Algoritmica grafurilor

Curs 5 Grafuri ponderate, Algoritmul lui Bellman-Ford

Introducere

- Un graf ponderat este un graf G = (V, E) la care i sa asociază o funcție $w : E \rightarrow R$ care atribuie o pondere w(e) fiecărei muchii $e \in E$.
- Ponderile pot reprezenta distanțe dintre noduri dar și alte metrici precum timpi, costuri, penalizări, pierderi sau alte cantități care se acumulează liniar de-a lungul unui drum și pe care vrem să le optimizăm.
- Vom studia doar grafuri ponderate simple, adică fără bucle cu cel mult o muchie de la un nod la alt nod
- Vom scrie w(x, y) ın loc de w(e) atunci cand e este muchia $x-y$ sau arcul $x \rightarrow y$.
- Deasemenea, vom presupune că w(x, x) = 0 și $w(x, y) = +\infty$ dacă nu există muchie de la x la y.

Notiuni de bază

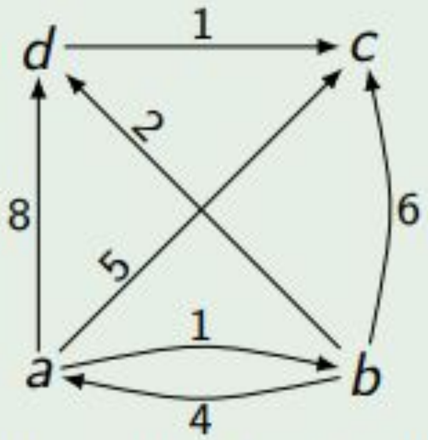
- Scriem x $\overset{\pi}{\rightsquigarrow}$ y pentru a indica faptul ca π este o cale de la x la y. Lungimea ponderata a unei liste de noduri π = [x1, x2, . . . , xn] este

$$\text{length}_w(\pi) = \sum_{i=1}^{n-1} w(x_i, x_{i+1}).$$

- Dacă n = 1 atunci π = [x1] ¸si length_w (π) = 0.
- Distanța ponderată de la x la y în G este

$$\delta_w(x, y) = \begin{cases} +\infty & \text{dacă } x \not\rightsquigarrow y, \\ \min\{\text{length}_w(\pi) \mid x \overset{\pi}{\rightsquigarrow} y\} & \text{în caz contrar.} \end{cases}$$

Lungimi si distanțe într-un graf ponderat- Exemplu



$\delta_w(x,y)$	$y=a$	$y=b$	$y=c$	$y=d$
$x=a$	0	1	4	3
$x=b$	4	0	3	2
$x=c$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
$x=d$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

$\text{length}_w([a,b,c]) = 7$,
$\text{length}_w([a,d,c]) = 9$,
$\text{length}_w([a,b,d,c]) = 4$.

Grafuri ponderate. Probleme fundamentale

P1: Drumul de cost minim între două noduri

P2: Drumul de cost minim între un nod și toate celelalte noduri ale grafului

P3: Toate drumurile de cost minim intre oricare doua noduri ale grafului intre care există cale

Principalele probleme de cost minim - Remarci

- reprezentarea prin matrice de cost-adiacentă a perechii (G,w) implică $\pi(i,j)\neq\emptyset$ pentru oricare $i,j\in V(G)$ daca $w(i,j)<\infty$ atunci $w(i,j)$ este un drum real in G iar daca $w(i,j)=\infty$ atunci $\pi[i,j]$ nu este un drum in G dar este un drum in graful obtinut din G prin adaugarea tuturor arcelor lipsa cu cost ∞.
- Algoritmii pentru rezolvarea problemei P1 se pot obtine din algoritmii pentru rezolvarea problemei P2 prin adăugarea unei conditii de oprire
- Problema P3 poate fi rezolvata prin iterarea oricarui algoritm pentru problema P2. Există totusi si solutii mai eficeinte.

Drumuri de cost minim.Aplicatii

1. Rețele de comunicatii(communication networks) Graful G(V,E) reprezintă o rețea de comunicați intre nodurile din V iar mulțimea E modelează legăturile orientate dintre noduri
   - daca $w(e) \geq 0, (\forall e \in E)$ reprezinta costul muchiei e atunci problemele eneunțate sunt probleme de cost minim.
   - daca $w(e) \geq 0, (\forall e \in E)$ reprezinta timpul necesa parcurgerii muchiei e atunci problemele P1-P3 probleme pentru determinarea drumurilor cele mai rapide
   - $w(e) \in [0,1] (\forall e \in E)$ reprezinta probabilitatea functionarii corecte a conexiunii directe dintre extremitatile lui e atunci problemele P1-P3 probleme pentru determinarea drumurilor celor mai sigure

Drumuri de cost minim.Aplicatii

2. Retele PERT- Metoda drumului critic(Critical Path Method - CPM). PERT(Path Evaluation and Review Technique) este o metodă de analiza îndeplinirea fiecărei sarcini dintr-un proiect mai complex

- Fie P={A1,A2, …An} activitati atomice ale unui mare proiect P(unde n este foarte mare), (P,<) este o multime partial ordonata , unde Ai<Aj daca activitatea Aj poate incepe dupa ce activitatea Ai s-a terminat
- Pentru fiecare activitate Ai timpul necesar finalizarii Ti este cunoscut (sau doar estimat).

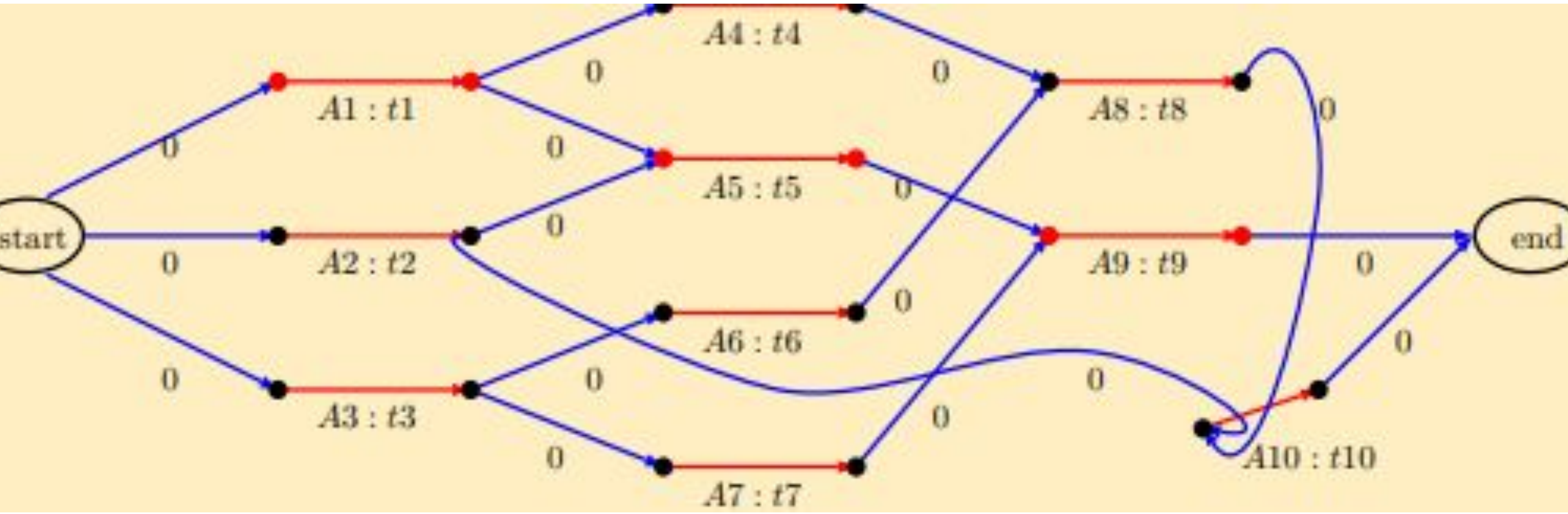
  Problema consta in a gasi o planificare a activitatilor care sa minimizeze durata totala pina la finalizare..

Drumuri de cost minim.Aplicatii

Asociem acestei probleme un graf orientat aciclic astfel:

- pentru fiecare activitate Ap (p $\in${1,2, … n}) adaugam un arc ($i_p j_p$) de cost $w(i_p,j_p)=t_p$;
- nodul i_p corespunde startului activitatii A_p iar nodul j_p este asociat finaliz[rii ei;
- daca activitatea A_k poate porni doar dupa activitatea A_p adaugam arcul $j_p i_k$ (o activitate fictiva - dummy activity)
- costructia grafului este incheiata după adăugarea unui nod s corespunzator momoentului initial al proiectului legat prin arce si_p pentru fiecare activitate A_p din care nu iese vreun arc
- In digraful obținut costul maxim al unui drum de la s la t este egal cu tiimpul minim necesar indeplinirii in intregime a poriectului
- un drum de cost maxim este numit si drum critic deaorece orice intarziere a unei activitati de pe acest drum implica o iontarziere a intregului 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

Drumuri de cost minim.Aplicatii

3. Problema rucsacului (0,1). Avem un rucsac de dimensiune b∈N, și n obiecte de dimensiuni $a_1, a_2, \ldots, a_n$.∈N.. Se cunoaste și profitul p_i ∈N al adaugarii obiectului a_i în rucsac. Se cere sa se gasească o completare a rucsacului care sa maximizeze profitul total.

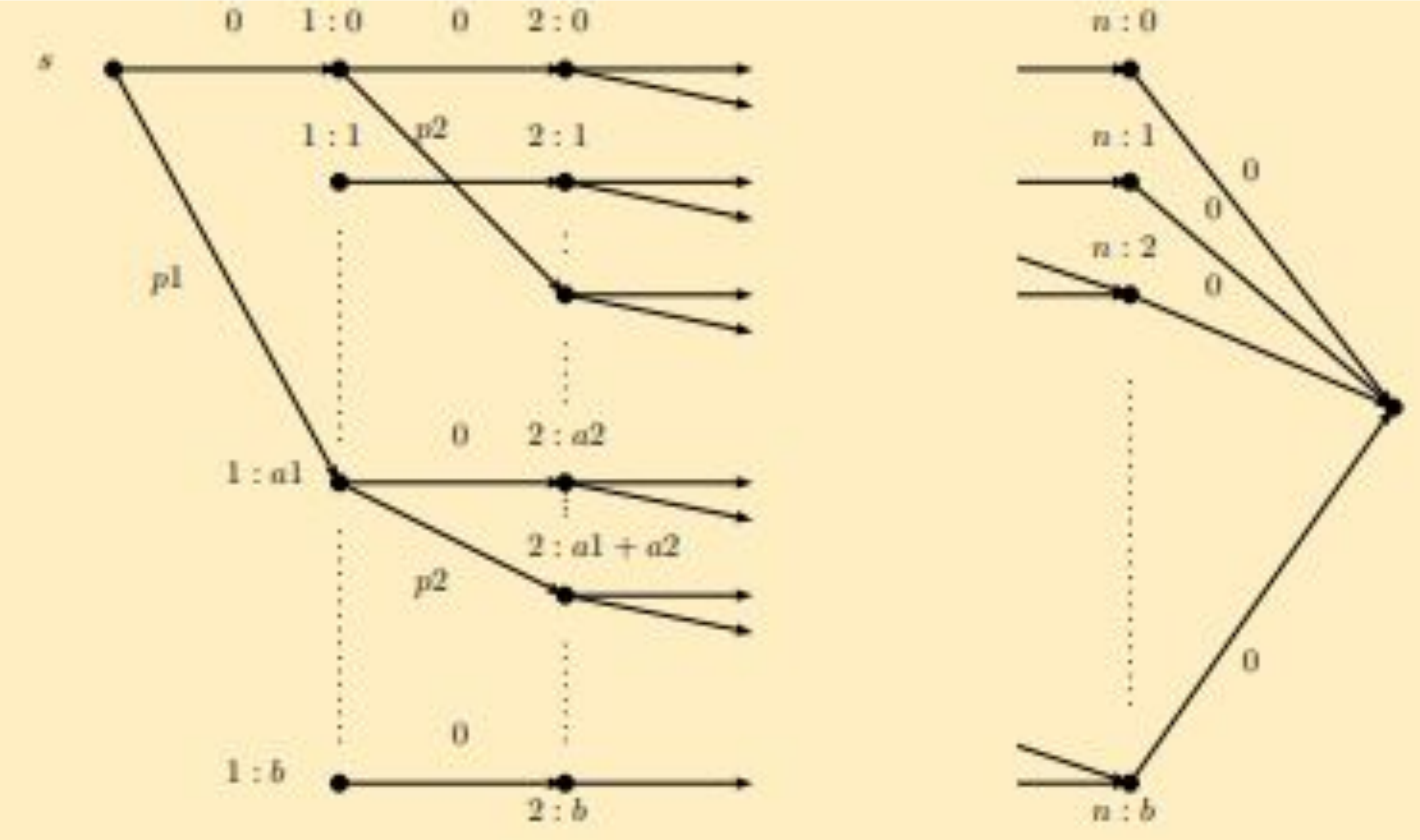
Fie x_i pentru i∈Ă1,2, … ,nÎ, o variabilă boolean cu seminificata că x_i =1 daca și numai daca obiectul i este in rucsac. Problema poate fi descrisa matematic astfel

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

Problema rucsacului

- Fie G = (V, E) unde V ={s} ∪ V_1 ∪ V_2 ∪ … ∪ V_n ∪ {t} unde V_i ={ i^1, i^2 , … ,i^b} este asociat obiectului i, i de la 1 la n.
- Arcele lui G si costurile acestora sunt :
  - s10 si s101 cu w(s10)=0 , w(s101) =p_1 (obiectul 1 este adaugat rucsacului cu profitul $p_1$si nivelul de umplere a_1 sau nu este adaugat cu profitul si nivelul de umpelre 0
  - obiectul i nu este adaugat rucsacului: dupa competarea cu primele i -1 obiecte si nivelul de umplere j se trece la umplerea cu primele i objecte,fara obiectul i; nivelul de umplere ramâne neschimbat j iar profitul adaugat este 0
  - se poate ajunge la nivelul de umplere j prin ad ugarea obiectului i la o umplere cu primele i-1 obiecte, cu nivelul de umplere j-a
  -
  -

Problema rucsacului - Exemplu



Rezolvarea problemei P2

P2: Sa se determine căi cu lungime ponderată minimă de la un nod sursă *s* la toate nodurile la care se poate ajunge din s.

Remarcă:

Dacă $\pi = [x_1, x_2, . . . , x_k]$ este o cale de la x_1 la x_k cu $length_w(\pi) = \delta_w(x_1, x_k)$, atunci pentru toți $1 \leq i \leq j \leq n$: Dacă $\pi_{i,j} = [x_i , x_{i+1}, . . . , x_j]$ atunci

$length_w(\pi_{i,j}) = \delta(x_i , k_j)$.

Altfel spus, toate subcăile unei căi cu lungime ponderată minimă au lungime ponderată minimă.

Cicluri cu lungimi ponderate negative

Muchiile e cu $w(e) < 0$ pot forma cicluri cu lungimi ponderate negative ⇒ pentru orice noduri x, y:

- Dacă există un nod z într-un ciclu c cu lungime ponderată negativă ¸si $x \rightsquigarrow z \rightsquigarrow y$ atunci nu exista $x \rightsquigarrow y$ cu lungime ponderată minimă fiindcă traversarea repetată a lui c produce căii cu lungime ponderată ce tinde la $-\infty$
- În acest caz definim $\delta_w(x, y) = -\infty$.
- În caz contrar, $\delta_w(x, y) \in R$ ¸si exista $x \rightsquigarrow y$ cu $length_w(\pi) = \delta_w(x, y)$.

Exemplu

Digraful de mai jos are cicluri cu lungime ponderată negativă:

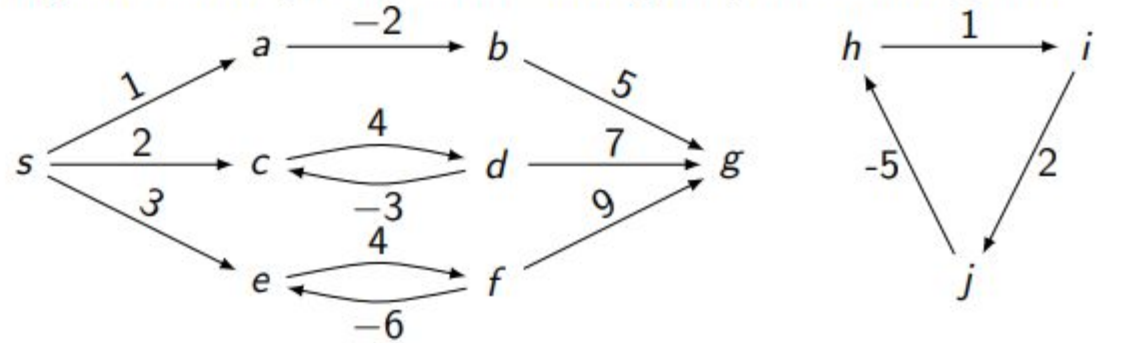
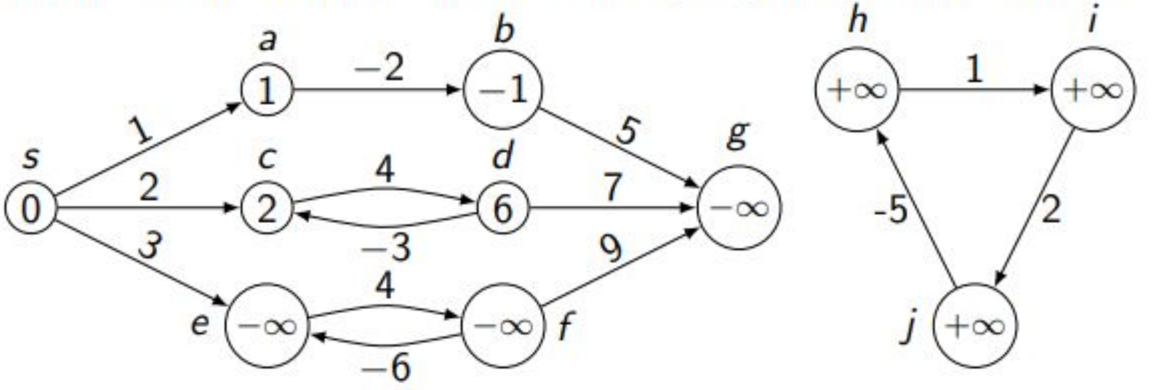


Figura următoare indică valorile lui $\delta_w(s, x)$ pentru toate nodurile x:



Cai cu lungime ponderată minimă

Fie π o cale minima de la nodul x la nodul y . Observam că:

- π nu poate conține un ciclu cu lungime ponderată strict negativă pentru că ar rezulta c $\delta_w (x, y) = -\infty$
- π nu poate conține un ciclu cu lungime ponderată zero pentru că dacă-l eliminăm din π obținem o cale de la x la y π_1 cu

$$length_w (\pi_1) < length_w (\pi) = \delta_w (x, y),$$

  contradicție.

- Putem presupune că π nu conține cicluri cu lungime ponderată 0 fiindcă ele se pot elimina din π fără s˘a-i afecteze lungimea ponderată.

Deci ne putem limita să căutăm doar căii aciclice π de al i la j cu lungime ponderată minimă. Căile aciclice conțin cel mult |V| = n noduri distincte, deci cel mult n − 1 muchii.

Algoritmul lui Bellmann-Ford si algoritmul lui Dijsktra

Algoritmii calculează reprezentarea cu predecesori a unui arbore Ts cu rădăcina s astfel încât

1. Mulțiimea de noduri a lui T_s este $S_s = \{x \in V \mid s \rightsquigarrow x\}$
2. Pentru fiecare $s \in S_s$, lista de noduri pe ramura de la s la x în T_s este o cale cu lungime ponderată minimă de la s la x în G.

Un astfel de arbore se numește arbore de căi cu lungimi ponderate minime de la s în G.

- **Algoritmul lui Dijkstra** este definit pentru grafuri ponderate cu **w(e) > 0** pentru toate muchiile e.
- **Algoritmul lui Bellman-Ford** este definit pentru cazul general, când putem avea muchii e cu w(e) < 0. Detectează eventualele cicluri cu lungime ponderată negativă la care se poate ajunge din s. În acest caz, returnează *false* pentru a semnala existența unui astfel de ciclu șii abandonează calculul arborelui T_s .

Principiul optimalităţii al optimizării dinamice

- **Enuntat de Bellman:**

    O politică este optimală dacă la fiecare moment, oricare ar fi deciziile precedente, deciziile care urmează a fi luate constituie o politică optimală în raport cu rezultatul deciziilor anterioare.

Algoritmul lui Bellmann-Ford si algoritmul lui Dijsktra

- Algoritmii operează cu:
  a. **reprezentarea cu predecesori a unui arbore A_s** cu rădăcina s și mulțimea de noduri V. Vom presupune că, pentru fiecare $x \in V$, π_x este lista de noduri pe ramura de la s la x in As .
  b. d[x]: o margine superioară a lui $length_w(\pi_x)$:

$$\forall x \in V.\delta_w(s, x) \leq length_w(s, x) \leq d[x]$$

Valorile inițiale sunt:

- p[s] = null ¸si p[x] = s pentru toți $x \in V - \{s\}$, ¸si
- d[s] = 0 ¸si d[x] = +∞ pentru tot¸i $x \in V - \{s\}$.

$V=\{s, x_1, x_2, \dots, x_n\}$

Algoritmul lui Bellmann-Ford si algoritmul lui Dijsktra

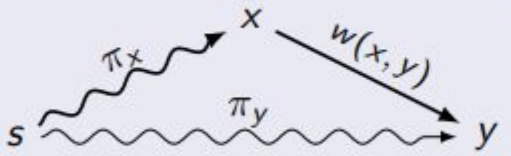
Valorile lui d[] ¸si p[] se modifica efectuand un numar finit de relaxari de muchii si garanteaza ca, atunci cand se termina:

- A_s este arbore de căi cu lungimi ponderate minime de la s în G.
- d[x] = δ_w (s, x) pentru toți x ∈ V.
- **Relaxarea unei muchii de la x la y**

  Dacă $d[x] + w(x,y) < d[y]$ și considerăm calea $\pi'_y = s \overset{\pi_x}{\rightsquigarrow} x \to y$ atunci

  $\delta_w(x,y) \leq \text{length}_w(\pi'_y) = \text{length}_w(\pi_x) + w(x,y) \leq d[x] + w(x,y) < d[y]$

  $\Rightarrow$ putem înlocui p[y] cu p[x] și d[y] cu $d[x] + w(x,y)$.




```
relax(x,y) {
   if (d[x] + w(x,y) < d[y]) {
        p[y] = x; d[y] = d[x] + w(x,y);
   }
}
```

Vizualizarea algorimiului

https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-bellman-ford/index_en.html