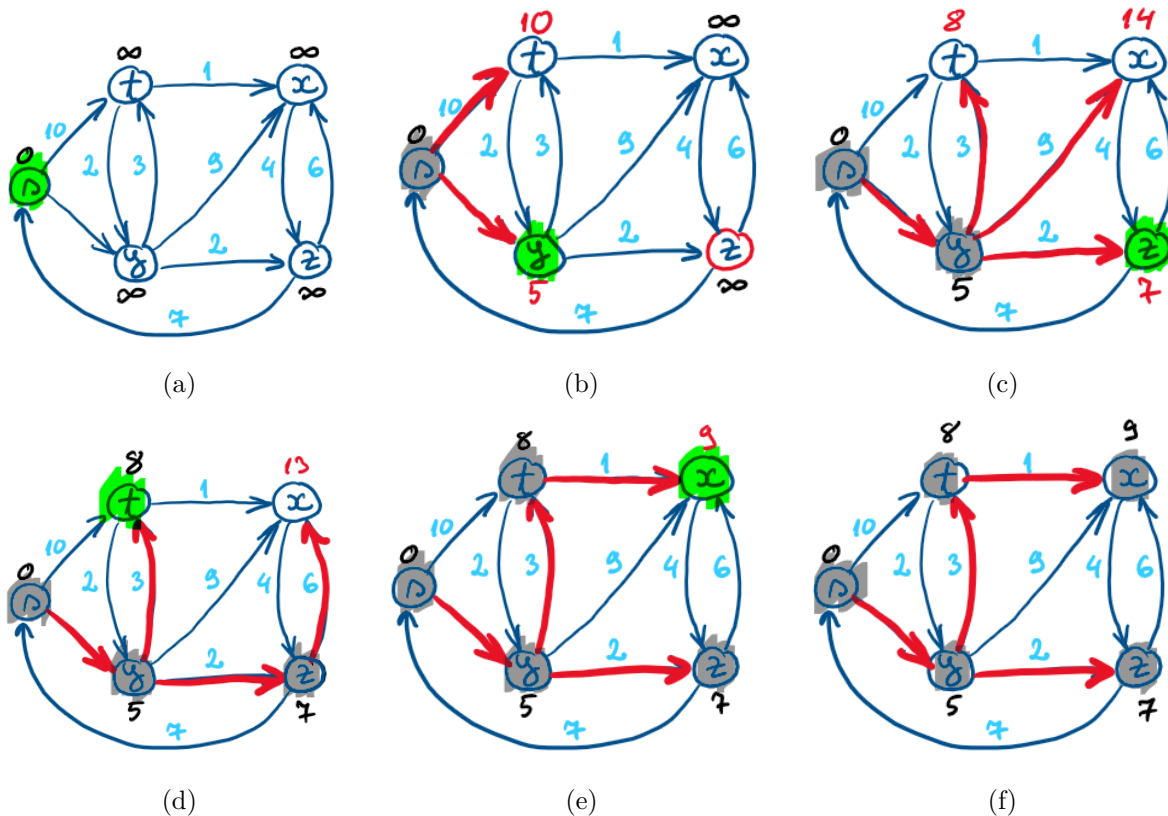


Figura 3: Execuția algoritmului Dijkstra. Vârful sursă este vârful  $s$ , vârfurile cu fundal gri sunt în setul  $S$  iar vârfurile pe un fundal alb sunt în  $Q = V - S$ . Figura (a) reprezintă graful dat înainte de prima iterație a buclei while (liniile 4-8) cu valoarea atributului  $d$  pentru fiecare vârf după inițializare. Vârful care are un fundal verde este cel cu valoarea minimă din  $Q$  și este vârful  $u$  din linia 5 a algoritmului. Figurile (b)-(f) prezintă attributele  $d$  și  $\pi$  pentru fiecare vârf după fiecare iterație a buclei while din algoritm, vârful care are un fundal verde este cel cu atributul  $d$  minim în fiecare iterație (vârful  $u$  din linia 5 a algoritmului).

**Algoritmul Bellman Kalaba** Rezolvă problema drumului de cost minim **de la toate vârfurile la un vârf dat** pentru un graf  $G = (V, E)$  simplu orientat sau neorientat. Este un algoritm matricial, se folosește de matricea de adiacență a grafului. Se alege un vârf (o coloană din matricea de adiacență) și se determină distanțele de la toate vârfurile la acest vârf. Coloana aleasă (vârful ales) este notată

cu  $V^{(1)} = (V_i^{(1)})_{i=1, \dots, n}$ . Dacă matricea de adiacență a grafului este  $A = (a_{ij})_{i,j=1, \dots, n}$ , unde  $a_{ij} = d_{ij}^{(0)}$ . Elementele matricii de adiacență au valoarea:

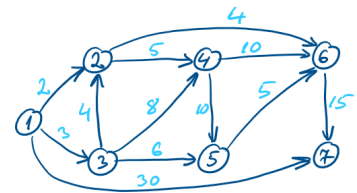
$$a_{ij} = \begin{cases} w(v_i, v_j) & \text{dacă } (v_i, v_j) \in E, \\ 0 & \text{dacă } i = j, \\ \infty & \text{dacă } (v_i, v_j) \notin E. \end{cases}$$

Se determină vectorii pentru  $k = 1, 2, \dots, n$ :

$$V_i^{(k)} = \min_{j=1, \dots, n} \{a_{ij} + V_j^{(k-1)}\}, \text{ pentru } i = 1, 2, \dots, n$$

până când  $V^{(t)} = V^{(t-1)}$  pentru un  $t$ .

**Problema 4.** Pentru graful alăturat să se determine folosind algoritmul lui Bellman-Kalaba costul minim pentru drumurile care duc în vârful 7.



**Soluție** Matricea de adiacență asociată grafului este

$$A = \begin{pmatrix} 0 & 2 & 3 & \infty & \infty & \infty & 30 \\ \infty & 0 & \infty & 5 & \infty & 4 & \infty \\ \infty & 4 & 0 & 8 & 6 & \infty & \infty \\ \infty & \infty & \infty & 0 & 10 & 10 & \infty \\ \infty & \infty & \infty & \infty & 0 & 5 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 15 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

iar vectorii distanță sunt

$$V^{(1)} = \begin{pmatrix} 30 \\ \infty \\ \infty \\ \infty \\ \infty \\ 15 \\ 0 \end{pmatrix}, V^{(2)} = \begin{pmatrix} 30 \\ 19 \\ \infty \\ 25 \\ 20 \\ 15 \\ 0 \end{pmatrix}, V^{(3)} = \begin{pmatrix} 21 \\ 19 \\ 23 \\ 25 \\ 20 \\ 15 \\ 0 \end{pmatrix}, V^{(4)} = \begin{pmatrix} 21 \\ 19 \\ 23 \\ 25 \\ 20 \\ 15 \\ 0 \end{pmatrix}.$$

**Problema 5** Pentru algoritmul lui Dijkstra, dacă se schimbă linia 4 cu următoarea linie

**while**  $|Q| > 1$

schimbarea face ca bucla while să se execute de  $|V| - 1$  ori în loc de  $|V|$  ori. Mai este acest algoritm corect (găsește drumul de cost minim de la vârful  $s$  la fiecare vârf din graf accesibil din  $s$ )?

**Problema 6.** Modificați algoritmul lui Bellman-Ford astfel încât  $v.d = -\infty$  pentru toate vârfurile  $v$  ce aparțin unui circuit de pondere negativă de pe un drum de la  $s$  la  $v$ . Graful alăturat conține un circuit negativ accesibil din vârful sursa (sursa este vârful  $s$ ).

