

Algoritmul lui BELLMAN – FORD

► Ipoteză:

Arcele pot avea și cost negativ

Graful nu conține circuite de cost negativ

(dacă există – algoritmul le va detecta \Rightarrow nu soluție)

Toate vârfurile sunt accesibile din s

Algoritmul lui BELLMAN – FORD

pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

pentru $i = 1, n-1$ executa

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

$tata[v] = u$

Complexitate: $O(nm)$

Algoritmul lui BELLMAN – FORD

- ▶ După k etape $d[u] \leq$ costul minim al unui drum de la s la u cu cel mult k arce

\Rightarrow după k etape sunt corect calculate etichetele $d[u]$ pentru acele vârfuri u pentru care **există un $s-u$ drum minim cu cel mult k arce**

Detectarea de circuite negative (curs)

Detectarea de circuite negative

- ▶ Există un circuit negativ în G (accesibil din s) \Leftrightarrow dacă algoritmul ar mai face o iterație s -ar mai actualiza etichete de distanță
- \Leftrightarrow După $n-1$ iterații există un arc uv cu
$$d[v] > d[u] + w(uv)$$

Algoritmul lui BELLMAN – FORD

pentru fiecare $u \in V$ executa

$d[u] = \infty$; $tata[u] = 0$

$d[s] = 0$

pentru $i = 1, n-1$ executa

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

$tata[v] = u$

pentru fiecare $uv \in E$ executa

daca $d[u] + w(u, v) < d[v]$ atunci

$d[v] = d[u] + w(u, v)$

$tata[v] = u$

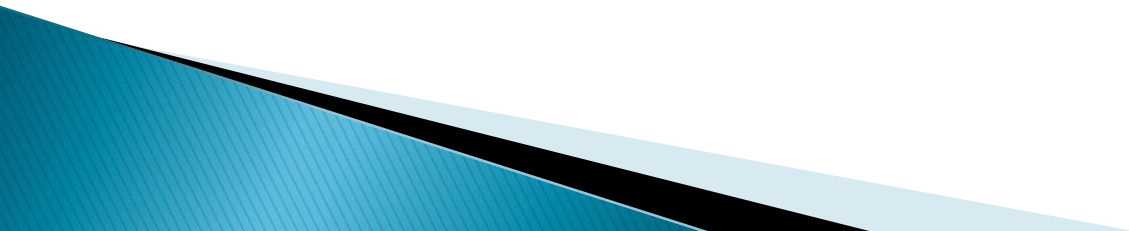
STOP, exista circuit negativ

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există circuite negative (accesibile din s) \Leftrightarrow

nu se mai fac actualizări la pasul n



Detectarea de circuite negative

Demonstrație: Arătăm că

nu există circuite negative (accesibile din s) \Leftrightarrow

nu se mai fac actualizări la pasul n

- ▶ Dacă nu există circuite negative \Rightarrow nu se mai fac actualizări la pasul n (din corectitudine)
- ▶ Dacă nu se mai fac actualizări la pasul n , pentru orice circuit $C=[v_0, \dots, v_p, v_0] \Rightarrow d[v_{i+1}] \leq d[v_i] + w(v_i v_{i+1})$

Însumând pe circuit:

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există circuite negative (accesibile din s) \Leftrightarrow

nu se mai fac actualizări la pasul n

- ▶ Dacă nu există circuite negative \Rightarrow nu se mai fac actualizări la pasul n (din corectitudine)
- ▶ Dacă nu se mai fac actualizări la pasul n , pentru orice circuit $C=[v_0, \dots, v_p, v_0] \Rightarrow d[v_{i+1}] \leq d[v_i] + w(v_i v_{i+1})$

Însumând pe circuit:

Detectarea de circuite negative

Demonstrație: Arătăm că

nu există circuite negative (accesibile din s) \Leftrightarrow

nu se mai fac actualizări la pasul n

- ▶ Dacă nu există circuite negative \Rightarrow nu se mai fac actualizări la pasul n (din corectitudine)
- ▶ Dacă nu se mai fac actualizări la pasul n , pentru orice circuit $C=[v_0, \dots, v_p, v_0] \Rightarrow d[v_{i+1}] \leq d[v_i] + w(v_i v_{i+1})$

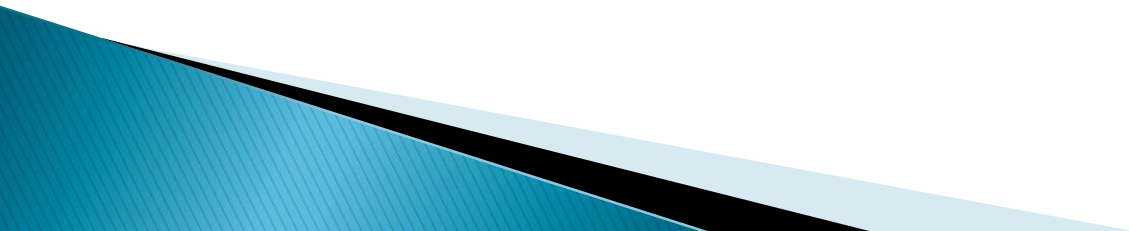
Însumând pe circuit:

$$d[v_0] + \dots + d[v_p] \leq d[v_0] + \dots + d[v_p] + w(v_0 v_1) + \dots + w(v_p v_0)$$

$$\Rightarrow 0 \leq w(v_0 v_1) + \dots + w(v_p v_0) = w(C)$$

Detectarea de circuite negative

Afişarea circuitului negativ detectat – folosind tata:



Detectarea de circuite negative

Afişarea circuitului negativ detectat – folosind tata:

- ▶ Fie v un vârf al cărui etichetă s-a actualizat la pasul suplimentar n

Detectarea de circuite negative

Afișarea circuitului negativ detectat – folosind tata:

- ▶ Fie v un vârf al cărui etichetă s-a actualizat la pasul suplimentar n
- ▶ Facem n pași înapoi din v folosind vectorul $tata$ (către s) ;
fie x vârful în care am ajuns
- ▶ Afișăm circuitul care conține pe x folosind $tata$ (din x până ajungem iar în x)

Implementare

Posibile optimizări:

- Oprește când nu s-au mai actualizat etichete
- Păstrarea unei cozi cu vârfurile a căror etichetă s-a modificat

```

queue<int> q; d[s] = tata[s] = 0;
q.push(s);
in_q[s] = 1;
while (!q.empty()) {
    u = q.front();
    q.pop();
    in_q[u] = 0;
    for(j=0;j<la[u].size();j++){
        v = la[u][j].first;
        w_uv = la[u][j].second;
        if (d[u]<infininit && d[u] + w_uv < d[v]) {
            d[v] = d[u] + w_uv ;
            tata[v] = u
        }
    }
}

```

```

queue<int> q; d[s] = tata[s] = 0;
q.push(s);
in_q[s] = 1;
while (!q.empty()) {
    u = q.front();
    q.pop();
    in_q[u] = 0;
    for(j=0;j<la[u].size();j++){
        v = la[u][j].first;
        w_uv = la[u][j].second;
        if (d[u]<infininit && d[u] + w_uv < d[v]) {
            d[v] = d[u] + w_uv ;
            tata[v] = u
            if (in_q[v]==0) {
                q.push(v);
                in_q[v] = 1;
            }
        }
    }
}

```



```

queue<int> q; d[s] = tata[s] = 0;
q.push(s);
in_q[s] = 1;
while (!q.empty()) {
    u = q.front();
    q.pop();
    in_q[u] = 0;
    for(j=0;j<la[u].size();j++){
        v = la[u][j].first;
        w_uv = la[u][j].second;
        if (d[u]<infinitt && d[u] + w_uv < d[v]) {
            d[v] = d[u] + w_uv ;
            tata[v] = u
            if (in_q[v]==0) {
                q.push(v);
                in_q[v] = 1;
            }
        }
    }
}

```

Cum detectăm circuit negativ?

```

queue<int> q; d[s] = tata[s] = 0;
q.push(s);
in_q[s] = 1;
while (!q.empty()) {
    u = q.front();
    q.pop();
    in_q[u] = 0;
    for(j=0;j<la[u].size();j++){
        v = la[u][j].first;
        w_uv = la[u][j].second;
        if (d[u]<infinitt && d[u] + w_uv < d[v]) {
            d[v] = d[u] + w_uv ;
            tata[v] = u
            if (in_q[v]==0) {
                q.push(v);
                in_q[v] = 1;
                nr[v]=nr[u]+1 //numarul de arce din drum
                //nr[v]++; //de cate ori a fost inserat in q
                if (nr[v] > n-1)
                    return v;
            }
        }
    }
}

```

```
q = deque(); d[s] = tata[s] = 0
q.append(s)
in_q[s]=1
while len(q)>0:
    u = q.popleft()
    in_q[u]=0

    for (v,w_uv) in la[u]:
        if d[u]<infinitt and d[u]+w_uv<d[v]:
            d[v] = d[u]+w_uv
            tata[v] = u
            if in_q[v]==0:
                q.append(v)
                in_q[v]=1

return None,d,tata
```

