Start 11.4.

Základní grafové algoritmy

konečná množina vrcholů

$G = (V, E)$

množina hran

$|V| = n$

$|E| = m$

E {
- neorientované: $E \subseteq \binom{V}{2}$, $\{u,w\}$, uw
- orientované: $E \subseteq V \times V$, (u,w), uw, $uw \in E$

smyčka

$v \in V$: sousedi

$N(v) = \{u \in V : uv \in E\}$

$\deg(v) = |N(v)|$

$N^{IN}(v) = \{u \in V : uv \in E\}$

$N^{OUT}(v) = \{u \in V : vu \in E\}$

$\deg^{OUT}(v) := |N^{OUT}(v)|$

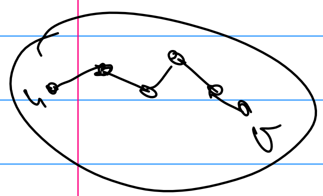
$\deg^{IN}(v)$ $|N^{IN}(v)|$

---

vstup: $G = (V, E)$, $u, v \in V$ (neorientovaný)

otázka: „lze v grafu G se dostat z u do v"

G

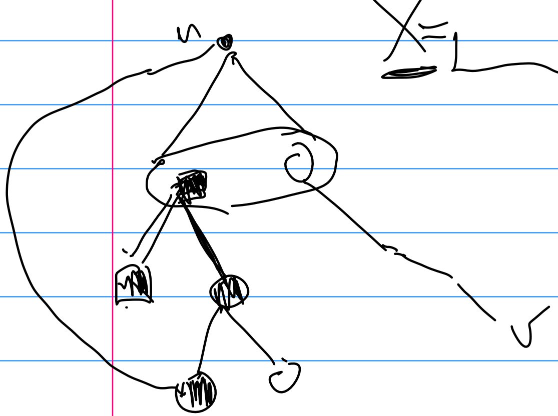


$\exists$ cesta v G z u do v?

N 1) $\forall e \in E$ $e = uv$? YES → RETURN YES

Y 2) $\deg(u) > 0$ & $\deg(v) > 0$? NO → RETURN NO

PROHLEDÁVÁNÍ DO HLOUBKY // DFS



$X =$

označ u; vlož(X, u)

while TRUE

0) if X prázdné RETURN NO; $x :=$ vrchol X

1) if $x = v$: RETURN YES

2) if $\exists y$ neoznačený soused x:
 vlož(X, y)
 označ y
 else: odeber(X, x)

DFS(x)
 označ x
 if $x = v$: RETURN YES
 for y soused:
  if y neoznačený: DFS(y)

DFS(u)

Depth first search: VSTUP $G=(V,E)$ orientovaný, $u \in V$

informace: 1) $\forall v \in V$: stav – nenalezený / otevřený / uzavřený

2) $\forall v$: $in(v)$: čas objevení v; $out(v)$: čas opuštění v

3) T: = čas

DFS_INIT(G,u)

1) $\forall v \in V$: stav(v) = nenalezený

$in(v) = out(v) = -1$

2) $T := 0$

3) DFS_RECURSIVE(u)

DFS_RECURSIVE(x):

1) stav(x) := otevřený
2) $T += 1$
3) $in(x) = T$
4) $\forall y \in N^{OUT}(x)$, IF stav(y) = nenalezený: DFS_RECURSIVE(y)
5) stav(x) := uzavřený
6) $T += 1$
7) $out(x) = T$

$\deg^{OUT}(x)$

PAMĚŤ: $O(n)$

ČAS: $O(n \cdot n) = O(n^2)$

$\Omega(n+m)$, $O(n+m) \Rightarrow \Theta(n+m)$

FAKT: $> 99\%$ grafů na n vrcholech má $\geq 0.49 \cdot \binom{n}{2}$ hran

většina grafů v reálném světě má $o(n^2)$ hran

čas $= O\left(n + \sum_{x \in V} \deg^{OUT}(x)\right) = O(n+m)$

DATA

REPREZENTACE grafů $G=(V,E)$ BÚNO $V = \{1, \dots, n\}$

- seznam sousedů: sousedé$[n]$

sousedé$[i]$:= seznam sousedů j t.ž. $ij \in E$

$i \longrightarrow j$

- matice sousednosti $n \times n$

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

$a_{ij} = \begin{cases} 0 & ij \notin E \\ 1 & ij \in E \end{cases}$

Lemma (korektnost):

1) DFS skončí, a co víc, v čase $O(n+m)$

2) DFS "najde" (označí) právě ty vrcholy w: t.ž. z u do w vede cesta

Dk (2): "$\Rightarrow$" indukcí dle T:

$T=1$: $u \to y \in N^+(u)$

$T \geq 2$: y se označil $x \to y \in N^+(x)$

z algoritmu víme, že x označí y

a z indukce víme, že $u \to x$ vede cesta

$u \to x \to y$

Lemma: když z $u \to x$ vede cesta $x \to y$ vede cesta

tak $u \to y$ vede cesta

"$\Leftarrow$" $\exists w$ t.ž. z $u \to w$ vede cesta, ale stav(w) je nenalezený na konci DFS

tak vybereme w_0 ze všech takových w že vzdálenost $u \to w_0$ je nejkratší

P nejkratší cesta z $u \to w_0$ $u \to v_1 \to v_2 \to \dots \to \boxed{v_k} \to w_0$

v_k byl označen DFS, jinak spor s volbou w_0

Podívejme se na volání DFS_RECURSIVE(v_k),

o něm speciálně zavolání DFS_RECURSIVE(w_0),

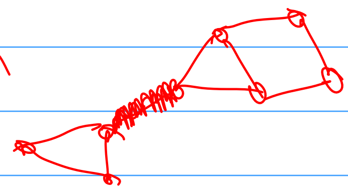
ale to je by nebylo označilo w_0

Hledání mostů (hranový řez velikosti 1)

G neorientovaný, (V,E), #komponent souvislosti $= k$

DEF: $e \in E$ je most pokud $G-e$ $(G-e = (V, E\setminus\{e\}))$ má $> k$ komponent souvislosti $(k+1)$

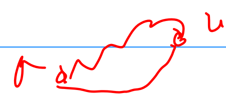
GÓL: algoritmus na hledání mostu



$$O(m(n+m)) = O(nm+m^2)$$

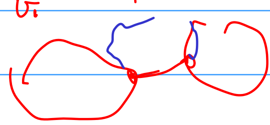
$\exists\, u,v \in V$ tž. $uv \in E$ ale existuje právě jedna cesta z u do v

BTW $uv \in E$ & existuje další cesta $v \to u$
$\Rightarrow uv \xrightarrow{P} u$ jsou cyklus v G

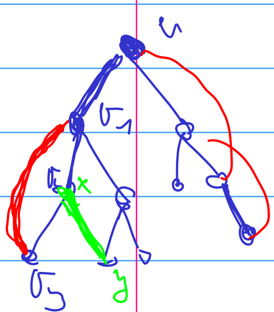


uv je most $\Leftrightarrow$ neexistuje kružnice v G obsahující uv v G

Dk: "$\Rightarrow$" $\neg$ existuje kružnice v G obsahující uv ale pak v $G-uv$ existuje cesta z u do v ↯



"$\Leftarrow$" kružnici obsahující uv, tak $G-uv$ má stejný #komponent jako G, speciálně uv není most



stromové hrany

zpětné

ZELENÁ NE $x \sim y$

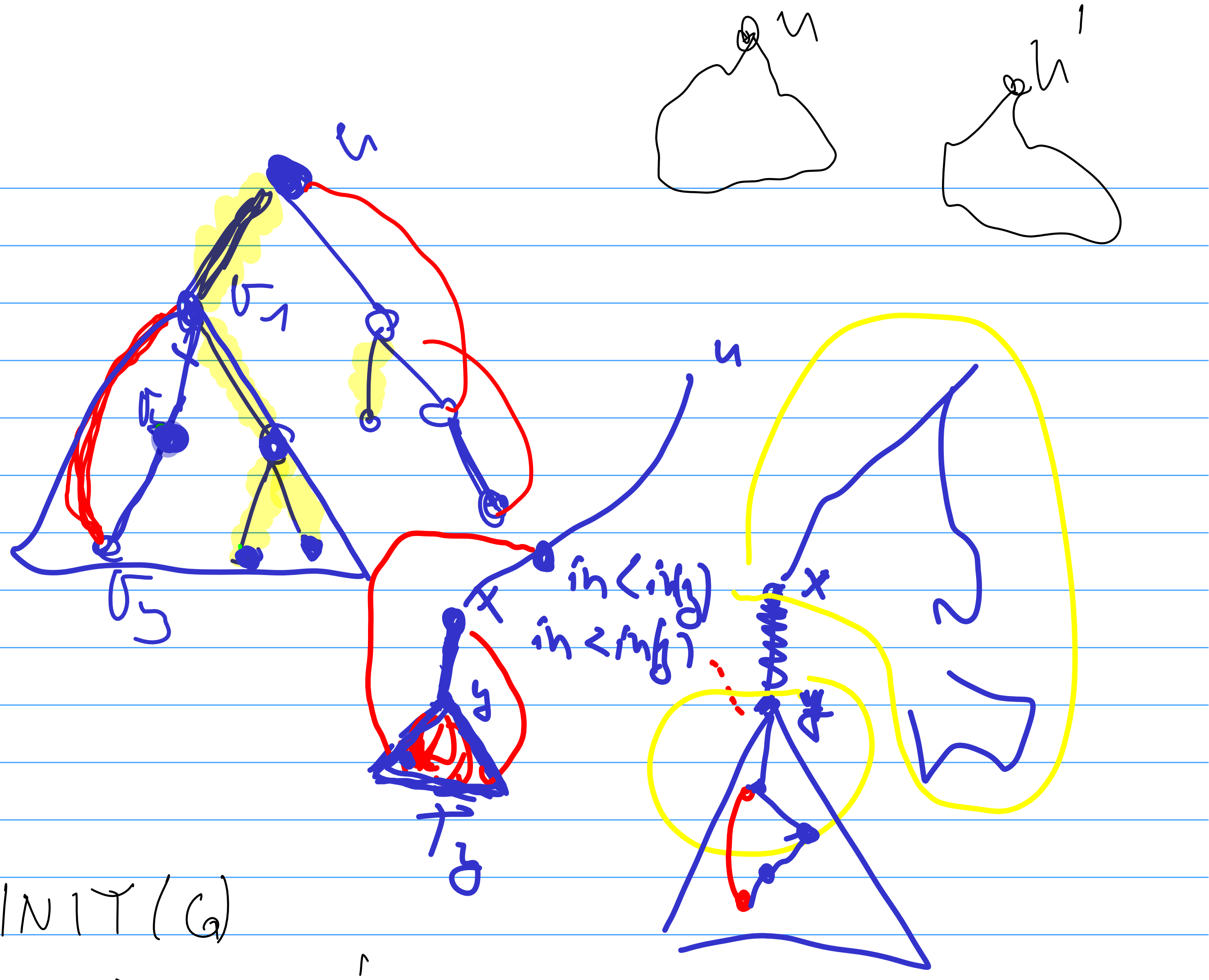
$out(x) < in(y)$

$xy \in E$

$in(x) < in(y) < out(y) < out(x)$

zpětné nejsou nikdy most

in
out



MOST_INIT(G)

1) $\forall v \in V$: stav(v) = nenalezený
   in(v) = out(v) = −1
   low(v) = +∞

$\Theta(n+m)$

2) T := 0

3) $\forall u \in V$: if stav(u) = nenalezený: MOST_RECURSIVE(u, ∅)

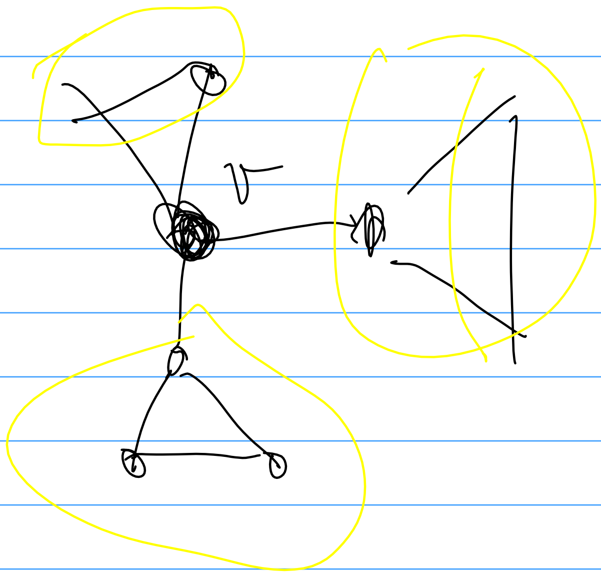
MOST_RECURSIVE(x, otec)

1) stav(x) := otevřený
2) T += 1
3) in(x) = T
4) $\forall y \in N(x) \setminus \{otec\}$: IF stav(y) = nenalezený: MOST_RECURSIVE(y, x)
   LOW(x) = min(low(x), low(y))
   if low(y) > in(x): xy most
   ELSE: LOW(x) := min(low(x), in(y))
5) stav(x) := uzavřený
6) T += 1
7) out(x) = T

DCV:

Artikulace grafu $G=(V,E)$

$v \in V$ t.ž. $G-v$ má více komponent souvislosti než G



Alg na vypsání všech artikulací v G v čase $\Theta(m+n)$