Matice sousednosti NG

$\mathbf{V} = [\, v_{ij} \,]$ celočíselná čtvercová matice řádu $|U|$

$\mathbf{v_{ij}} = |\, \rho^{-1}([\mathbf{u_i}, \mathbf{u_j}]) \,|$ ... tedy počet hran mezi u_i a u_j

?Jaké vlastnosti má matice sousednosti?

? Smyčky, rovnoběžné hrany?

$\mathbf{V} = \mathbf{V}^T$

$\mathbf{V}^r = [v_{ij}^{(r)}]$ ... počet sledů délky r mezi u_i a u_j

$\mathbf{A} \,.\, \mathbf{A}^T = \mathbf{V} + \mathbf{D}$, $\mathbf{D} = [\, d_{ii} \,]$, kde $d_{ii} = \delta(u_i)$

Matice incidence OG

A = **[** a_{ik} **]** celočíselná obdélníková matice typu |U| x |H|

$$a_{ik} = \begin{cases} 1 & \text{.... hrana } h_k \text{ vystupuje z uzlu } u_i \\ -1 & \text{.... hrana } h_k \text{ vstupuje do uzlu } u_i \\ 0 & \text{.... jinak} \end{cases}$$

? Co nám říká matice incidence o grafu?

Vlastnosti matice incidence OG jsou podobné jako jsme viděli u NG.

Matice sousednosti OG

$\mathbf{V} = [\, v_{ij} \,]$ celočíselná čtvercová matice řádu |U|

$v_{ij} = |\, \sigma^{-1}(\, [u_i, u_j]\,)\, |$ ... tedy **počet hran** z u_i do u_j

?Jaké vlastnosti má matice sousednosti?

$\mathbf{V}^* = \sum \mathbf{V}^i$, i=0, ... d, kde d = min(|H|, |U|-1)

matice dostupnosti $\mathbf{R} = [r_{ij}]$, $r_{ij} = 1$ pokud $u_i \to^* u_j$, jinak 0

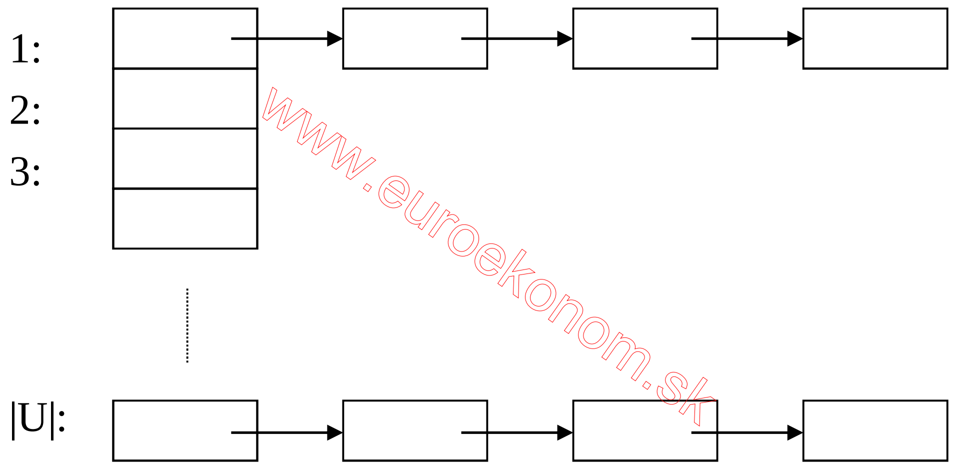
rozklad na silné komponenty $\mathbf{R}' = \mathbf{R} \cap \mathbf{R}^T$

$\mathbf{A} \,.\, \mathbf{A}^T = \mathbf{D} - \mathbf{V} - \mathbf{V}^T$, $\mathbf{D} = [\, d_{ii} \,]$, kde $d_{ii} = \delta^+(u_i) + \delta^-(u_i)$

Spojová reprezentace grafu

N.G. - seznamy sousedů O.G. - seznamy následníků

Adj[u]



Srovnání paměťové složitosti:

NG A: |U|.|H| (bitů!) V: |U|.|U| (integer ? Boolean)

Adj: |U| + 2.|H| **OG** Adj: |U|+|H|

Prohledávání grafu do šířky

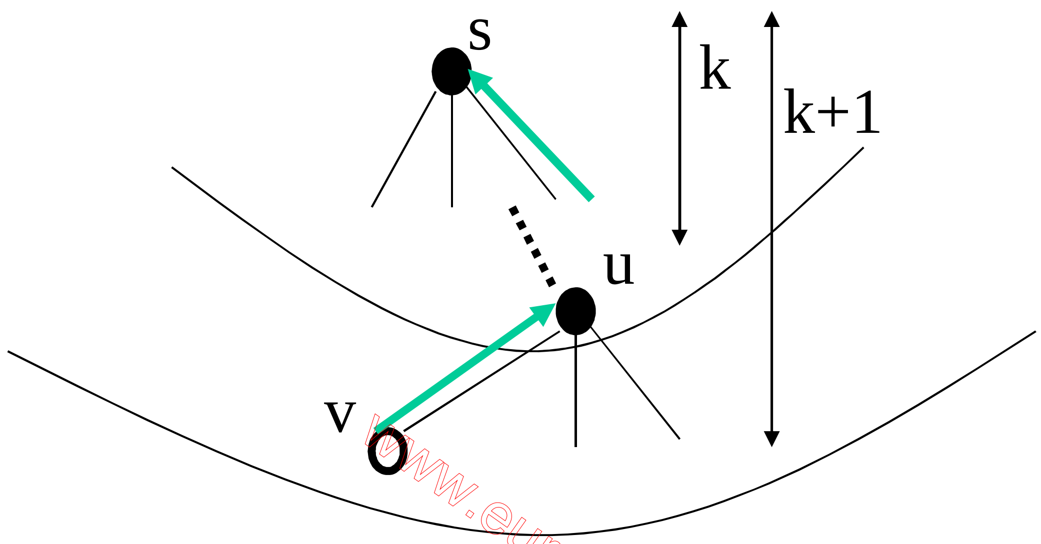
BFS - Breadth-First Search

Je zadán graf $G = \langle H,U,\sigma \rangle$ (není podstatné, zda NG nebo OG) a jeho uzel $u \in U$.

Prohledáním do šířky dostaneme strom (nejkratších) $s \to^* u$ cest

Stavy uzlů:

- **FRESH** - nový (dosud neobjevený) uzel
- **OPEN** - právě objevený („nadějný“) uzel
- **CLOSED** - vyčerpaný uzel



Datové struktury:

stav[u] - FRESH / OPEN / CLOSED

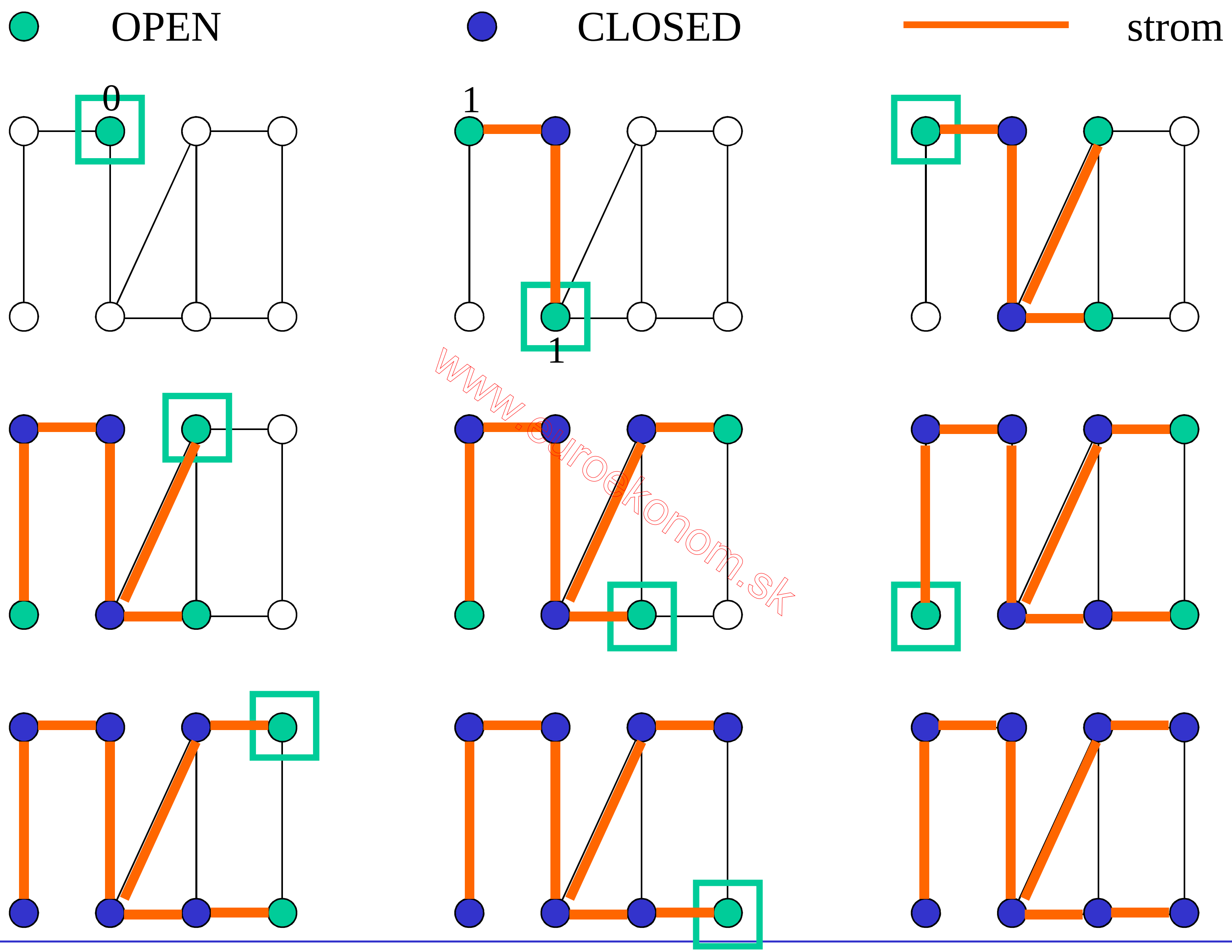
d[u] - zjištěná vzdálenost s $\rightarrow^*$ u

p[u] - předchůdce uzlu u (viz ⟶)

fronta OPEN uzlů

BFS (G, s) . . . pseudokód

```
for každý uzel u ∈ U-{s} do
        stav[u]:=FRESH; d[u]:=∞; p[u]:=nil
stav[s]:=OPEN; d[s]:=0; p[s]:=nil;
InitQueue; Enqueue(s);
while not EmptyQueue do
        u:=QueueFirst;
        for každé v ∈ Adj[u] do
            if stav[v]=FRESH
                then stav[v]:=OPEN; d[v]:=d[u]+1;
                     p[v]:=u; Enqueue(v);
        Dequeue;
        stav[u]:=CLOSED
```



OPEN
CLOSED
strom
0
1
1

Vlastnosti BFS algoritmu

Složitost: cykl 1 - 4 .... **O(|U|)**

operace s frontou O(1) na uzel $\Rightarrow$ celkem **O(|U|)**

cykly 5 + 7 pro každého souseda ... **O(|H|)**

$\Rightarrow$ **O(|U| + |H|)**

Vlastnosti:

- pro každou hranu $h \in H$: $h=(u, v)$ platí **$d(s,v) \leq d(s, u)+1$**
- **$d[v] \geq d(s, v)$**
- pokud fronta obsahuje uzly $v_1, v_2, \ldots, v_r$, potom platí

  $d[v_r] \leq d[v_1] + 1$

  $d[v_i] \leq d[v_{i+1}]$ pro $i=1,2,\ldots,r-1$
- konec: $d[v]=d(s,v)$ a $(p[v],v)$ je hrana nejkratší cesty $s \rightarrow^* v$

Prohledávání grafu do hloubky

DFS - Depth-First Search

Výsledkem bude DF strom (nebo les)

Uzly jsou opět FRESH, OPEN nebo CLOSED, ale mají

časové značky s hodnotami 1 ... 2*|U|

- **d[u]** přidělí se uzlu při jeho otevření
- **f[u]** přidělí se uzlu při jeho uzavření (tzn. d[u]<f[u])

Postup prohledávání

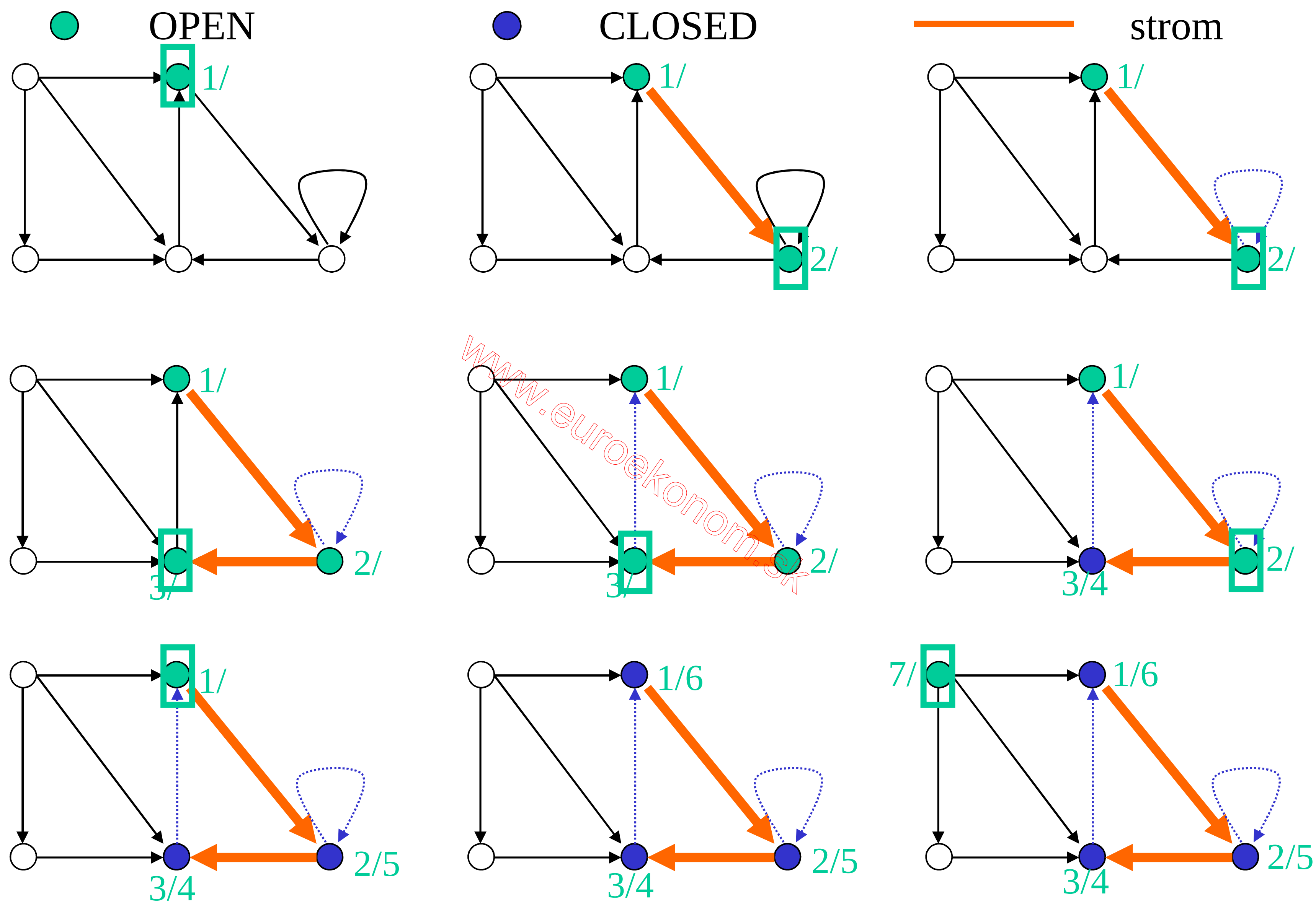
„Pokud existuje FRESH následník, přejdi na něj, jinak uzel uzavři a vrať se k předchůdci.“

DFS (G) pseudokód

```
for každý uzel u ∈ U  do
        stav[u]:=FRESH; p[u]:=nil;
i:=0;
 for každý uzel u ∈ U  do
        if stav[u] = FRESH then DFS-Projdi(u);
```


DFS-Projdi(u)

```
stav[u]:=OPEN; i:=i+1; d[u]:=i;
for každý uzel v ∈ Adj[u]  do
        if stav[u] = FRESH
            then p[v]:=u; DFS-Projdi(v);
stav[u]:=CLOSED; i:=i+1; f[u]:=i;
```


OPEN
CLOSED
strom
1/
2/
3/
3/4
2/5
1/6
7/

Složitost DFS

cykly 1 - 2 a 4 - 5 ... $O(|U|)$

DFS-Projdi ... volá se $|U|$/krát

cykl 2 - 4 ... $|Adj[u]|$-krát ,

tedy celkem $\sum|Adj[u]| = \Theta(|H|)$

$\Rightarrow$ **$O(|U|+|H|)$**

Vlastnosti DFS

- **závorkový teorém:** nastává právě jedna z možností
  - $\langle d[u], f[u]\rangle \cap \langle d[v], f[v]\rangle = \emptyset$ (vztah intervalů)
  - $\langle d[u], f[u]\rangle \subset \langle d[v], f[v]\rangle$ - potom $v \rightarrow u$ (v DFS stromu)
  - $\langle d[v], f[v]\rangle \subset \langle d[u], f[u]\rangle$ - potom $u \rightarrow v$ (v DFS stromu)
- **vnořování intervalů:**

  $u \rightarrow v$ (v DFS lese) $\Leftrightarrow d[u] < d[v] < f[v] < f[u]$
- **teorém o nové cestě:**

  $u \rightarrow v$ (v DFS lese) $\Leftrightarrow$ v okamžiku d[u] existuje „nová“ cesta P(u, v)
- **stromové / zpětné / dopředné / příčné hrany**

? Jak se pozná typ hrany (u, v)?

FRESH v ⇒ stromová

OPEN v ⇒ zpětná

CLOSED v ⇒ dopředná nebo příčná ?

$d[u] < d[v]$ **$d[u] > d[v]$**

? DFS neorientovaného grafu ?

Orientace hrany se provede **podle prvního průchodu**, takže dostaneme jen **stromové** a **zpětné** hrany.

Topologické uspořádání

Top-Sort-1 (G)

- $S := \emptyset$
- prováděj DFS(G) a v okamžiku f[u] ulož uzel u na začátek seznamu S
- S obsahuje uzly v topologickém uspořádání

<u>V:</u> G je acyklický $\Leftrightarrow$ DFS(G) neobjeví zpětnou hranu

Top-Sort-2 (G) eliminací kořenů - viz dříve

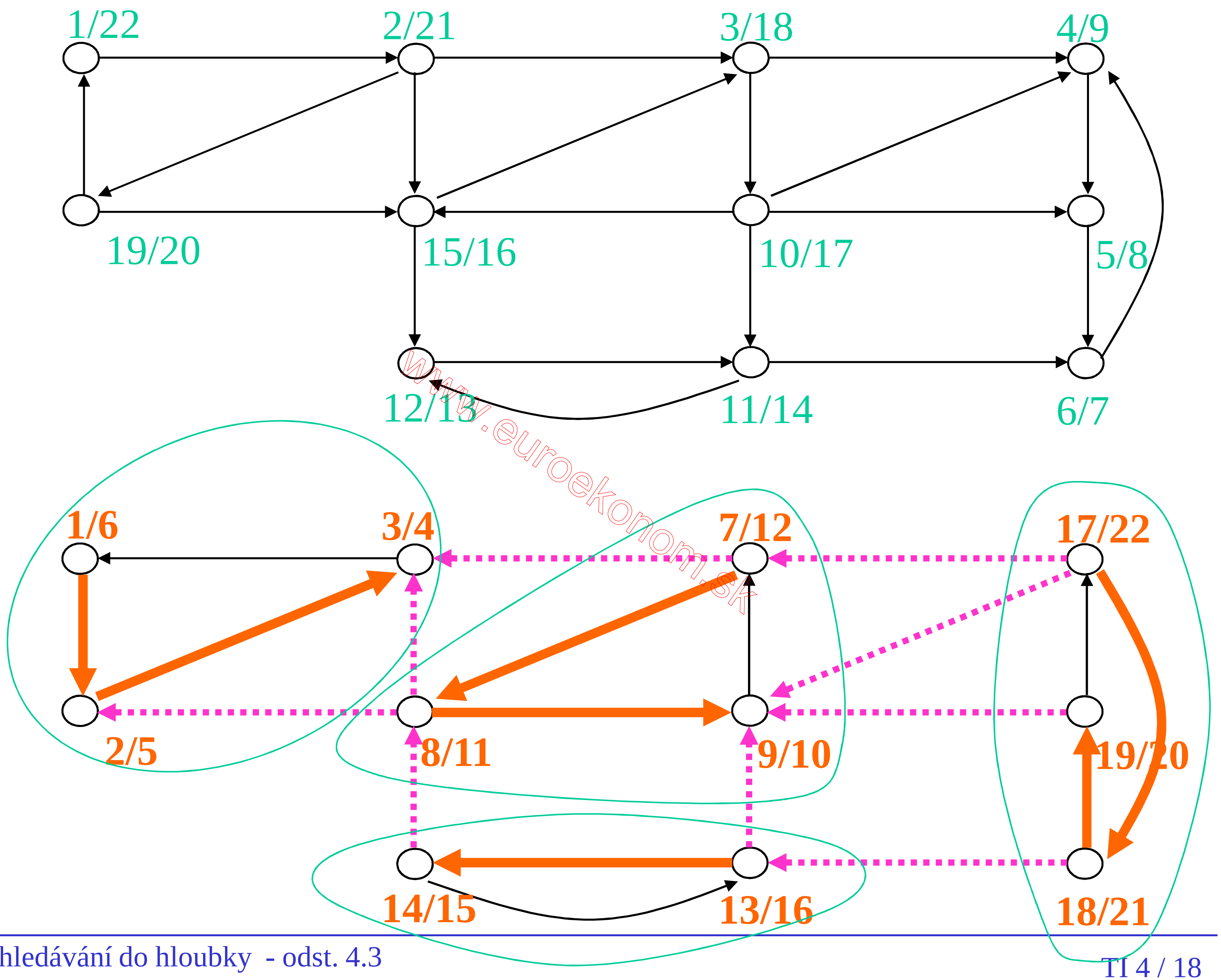
$\delta[u] = \delta_G[u]$

M - množina kořenů (fronta)

Silné komponenty

S-COMP (G)

- pomocí DFS(G) se určí f[u] pro všechny $u \in U$
- vytvoří se G^- (opačně orientovaný graf)
- provede se DFS(G^-) s tím, že uzly v hlavním cyklu se berou v klesajícím pořadí f[u]
- stromy DFS-lesa určují silné komponenty


1/22
2/21
3/18
4/9
19/20
15/16
10/17
5/8
12/13
11/14
6/7
1/6
3/4
7/12
17/22
2/5
8/11
9/10
19/20
14/15
13/16
18/21