Průzkum grafu

Everything on earth can be found, if only you do not let yourself be put off searching. *Philemon of Syracuse (ca. 360 BC–264 BC)*

- pro daný graf G a vrchol s grafu je cílem
  - navštívit všechny vrcholy grafu dosažitelné z vrcholu s, resp.
  - navštívit všechny vrcholy grafu
- průzkum realizovat maximálně efektivně, tj. se složitostí $\mathcal{O}(V + E)$ (*vyhnout se opakovaným návštěvám*)
- jaké další informace o grafu zjistíme v průběhu průzkumu???

Průzkum grafu do šířky vs do hloubky

Theseus si před bludištěm uváže jeden konec nitě na strom a vsoupí dovnitř. V prvním vrcholu (křižovatce) si vybere jednu možnou cestu / hranu a projde po ní do dalšího vrcholu. Aby Theseus neměl zmatek v tom, které hrany už prošel, tak si všechny hrany, které prochází označuje křídou – a to na obou koncích. V každém vrcholu, do kterého Theseus dorazí, provede následující:

- *Pokud na zemi najde položenou niť, tak ví, že už ve vrcholu byl a že se do něj při namotávání nitě zase vrátí. Odloží tedy další prozkoumávání tohoto vrcholu na později, provede čelem vzad a začne namotávat niť na klubko. To ho dovede zpátky do předchozího vrcholu.*
- *Pokud na zemi žádnou niť nenajde, tak se vydá první možnou neprošlou hranou. Pokud by taková hrana neexistovala, tak je vrchol zcela prozkoumán. V tom případě Theseus neztrácí čas a začne namotávat niť na klubko. Tím se dostane zpátky do předchozího vrcholu.*

Tímto postupem prozkoumá celé bludiště a nakonec se vrátí do výchozího vrcholu.[6]

[6] Jakub Černý: Základní grafové algoritmy `http://kam.mff.cuni.cz/~kuba/ka`

Průzkum grafu do šířky vs do hloubky

implementace

křída proměnná označující jestli jsme hranu prošli

klubko položená niť vyznačuje cestu z výchozího do aktuálního vrcholu, cestu si pamatujeme jako posloupnost vrcholů na této cestě. Pro uložení cesty použijeme zásobník. Odmotávání nitě odpovídá přidání vrcholu do zásobníka. Namotávání nitě při návratu zpět odpovídá odebrání vrcholu ze zásobníku.

Průzkum grafu do šířky vs do hloubky

Tento průchod (prohledání grafu) si můžeme představit tak, že se do výchozího vrcholu postaví miliarda trpaslíků a všichni naráz začnou prohledávat graf. Když se cesta rozdělí, tak se rozdělí i dav řítící se hranou. Předpokládáme, že všechny hrany jsou stejně dlouhé. Graf prozkoumáváme „po vlnách“. V první vlně se všichni trpaslíci dostanou do vrcholů, dokterých vede z výchozího vrcholu hrana. V druhé vlně se dostanou do vrcholů, které jsou ve vzdálenosti 2 od výchozího vrcholu. Podobně v k-té vlně se všichni trpaslíci dostanou do vrcholů ve vzdálenosti k od výchozího vrcholu. Kvůli těmto vlnám se někdy průchodu do šířky říka algoritmus vlny. [7]

implementace

V počítači vlny nasimulujeme tak, že při vstupu do nového vrcholu uložíme všechny s ním sousedící vrcholy do fronty. Frontu průběžně zpracováváme.

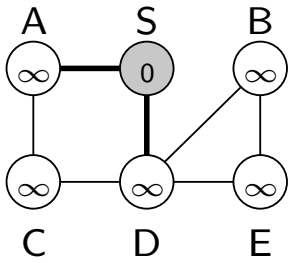
[7] Jakub Černý: Základní grafové algoritmy `http://kam.mff.cuni.cz/~kuba/ka`

Průzkum do šířky - strategie

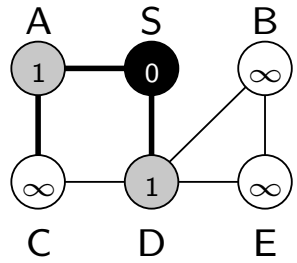
cílem je prozkoumat všechny vrcholy dosažitelné z daného iniciálního vrcholu s

- postupujeme od iniciálního vrcholu s po *vrstvách*
- nejdříve prozkoumáme všechny vrcholy dosažitelné z s po 1 hraně
- pak všechny vrcholy dosažitelné po 2 hranách, po 3 hranách atd.
- pro manipulaci s vrcholy používáme prioritní frontu Q
- $v \in Q$ právě když byl dosažen (objeven) ale ještě nebyl prozkoumán

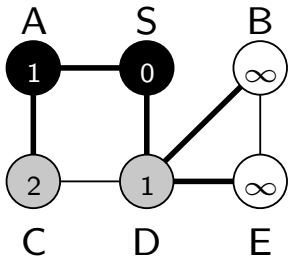
Průzkum do šířky - příklad



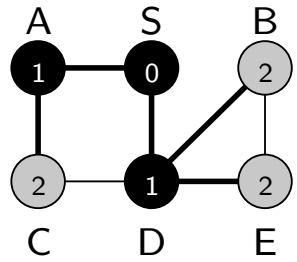
Q: S



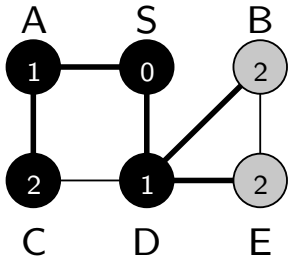
Q: AD



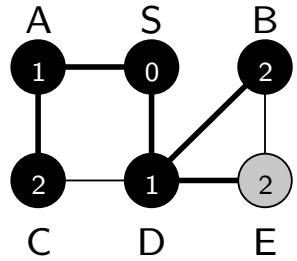
Q: DC



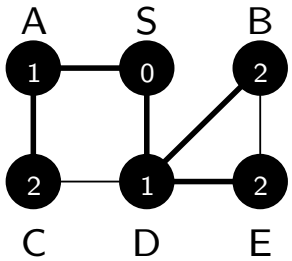
Q: CBE



Q: BE



Q: E



Q: ∅

Průzkum do šířky - atributy vrcholu

$v.color$

- v průběhu výpočtu je vrchol postupně objeven (je zařazen do fronty) a prozkoumán (všechny sousedící vrcholy jsou objeveny)
- vrchol má **černou** barvu právě když je dosažitelný z iniciálního vrcholu a byl již prozkoumán
- vrchol má **šedivou** barvu právě když je dosažitelný z iniciálního vrcholu, byl již objeven, ale nebyl ještě prozkoumán
- vrchol má **bílou** barvu právě když není dosažitelný z iniciálního vrcholu anebo ještě nebyl objeven

$v.\pi$

- vrchol, ze kterého byl v objeven

$v.d$

- délka (počet hran) cesty z s do v, na které byl v objeven
  ($=$ *délka nejkratší cesty z* s *do* v)

Průzkum do šířky - implementace

BFS(G, s)

```
1  foreach u ∈ V \ {s}
2     do u.color ← white;  u.d ← ∞;  u.π ← Nil od
3  s.color ← gray;  s.d ← 0;  s.π ← Nil
4  Q ← ∅
5  Enqueue(Q, s)
6  while Q ≠ ∅ do
7           u ← Dequeue(Q)
8           foreach v ∈ Adj[u] do
9              if v.color = white
10                then v.color ← gray
11                     v.d ← u.d + 1
12                     v.π ← u
13                     Enqueue(Q, v) fi
14             u.color ← black od
15 od
```

BFS a nejkratší cesty v neohodnoceném grafu

Nejkratší cesta v neohodnoceném grafu

Délka nejkratší cesty z s do v, značíme $\delta(s, v)$, je definována jako minimální počet hran na cestě z s do v. Když neexistuje žádná cesta z s do v, tak $\delta(s, v) = \infty$.

Nejkratší cestou z s do v je každá cesta z s do v která má $\delta(s, v)$ hran.

BFS a nejkratší cesty v neohodnoceném grafu

Věta 12

Nechť $G = (V, E)$ je graf a $s \in V$ jeho vrchol, na které aplikujeme algoritmus BFS. *Pak po ukončení výpočtu pro každý vrchol $v \in V$ platí*

$$v.d = \delta(s, v)$$

dokážeme indukcí podle hodnoty $v.d$

1. vrchol s má $s.d = 0$ a nejkratší cesta z s do s má nula hran
2. předpokládejme, že pro všechny vrcholy s hodnotou $v.d \leq k$ je $v.d$ délka nejkratší cesty do v
3. potřebujeme ukázat indukční krok a to je, že každý vrchol v s $v.d = k + 1$ leží ve vzdálenosti $k + 1$ od s
   - pokud ne, tak existuje kratší cesta z s do v a nechť (w, v) je poslední hrana na tého cestě
   - $w.d < k$
   - potom se ale měl algoritmus při zpracovávání vrcholu w podívat na hranu (w, v) a nastavit hodnotu $v.d$ na $w.d + 1$
   - $w.d + 1 < k + 1$, spor

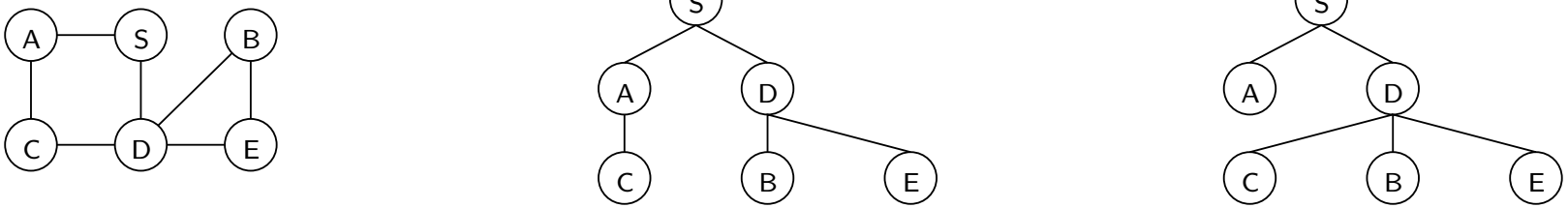
BFS strom a nejkratší cesty

- algoritmus BFS definuje přes atributy π **graf předchůdců**
- formálně: pro graf $G = (V, E)$ a iniciální vrchol s je graf předchůdců $G_\pi = (V_\pi, E_\pi)$ definovaný předpisem

$$V_\pi = \{v \in V \mid v.\pi \neq Nil\} \cup \{s\}$$
$$E_\pi = \{(v.\pi, v) \mid v \in V_\pi \setminus \{s\}\}$$

- graf předchůdců se nazývá **BFS strom**

- BFS strom je kostrou grafu
- pro každý vrchol $v \in V_\pi$ obsahuje BFS strom jedinou cestu z s do v, která je současně **nejkratší cestou z s do v**

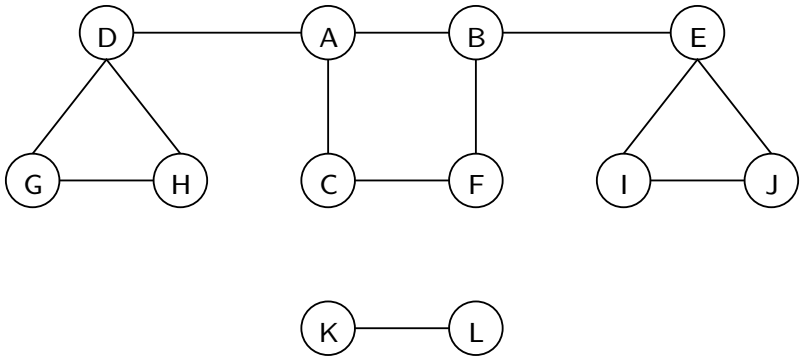


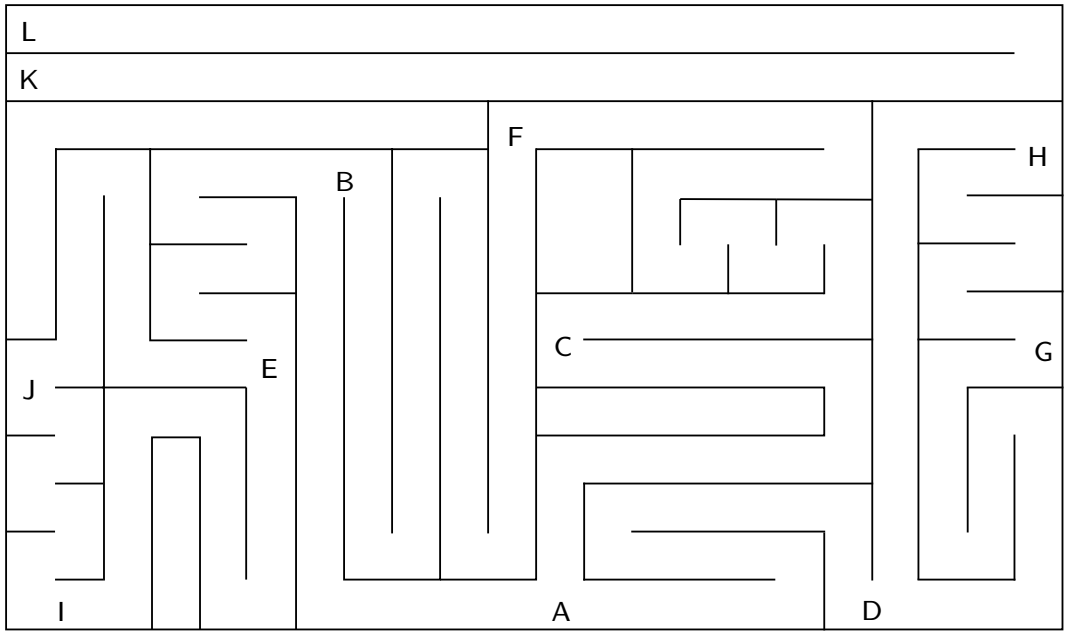
graf a jeho dva různé BFS stromy

Aplikace a algoritmy využívající BFS

- Peer to Peer Networks
- Crawlers in Search Engines
- Social Networking Websites - hledání osob *ve vzdálenosti* nejvíce k
- GPS navigační systémy
- broadcasting
- garbage collection
- Fordův Fulkersonův algoritmus pro hledání maximálního toku v síti
- testování bipartitnosti

Průzkum grafu do hloubky- motivace





pořadí, v němž BFS zkoumá vrcholy, netvoří souvislou cestu v grafu

Formulace problému

- průzkum do šířky a stejně tak i průzkum do hloubky je možné použít buď k prozkoumání té části grafu, která je dosažitelná z iniciálního vrcholu, anebo k prozkoumání celého grafu
- průzkum se dá aplikovat na orientované i neorientované grafy

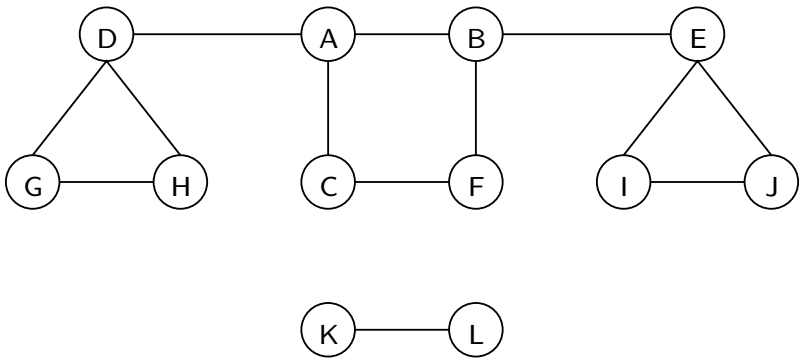
- prezentace průzkumu do hloubky předpokládá, že
  - vstupem je orientovaný graf a
  - cílem je prozkoumat celý graf

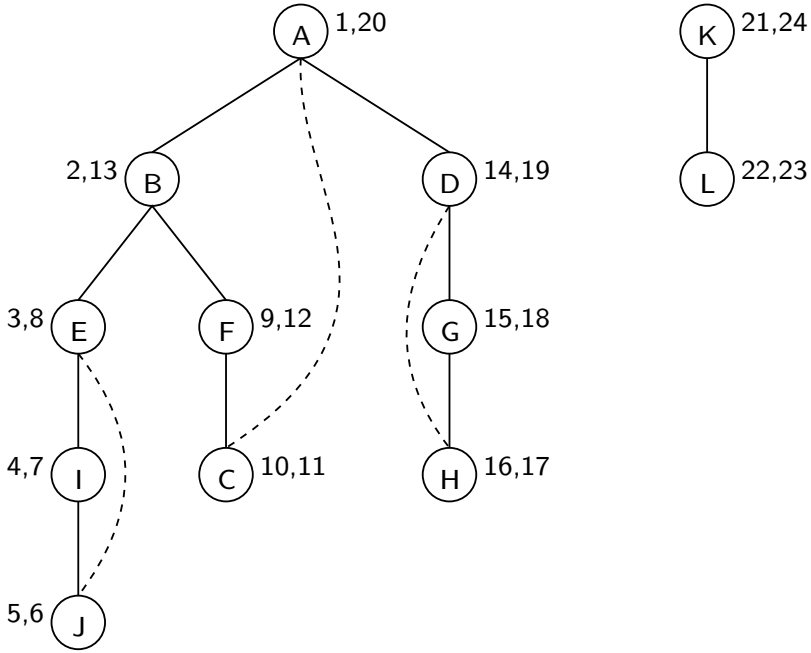
Průzkum do hloubky - strategie

- na začátku výpočtu a vždy po dokončení průzkumu vybereme jeden z dosud neprozkoumaných vrcholů a zvolíme ho za nový iniciální vrchol
- označ iniciální vrchol jako objevený
- vyber neprozkoumanou hranu (u, v), která vychází z naposledy objeveného vrcholu u, a když její koncový vrchol v ještě nebyl prozkoumán, tak ho označ jako objevený
- když všechny hrany vycházející z naposledy objeveného vrcholu u byly prozkoumány, tak ukonči průzkum vrcholu u a pokračuj vrcholem, ze kterého byl vrchol u objeven
- průzkum končí když jsou prozkoumány všechny vrcholy dosažitelné z iniciálního vrcholu

- pro manipulaci s vrcholy používáme zásobník

Průzkum grafu do hloubky - příklad





Průzkum do hloubky - atributy vrcholu

$v.color$

- vrchol má **černou** barvu právě když je dosažitelný z iniciálního vrcholu a byl již prozkoumán, tj. byly prozkoumány všechny hrany vycházející z vrcholu
- vrchol má **šedivou** barvu právě když je dosažitelný z iniciálního vrcholu, byl již objeven, ale nebyl ještě prozkoumán
- vrchol má **bílou** barvu právě když není dosažitelný z iniciálního vrcholu anebo ještě nebyl objeven

$v.\pi$

- vrchol, ze kterého byl v objeven

$v.d$

- časová značka, která zaznamenává čas první návštěvy vrcholu *(discovery time)*

$v.f$

- časová značka, která zaznamenává čas ukončení průzkumu vrcholu *(finishing time)*

Průzkum do hloubky - implementace

DFS(G)

```
1 foreach u ∈ V do u.color ← white;   u.π ← Nil od
2 time ← 0
3 foreach u ∈ V do
4     if u.color = white then DFS_Visit(G, u) fi od
```

DFS_Visit(G, u)

```
1 time ← time + 1
2 u.d ← time
3 u.color ← gray
4 foreach v ∈ Adj[u] do
5     if v.color = white then v.π ← u
6                             DFS_Visit(G, v) fi od
7 u.color ← black
8 time ← time + 1
9 u.f ← time
```

Průzkum do hloubky - iterativní implementace

DFS_Iterative_Visit(G, u)

```
1  S ← ∅
2  S.push(u)
3  time ← time + 1; u.d ← time
4  u.color ← gray
5  while S ≠ ∅ do
6          u ← S.pop()
7          if  existuje hrana (u, v) taková, že v.color = white
8             then S.push(u)
9                  S.push(v)
10                 v.color ← gray
11                 v.π ← u
12                 time ← time + 1; v.d ← time
13            else u.color ← black
14                 time ← time + 1; u.f ← time fi od
```

DFS strom

- analogicky jako u BFS definují atributy $.\pi$ graf předchůdců
- protože prohledáváme celý graf, který nemusí být nutně souvislý, graf předchůdců je DFS les, který se skládá z DFS stromů
- $G_\pi = (V, E_\pi)$

  $E_\pi = \{(v.\pi, v) \mid v \in V \text{ a } v.\pi \neq Nil\}$

DFS - vlastnosti časových značek

časové značky, které DFS přiřadí vrcholům grafu, obsahují informace o struktuře grafu a DFS stromů

- pro každý vrchol u platí $u.d < u.f$
- s každým vrcholem u je asociovaný interval $[u.d, u.f]$

časové značky určují uspořádání vrcholů

preoder uspořádání podle značky $.d$ (discovery time) v rostoucím pořadí

postorder uspořádání podle značky $.f$ (finishing time) v rostoucím pořadí

reverse postorder uspořádání podle značky $.f$ (finishing time) v klesajícím pořadí

DFS - klasifikace hran

stromová hrana (*tree edge*) je hrana (u, v) obsažená v DFS lese
při průzkumu hrany je vrchol v bílý
$u.d < v.d < v.f < u.f$

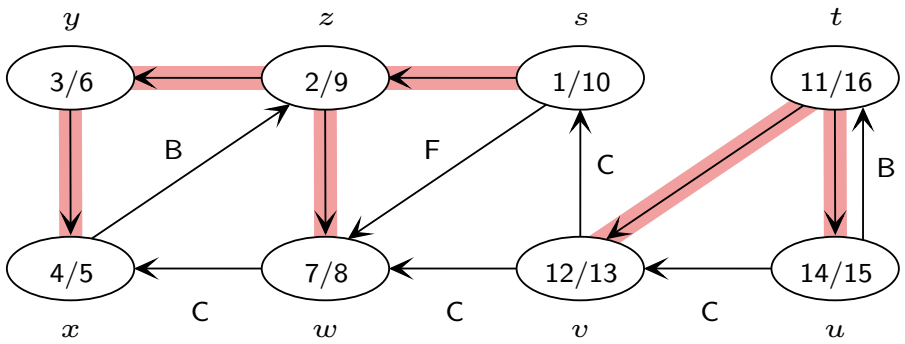
zpětná hrana (*back edge*) je hrana (u, v) která spojuje vrchol u s jeho předchůdcem v v DFS stromu
při průzkumu hrany je vrchol v šedivý
$v.d < u.d < u.f < v.f$

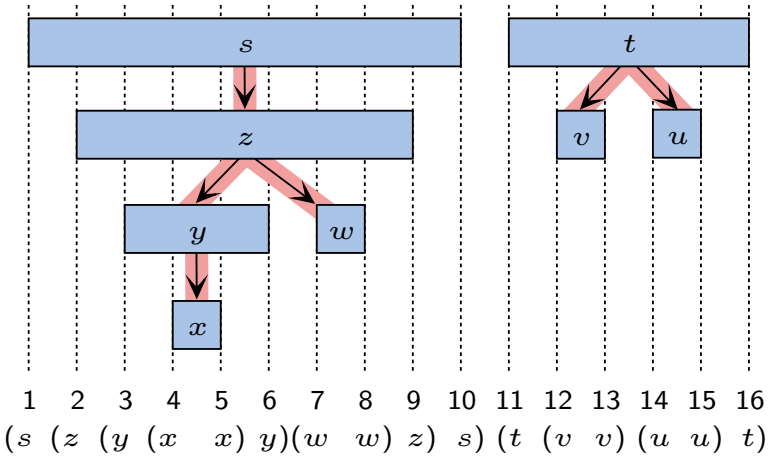
dopředná hrana (*forward edge*) je hrana (u, v), která nepatří do DFS stromu a která spojuje vrchol u s jeho následníkem v DFS stromu
při průzkumu hrany je vrchol v černý
$u.d < v.d < v.f < u.f$

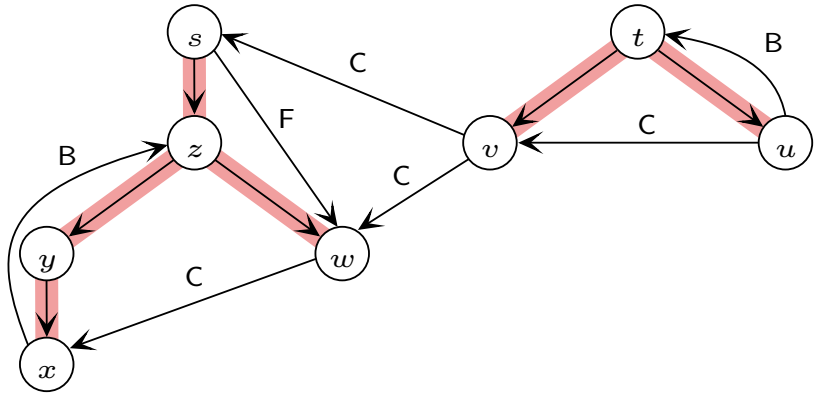
příčná hrana (*cross edge*) všechny ostatní hrany
při průzkumu hrany je vrchol v černý
$v.d < v.f < u.d < u.f$

všechny hrany v neorientovaném grafu jsou buď stromové anebo zpětné

Časové značky a klasifikace hran - příklad







stromové hrany jsou zvýrazněné, zpětné hrany jsou označeny písmenem B, dopředné písmenem F a příčné písmenem C

Aplikace a algoritmy využívající DFS

- topologické uspořádání
- komponenty souvislosti
- artikulace a mosty
- testování planarity
- hledání cesty v bludišti
- generování bludiště