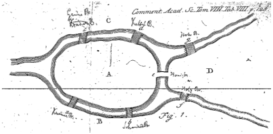
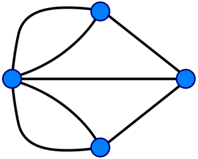




Algoritmica grafurilor
III. Drumuri în grafuri

Mihai Suciu

Facultatea de Matematică și Informatică (UBB)
Departamentul de Informatică

Martie, 11, 2025

Continut

1. Sortare topologica
2. Componente tare conexe
3. Drum de lungime minima
   - Sursa unica
     - Bellman-Ford
     - Grafuri orientate aciclice
     - Dijkstra
   - versiuni Floyd-Warshal

DFS

DFS(G**)**

```
for fiecare vârf u ∈ G.V do
    u.color = alb
    u.π = NIL
time = 0
for fiecare u ∈ G.V  do
    if u.color == alb then
        DFS_VISIT(G,u)
```

DFS_VISIT(G, u**)**

```
time = time + 1
u.d = time
u.color = gri
for fiecare v ∈ G.Adj[u] do
    if v.color == alb then
        v.π = u
        DFS_VISIT(G,v)
u.color = negru
time = time + 1
u.f = time
```

DFS (II)

Teorema (Teorema parantezelor)

în orice căutare în adâncime a unui graf $G = (V, E)$(orientat sau neorientat), pentru orice două vârfuri u și v, exact una din următoarele trei afirmații este adevărată:

- *intervalele $[u.d, u.f]$ și $[v.d, v.f]$ sunt total disjuncte*
- *intervalul $[u.d, u.f]$ este conținut în întregime în intervalul $[v.d, v.f]$ iar u este descendent al lui v în arborele de adâncime*
- *intervalul $[v.d, v.f]$ este conținut în întregime în intervalul $[u.d, u.f]$ iar v este descendent al lui u în arborele de adâncime*

DFS - clasificarea muchiilor

- pentru un graf $G = (V, E)$, fie $(u, v) \in E$, în funcție de timp tipul arcelor pentru DFS:

tip arc	d	f
t (tree)	$u.d < v.d$	$u.f > v.f$
b (back)	$u.d > v.d$	$u.f < v.f$
f (forward)	$u.d < v.d$	$u.f > v.f$
c (cross)	$u.d > v.d$	$u.f > v.f$

- $u.d$ marchează timpul când a fost descoperit vârful u
- $u.f$ marchează timpul când a fost explorat vârful u

Sortare topologica

sortare_topologică(G)

1: apel DFS(G) pentru a determina timpii $v.f, v \in V$
2: sortare descrescătoare în funcție de timpul de finalizare (când fiecare vârf e terminat, e inserat într-o listă înlănțuită)
3: **return** lista înlănțuită de vârfuri

Componente tare conexe

componente_tare_conexe(G)

1: apel DFS(G) pentru a determina timpii $v.f, v \in V$
2: determină G^T
3: apel DFS(G^T) dar în bucla principala a DFS nodurile sunt sortate descrescător după $v.f$
4: fiecare arbore din pădurea găsită de DFS în pasul 3 este o componentă tare conexă

Probleme de drum de lungime minimă

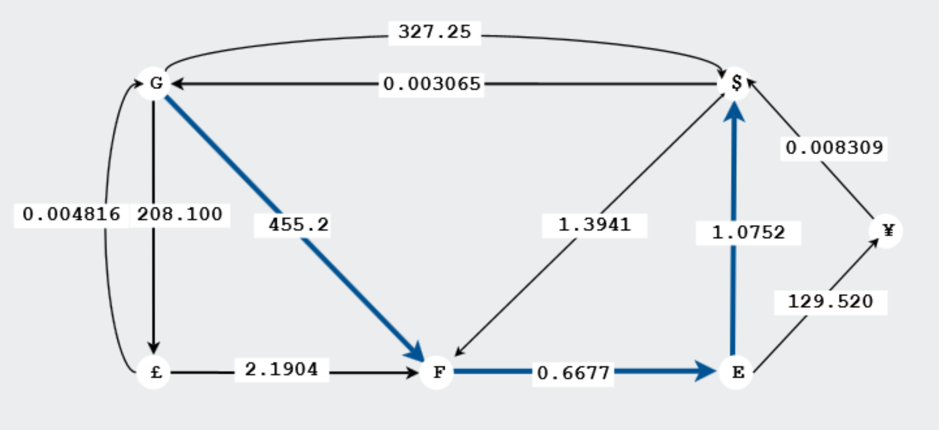
- se poate defini drumul cu pondere minimă $\delta(u, v)$ pentru un drum de la u la v:

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

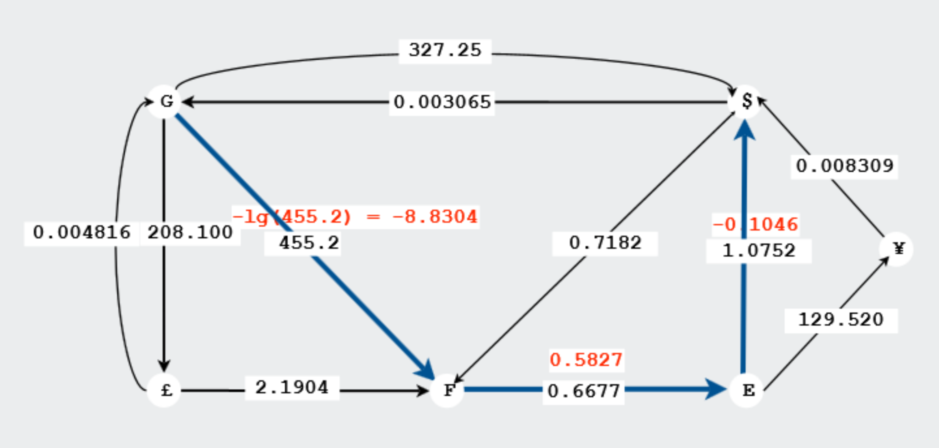
Exemplu

Currency	£	Euro	¥	Franc	$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

Exemplu (II)


327.25
G
0.003065
$
0.008309
0.004816
208.100
455.2
1.3941
1.0752
¥
129.520
£
2.1904
F
0.6677
E

Exemplu (III)


327.25
G
0.003065
$
0.008309
0.004816
208.100
-lg(455.2) = -8.8304
455.2
0.7182
-0.1046
1.0752
¥
129.520
0.5827
£
2.1904
F
0.6677
E

Algoritmul Bellman-Ford

- algoritmul Bellman-Ford rezolvă problema drumului minim de la un nod sursă s pentru cazul general când avem și ponderi negative

Bellman_Ford(G, w, s))

INITIALIZARE_S(G,s)
for $i = 1$ **la** $|V| - 1$ **do**
    for fiecare arc $\{u, v\} \in E$ **do**
        RELAX(u, v, w)
for fiecare arc $\{u, v\} \in E$ **do**
    if $v.d > u.d + w(u, v)$ **then**
        return FALSE
return TRUE

Bellman-Ford (II)

INITIALIZARE_S(G,s)

for $v \in V$ **do**
    $v.d = \infty$
    $v.\pi = NIL$
$s.d = 0$

RELAX(u, v, w)

if $v.d > u.d + w(u, v)$ **then**
    $v.d = u.d + w(u, v)$
    $v.\pi = u$

- Exemplu - click

Grafuri orientate aciclice

drum_minim_dag(G)

sortate_topologica(G)
INITIALIZARE_S(G,s)
for fiecare vârf v sortat topologic **do**
    for $v \in G.Adj[u]$ **do**
        RELAX(u,v,w)

Algoritmul Dijkstra

Dijkstra_queue(G)

INITIALIZARE_S(G,s)
$S = \varnothing$
$Q = V$
while $Q \neq \varnothing$ **do**
    $u = \text{EXTRACT\_MIN}(Q)$
    $S = S \cup \{u\}$
    for $v \in G.Adj[u]$ **do**
        RELAX(u,v,w)

Floyd-Warhsall

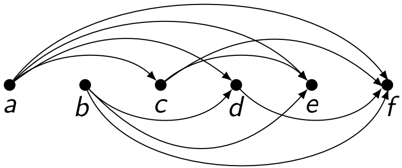
```
FloydWarshall(D_0)
    D := D_0
    for k := 1 to n do
        for i := 1 to n do
            for j := 1 to n do
                if d_ij > d_ik + d_kj then
                    d_ij := d_ik + d_kj
                    p_ij := p_kj
    return D, p
```

Floyd-Warshall pentru a determina nr de drumuri

FW(A, n)

1. $W \leftarrow A$
2. **for** $k \leftarrow 1$ **to** n
3. **for** $i \leftarrow 1$ **to** n
4. **for** $j \leftarrow 1$ n
5. **do** $w_{ij} \leftarrow w_{ij} + w_{ik}w_{kj}$
6. **return** W

Exemplu



Exemplu (II)

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

Rezultat FW:

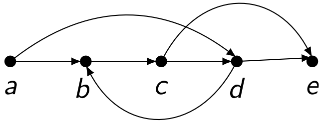
$$W = \begin{pmatrix} 0 & 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Floyd-Warshal-latin

Floyd–Warshall–Latin($\mathcal{A}$, n)

1. $\mathcal{W} \leftarrow \mathcal{A}$
2. **for** $k \leftarrow 1$ **to** n
3.   **for** $i \leftarrow 1$ **to** n
4.   **for** $j \leftarrow 1$ **to** n
5.   **if** $W_{ik} \neq \emptyset$ and $W_{kj} \neq \emptyset$
6. $W_{ij} \leftarrow W_{ij} \cup W_{ik} \cdot' W_{kj}$
7. **return** $\mathcal{W}$

Exemplu



Exemplu (II)

$$\begin{pmatrix} \emptyset & \{adb, ab\} & \{adbc, abc\} & \{abcd, ad\} & \{ade, adbce, abcde, abce\} \\ \emptyset & \emptyset & \{bc\} & \{bcd\} & \{bcde, bce\} \\ \emptyset & \{cdb\} & \emptyset & \{cd\} & \{cde, ce\} \\ \emptyset & \{db\} & \{dbc\} & \emptyset & \{dbce, de\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}.$$