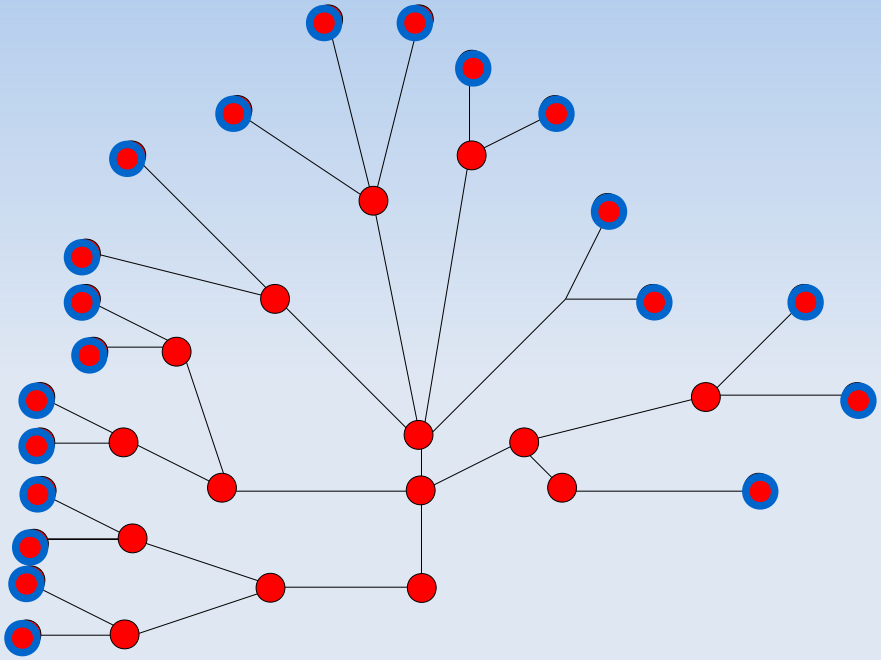
O čom bude prednáška?

Stromy

Definícia 1:
Nech $G = (V, E)$ je neorientovaný graf bez slučiek. Potom sa graf G nazýva **strom**, ak G je spojitý a neobsahuje cyklus.

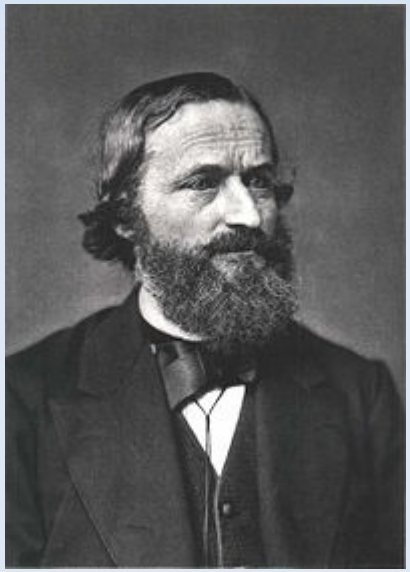


Vrcholy stupňa 1 nazývame **listy**.

Kto za to môže?

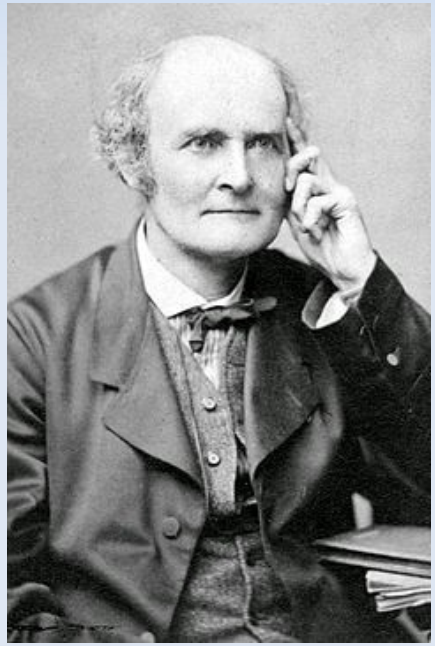
Gustav Kirchfof

(1847)

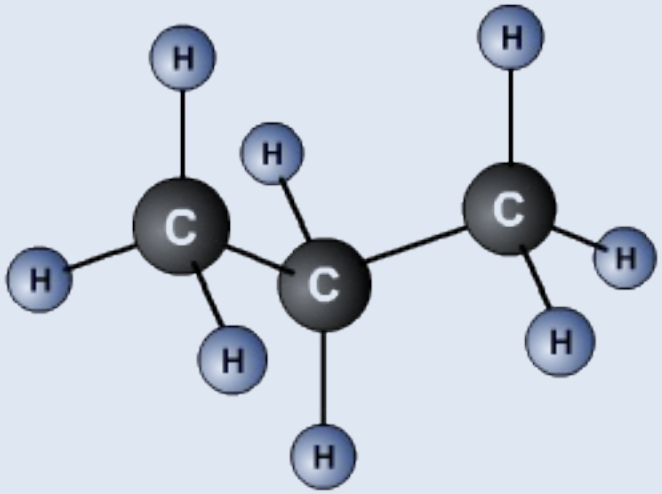


Arthur Cayley

(1857)



$CH_3\text{-}CH_2\text{-}CH_3$



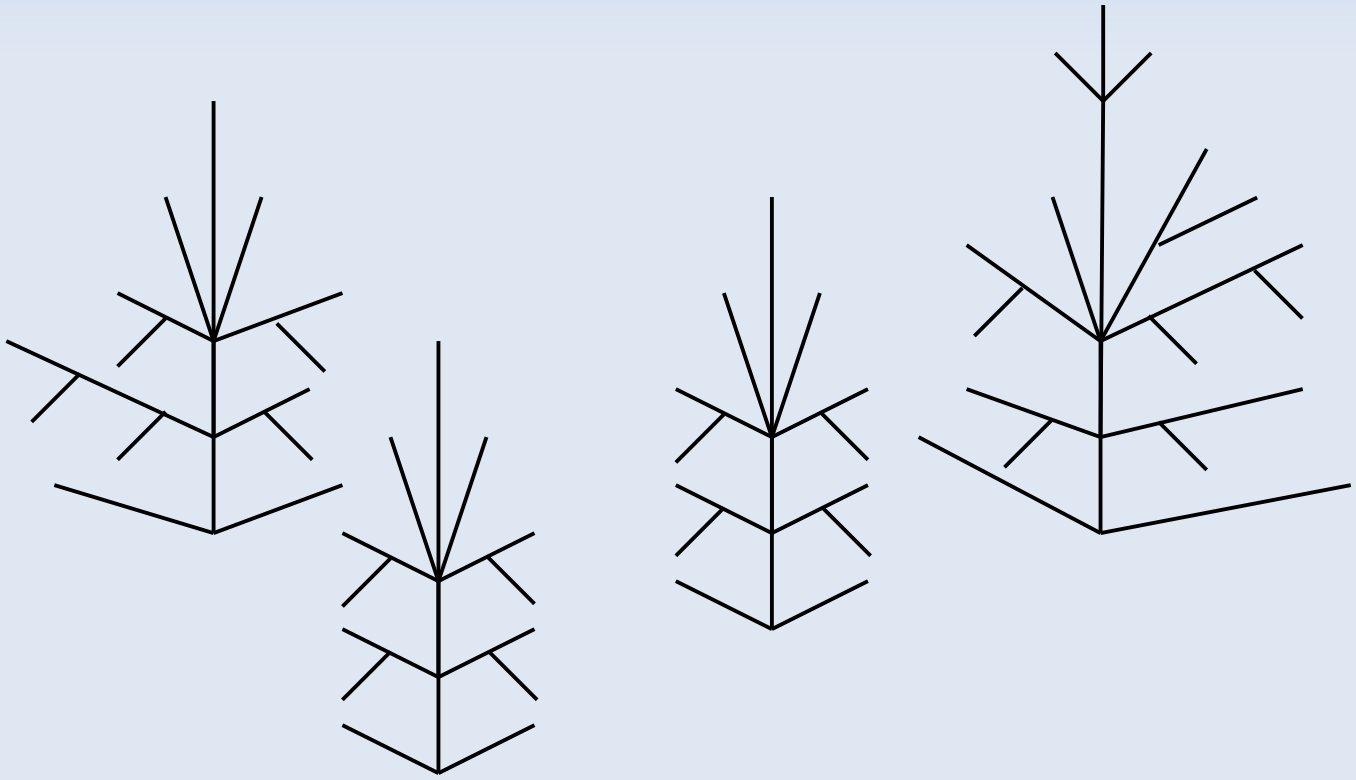
Propane

Základné pojmy, vety a dôsledky

Graf pozostávajúci z viacerých stromov sa nazýva **les**

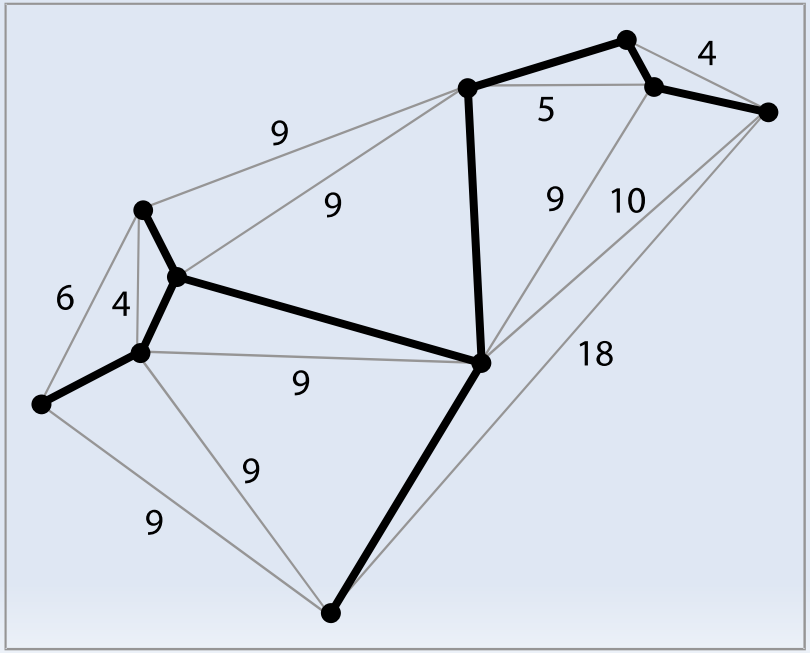
Základné pojmy, vety a dôsledky

Graf pozostávajúci z viacerých stromov sa nazýva **les**



Základné pojmy, vety a dôsledky

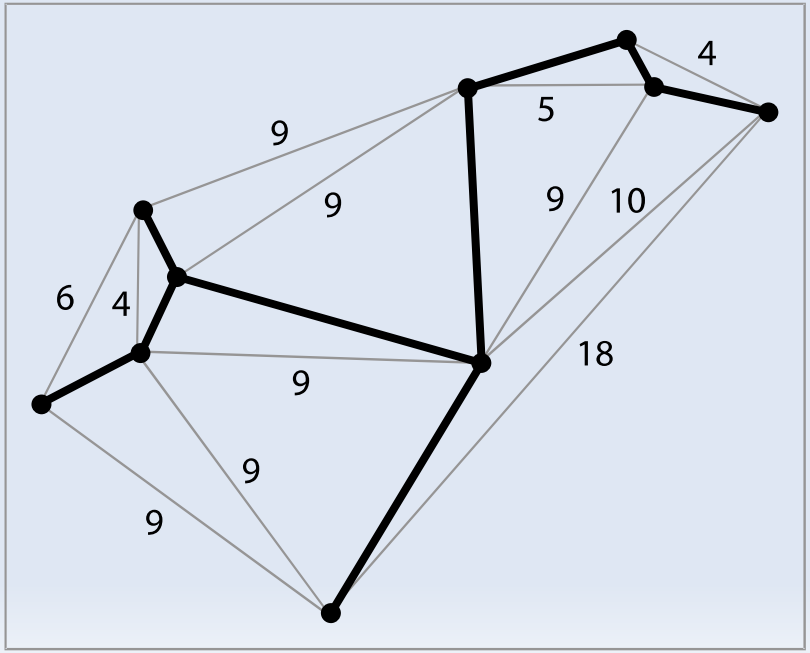
Pokrývajú ci strom grafu *G* je taký podgraf ***T***, ktorý je pokrývajúcim podgrafom grafu *G*, a je stromom



Takýto graf nazývame aj **kostrou** grafu

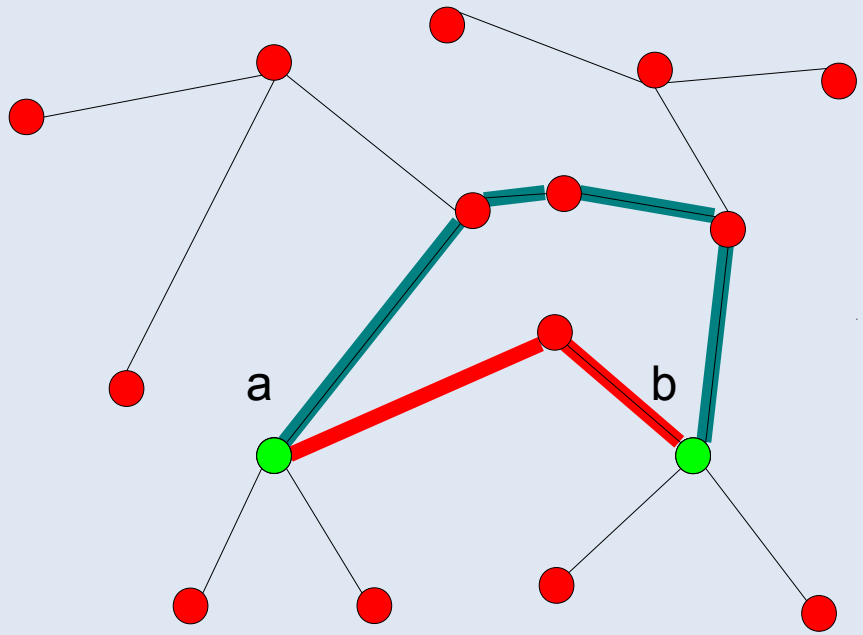
Základné pojmy, vety a dôsledky

Veta 1: Neorientovaný graf $G = (V, E)$ je spojitý vtedy a len vtedy keď má pokrývajúci strom.



Základné pojmy, vety a dôsledky

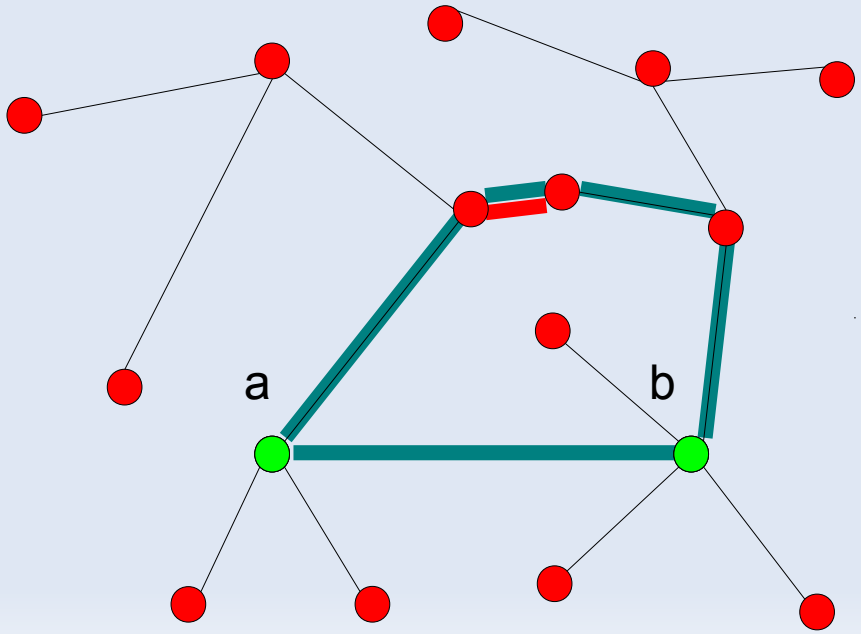
Veta 2: Ak a a b sú dva rôzne vrcholy stromu $T = (V, E)$, tak existuje práve jedna cesta, ktorá spája vrcholy a a b.



Základné pojmy, vety a dôsledky

Veta 3:

- Strom T je ”minimálne spojitý”
  (T-e je nesúvislý pre každú hranu $e \in T$)
- Strom T je ”maximálne acyklický”
  (Pridaním ľubovoľnej hrany sa stane cyklický)



Základné pojmy, vety a dôsledky

Spomenuté tvrdenia sú ekvivalentné:

(i) T je strom

(ii) Dva vrcholy z T sú spojené práve jednou cestou

(iii) T je mimálne spojitý

(iv) T je maximálne acyklický

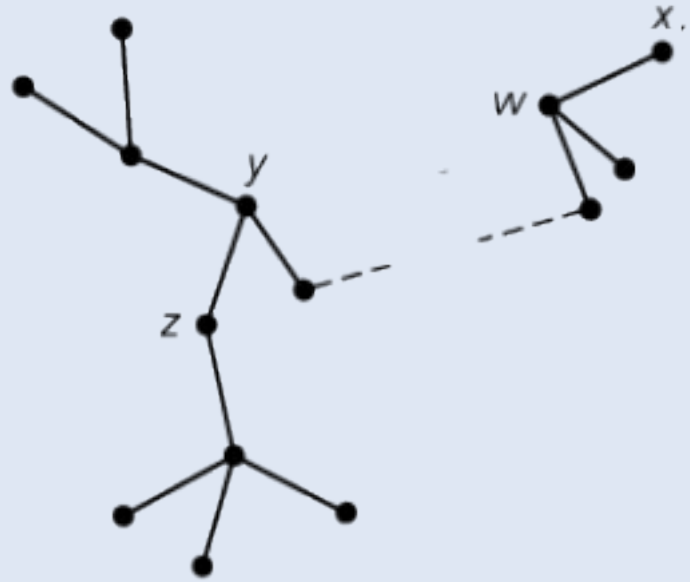
Základné pojmy, vety a dôsledky

Veta 4: V každom strome $T\,(V, E)$ platí:

$$|V| = |E| + 1$$

Dôkaz: indkuciou na počet hrán

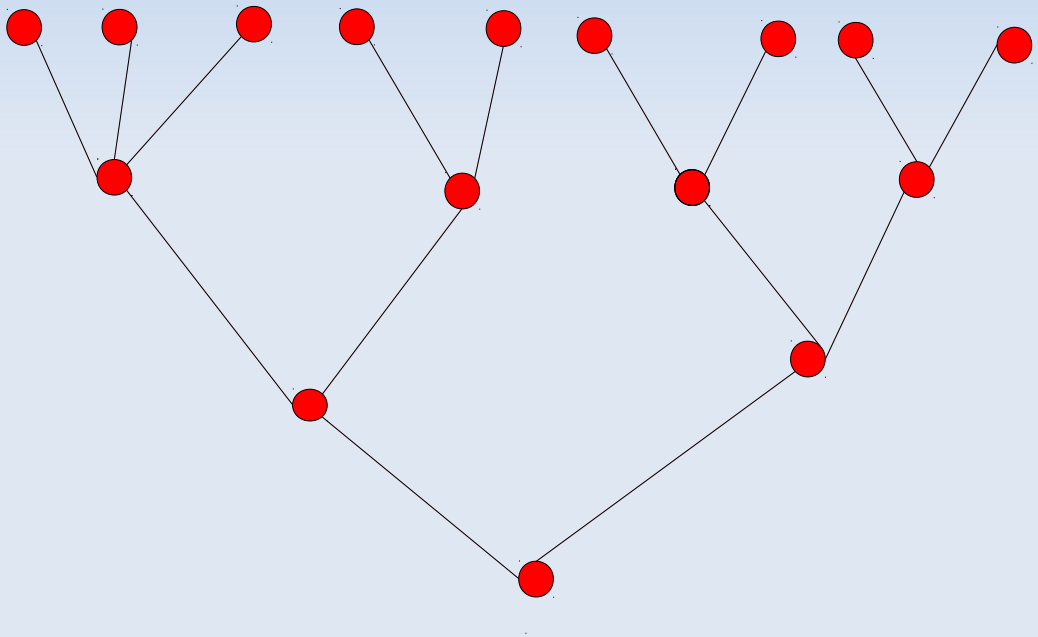
$$\begin{aligned} |V| &= |V_1| + |V_2| \\ &= (|E_1| + 1) + (|E_2| + 1) \\ &= (|E_1| + |E_2| + 1) + 1 = |E| + 1 \end{aligned}$$



Zakorenené stromy

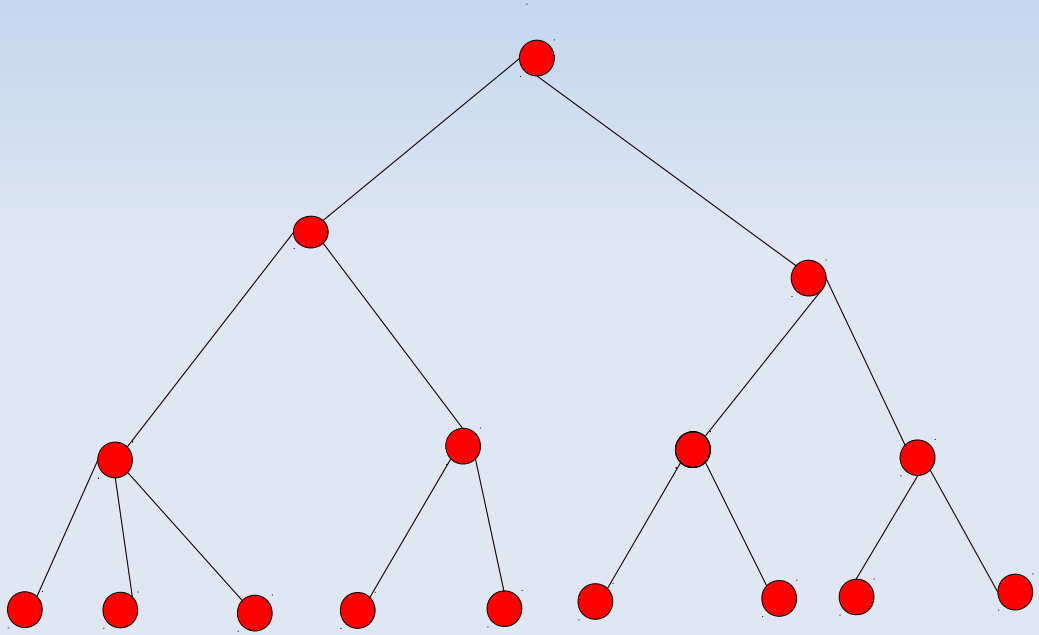
Zakorenené stromy

- Zvykneme hovoriť o hĺbke stromu
- Ako koreň stromu môžeme zvoliť ktorýkoľvek vrchol



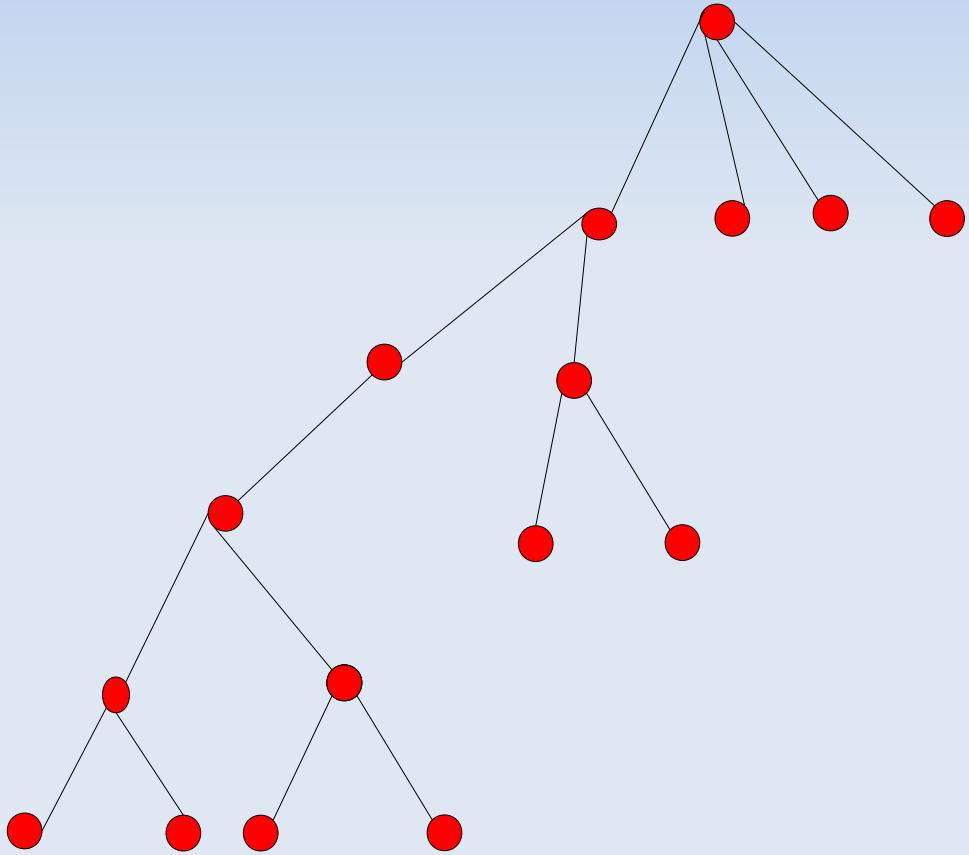
Zakorenené stromy

- Zvykneme hovoriť o hĺbke stromu
- Ako koreň stromu môžeme zvoliť ktorýkoľvek vrchol

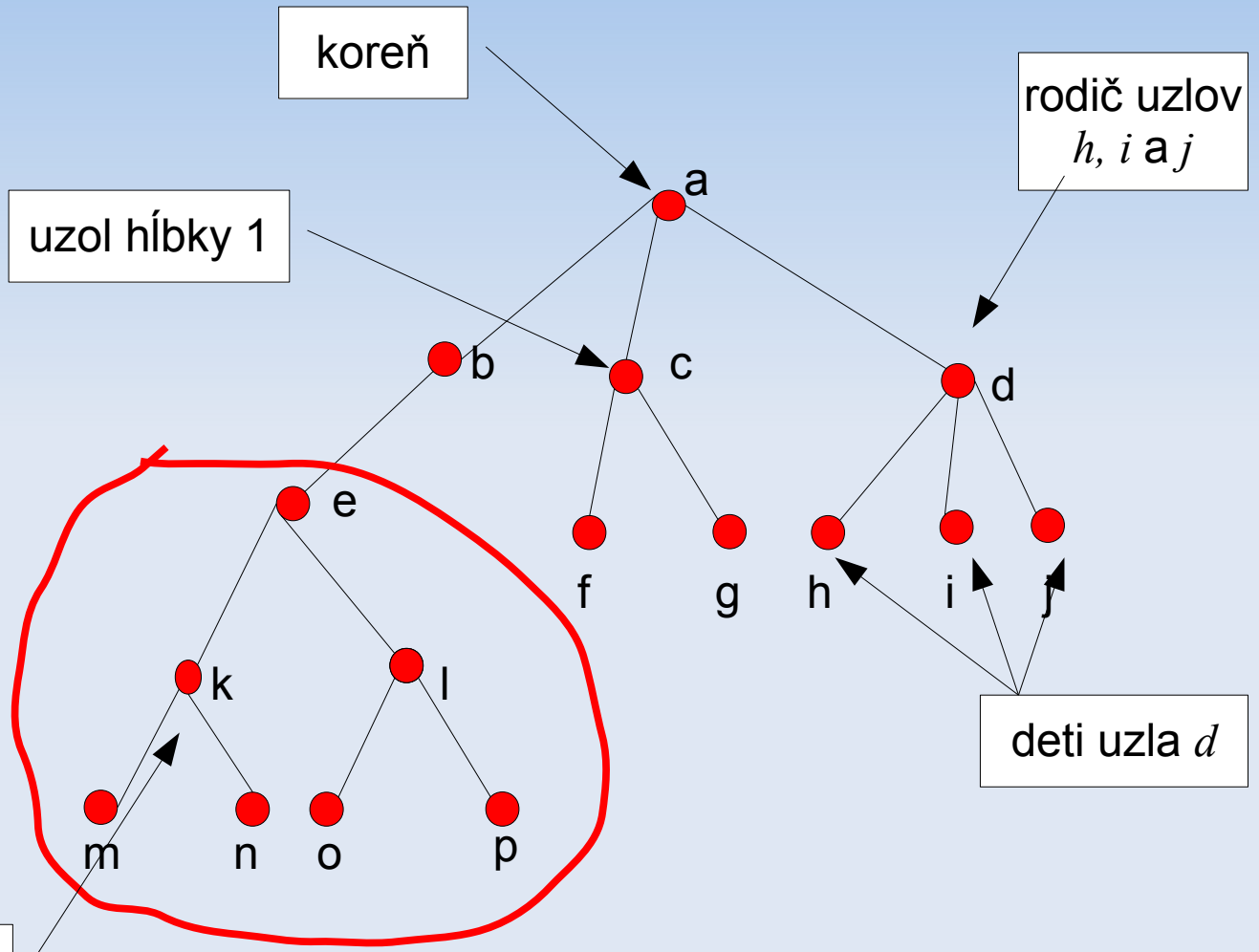


Zakorenené stromy

- Zvykneme hovoriť o hĺbke stromu
- Ako koreň stromu môžeme zvoliť ktorýkoľvek vrchol



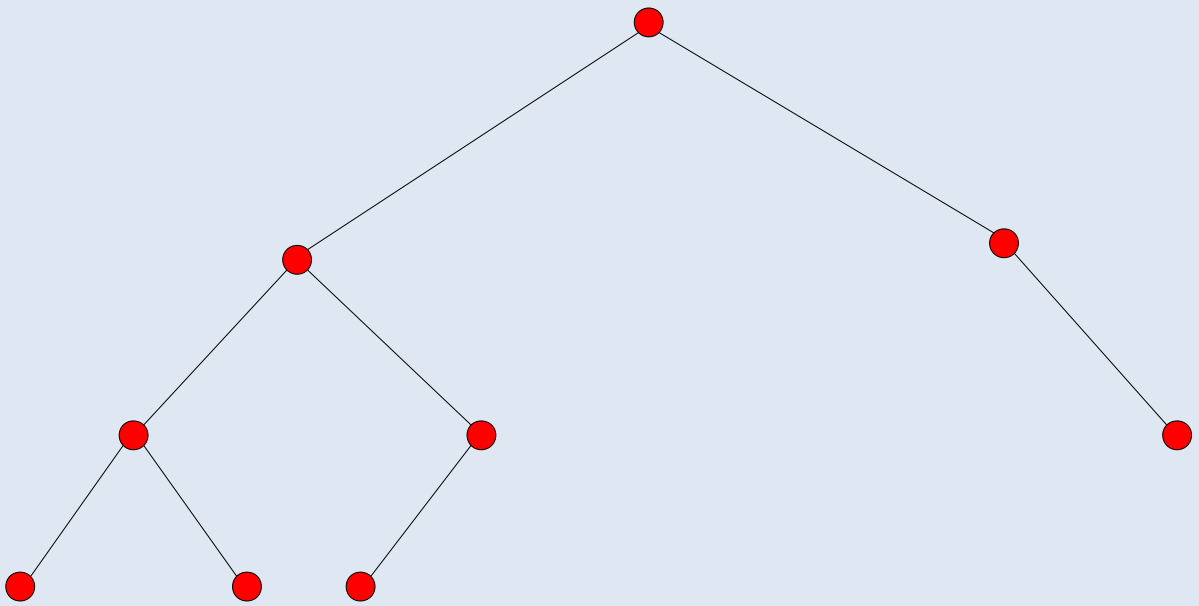
Zakorenené stromy



Binárne stromy

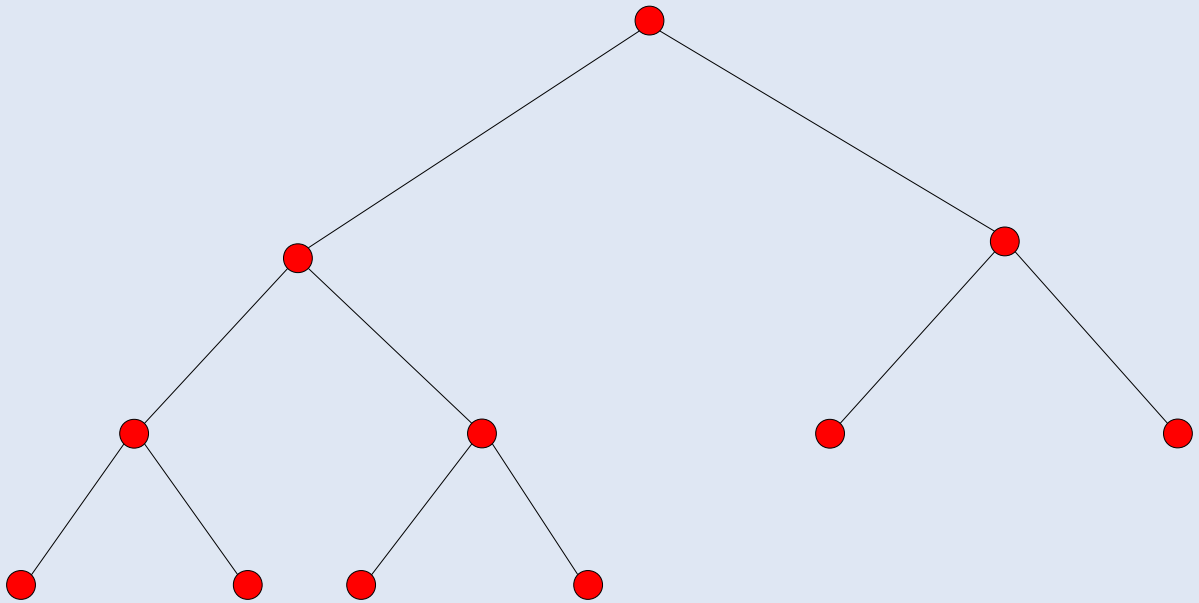
- Zakorenený strom nazývame binárnym, ak pre každý vrchol platí že má 0, 1 alebo 2 deti.

- Ak je hĺbka všetkých listov rovnaká, nazýva sa stromom vyváženým.



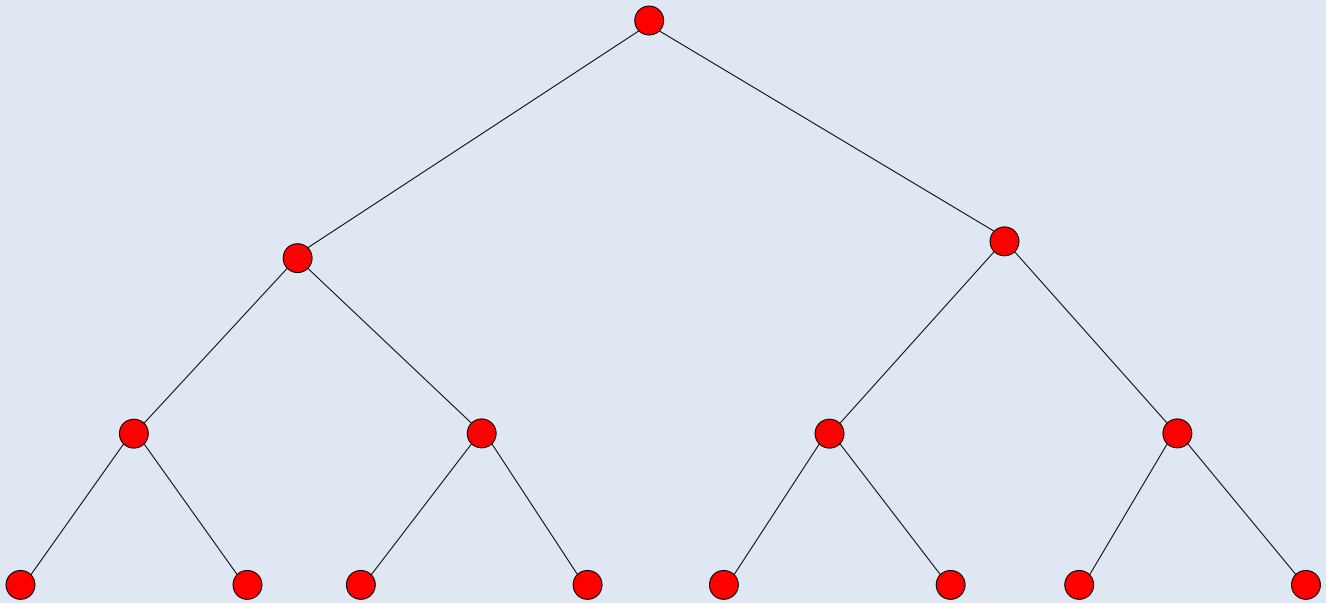
Binárne stromy

- Zakorenený strom nazývame binárnym, ak pre každý vrchol platí že má 0, 1 alebo 2 deti.
- Ak platí, že všetky vnútorné vrcholy majú dve deti a listy 0, nazývame takýto strom *kompletný*.



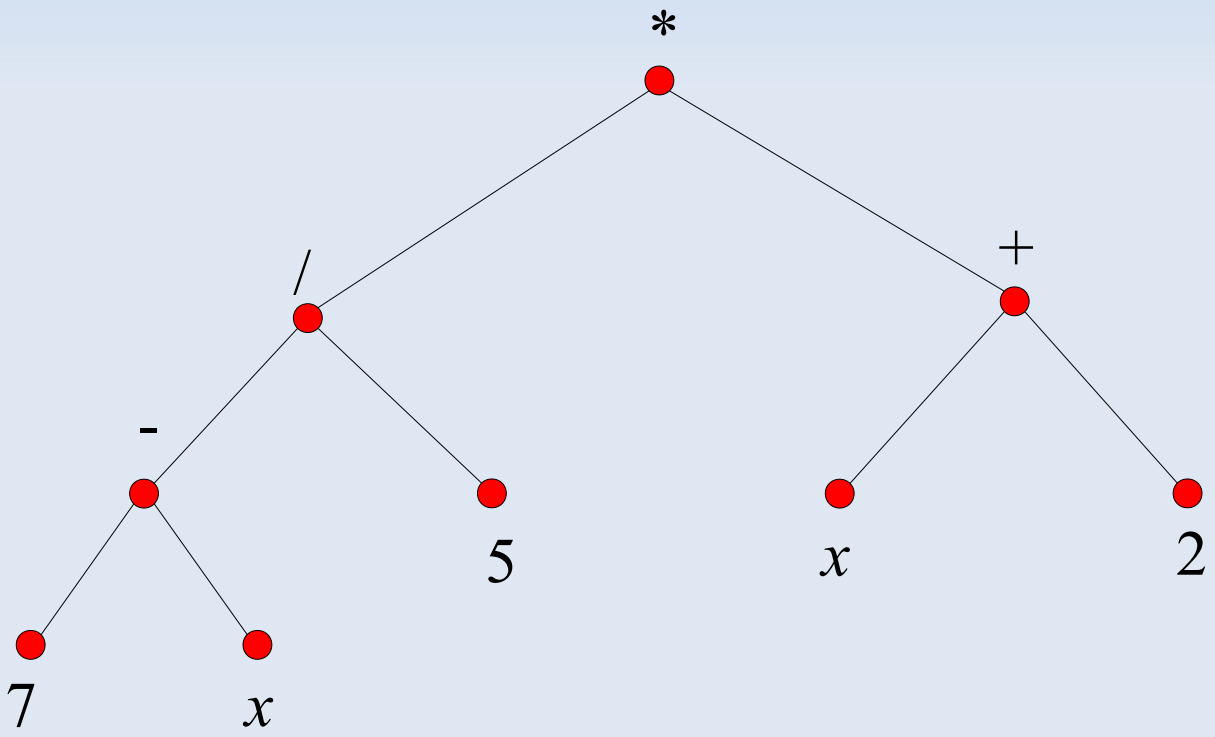
Binárne stromy

- Zakorenený strom nazývame binárnym, ak pre každý vrchol platí že má 0, 1 alebo 2 deti.
- Ak platí, že všetky vnútorné vrcholy majú dve deti a listy 0, nazývame takýto strom *kompletný*.
- Ak je hĺbka všetkých listov rovnaká, nazýva sa stromom vyváženým.



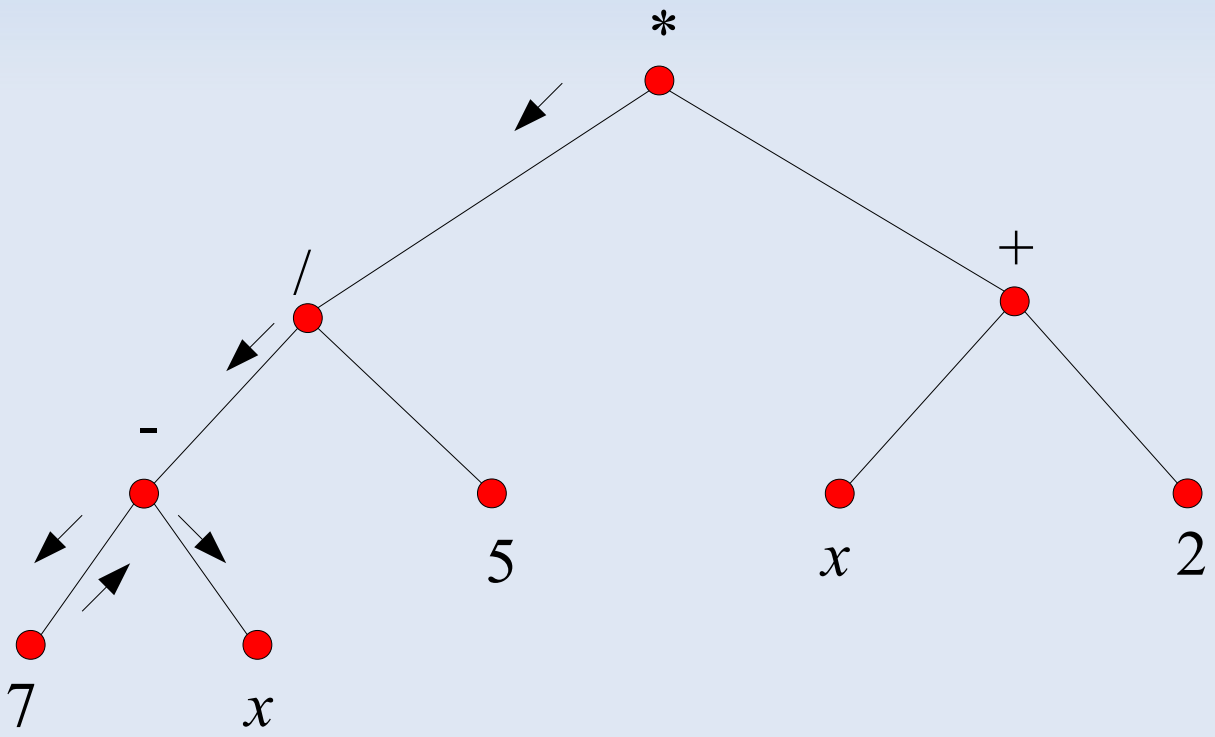
Binárne stromy

$$((7-x)/5)*(x+2)$$



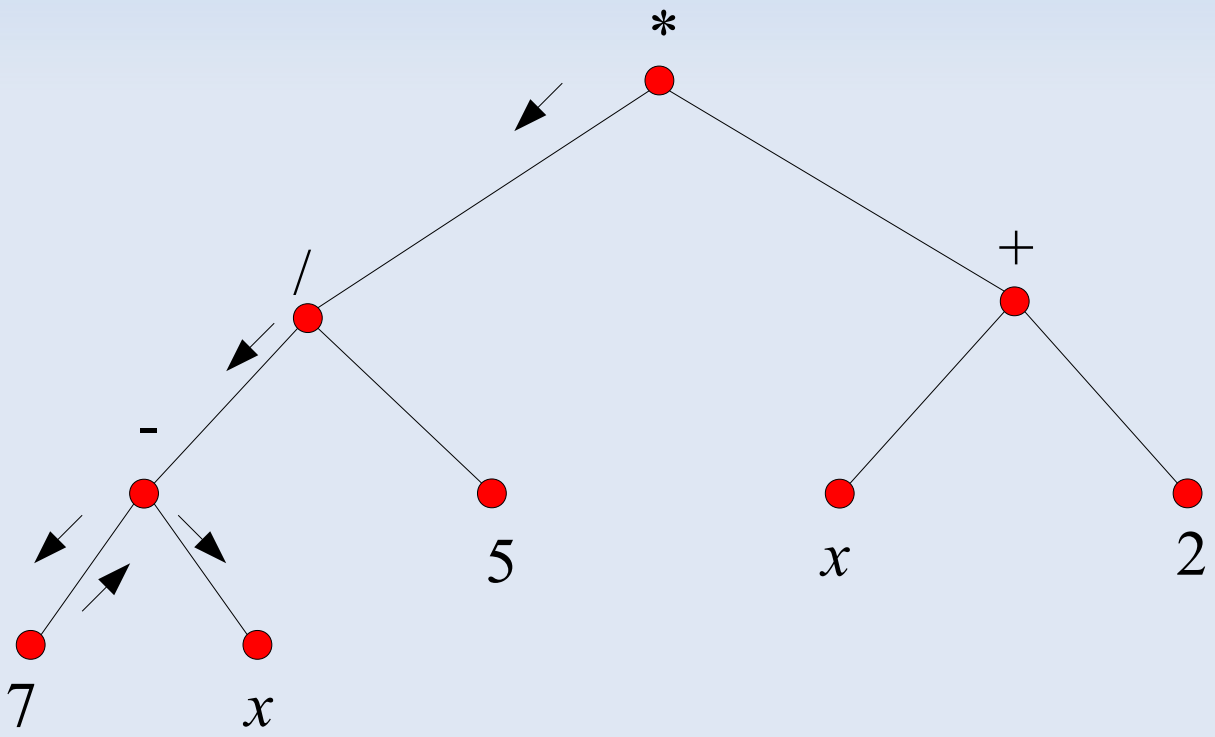
Strom ako dátová štruktúra

Infix (Inorder): $((7-x)/5)*(x+2)$



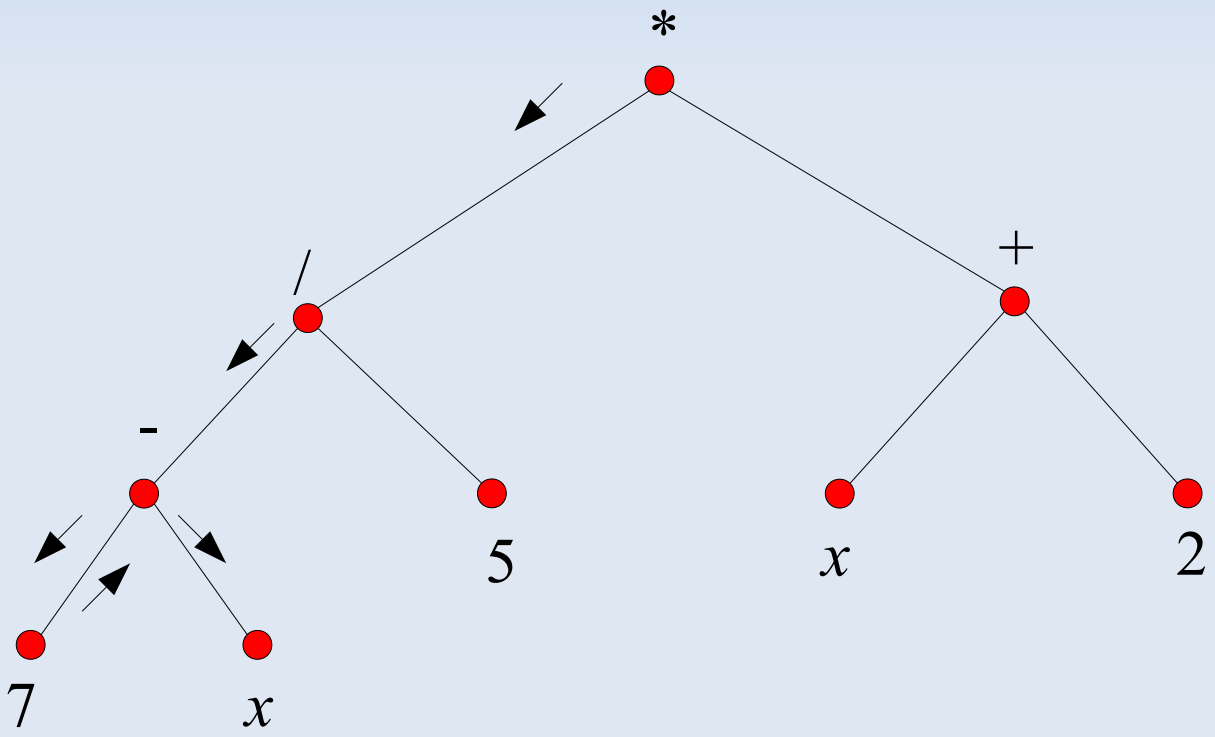
Strom ako dátová štruktúra

Prefix (Preorder): $* \quad / \quad -7 \quad x \quad 5 \quad + \quad x \quad 2$



Strom ako dátová štruktúra

Postfix (Postorder): $7 \quad x \quad - \quad 5 \quad / \quad x \quad 2 \quad + \quad *$



Aké sú veľké?

Veta 5: Nech $T = (V, E)$ je kompletný m-árny strom a $|V|=n$. Ak má T l listov, i vnútorných uzlov, tak

$$n = mi + 1$$

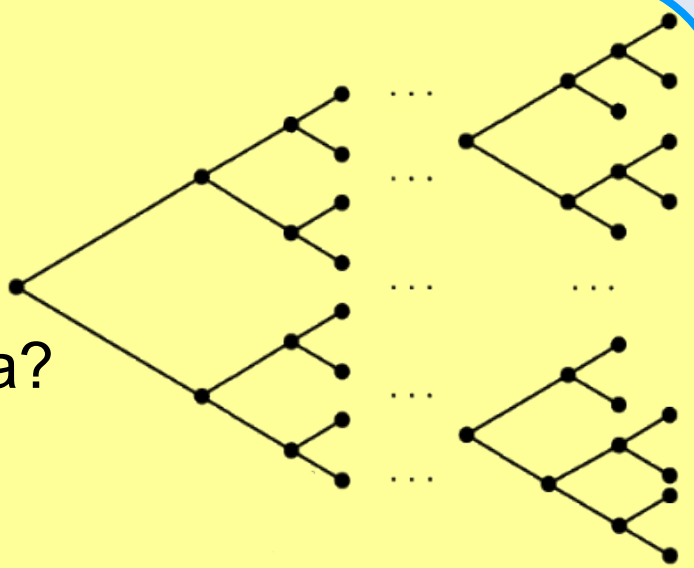
$$l = (m\text{-}1)i + 1$$

$$i = (l\text{-}1)/(m\text{-}1) = (n\text{-}1)/m$$

Príklad:

Nedávneho tenisového turnaja v Moskve sa zúčastnilo 28 hráčiek. Koľko sa muselo odohrať zápasov, aby sa dala určiť víťazka?

$$i = (l\text{-}1)/(m\text{-}1) = (28\text{-}1)/(2\text{-}1) = 27$$



Aké sú veľké?

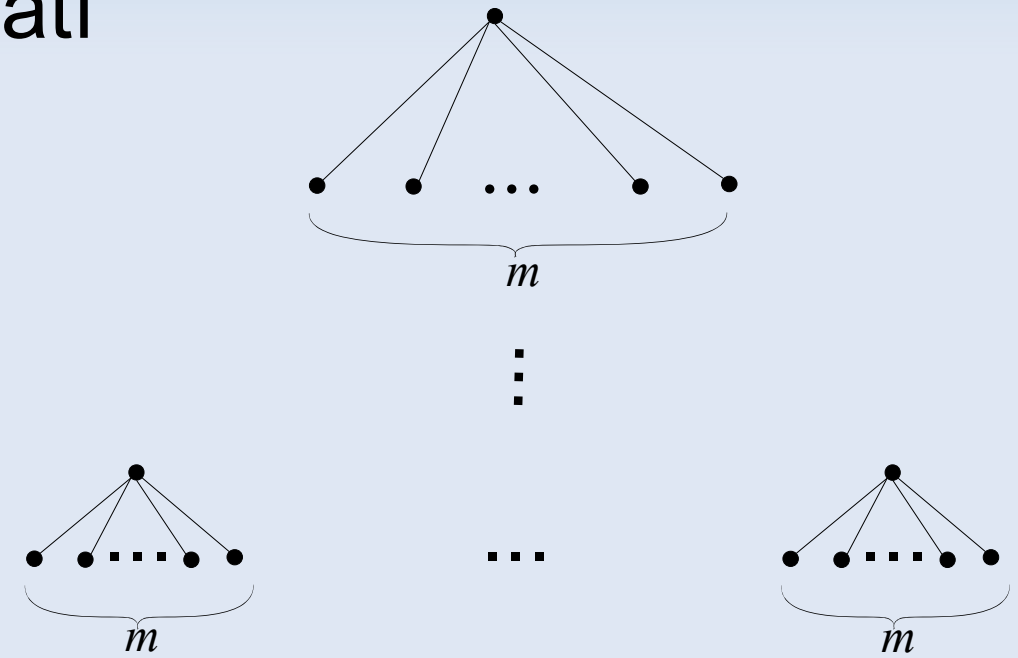
Veta 6: Nech $T = (V, E)$ je kompletný m-árny strom s výškou h a l listami. Potom $l \leq m^h$ a

Dôkaz: Indukciou, pre m podstromov stromu T, platí

$$l_i \leq m^{h-1}$$

a teda

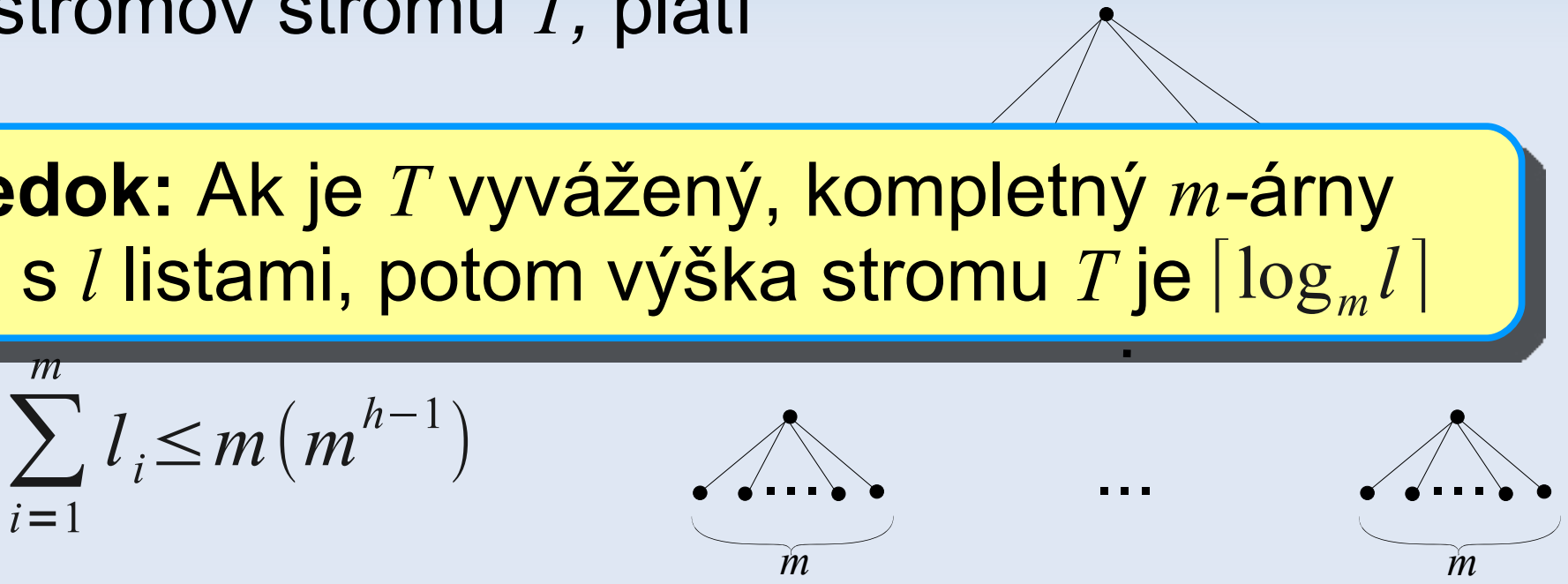
$$\sum_{i=1}^{m} l_i \leq m\left(m^{h-1}\right)$$



Aké sú veľké?

Veta 6: Nech $T = (V, E)$ je kompletný m-árny strom s výškou h a l listami. Potom $l \leq m^h$ a

Dôkaz: Indukciou, pre m podstromov stromu T, platí

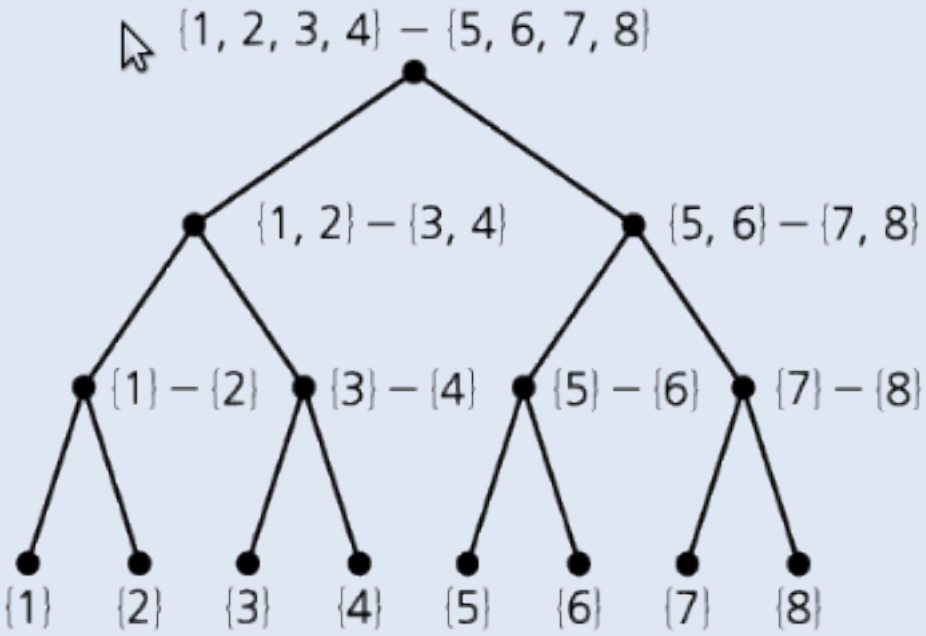


$$\sum_{i=1}^{m} l_i \leq m\left(m^{h-1}\right)$$

Dôsledok: Ak je T vyvážený, kompletný m-árny strom s l listami, potom výška stromu T je $\lceil \log_m l \rceil$

Aké sú veľké?

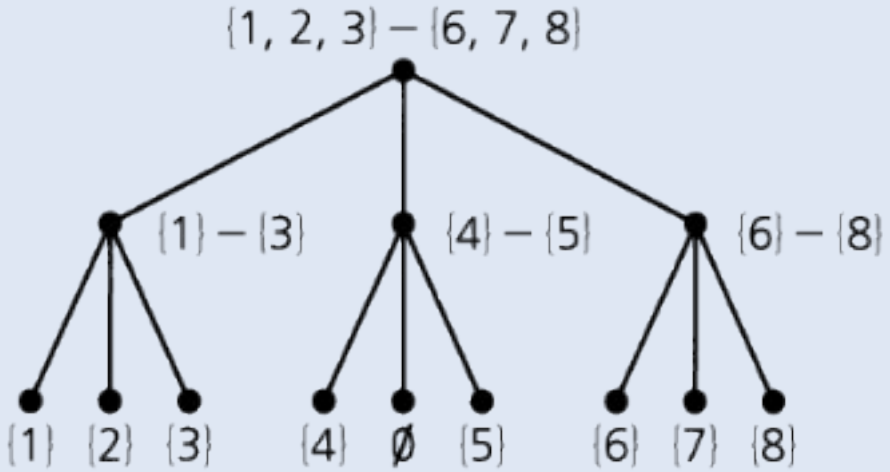
Príklad (rozhodovacie stromy):
Majme 8 mincí, z ktorých je jedna falošná a preto ťažšia a dvojramenné váhy. Ako nájdeme falošnú mincu?



Aké sú veľké?

Príklad (rozhodovacie stromy):
Majme 8 mincí, z ktorých je jedna falošná a preto ťažšia a dvojramenné váhy. Ako nájdeme falošnú mincu?

Váženie však môže mať 3 výsledky:
- Mince na váhach sú rovnako ťažké a preto pravé
- Mince na ľavej váhe sú ťažšie – obsahujú falošnú mincu
- Mince na ľavej váhe sú ťažšie – obsahujú falošnú mincu



$$h \geq \lceil \log_3 8 \rceil = 2$$

$$h \geq \lceil \log_3 n \rceil$$

O čom si povieme nabudúce?

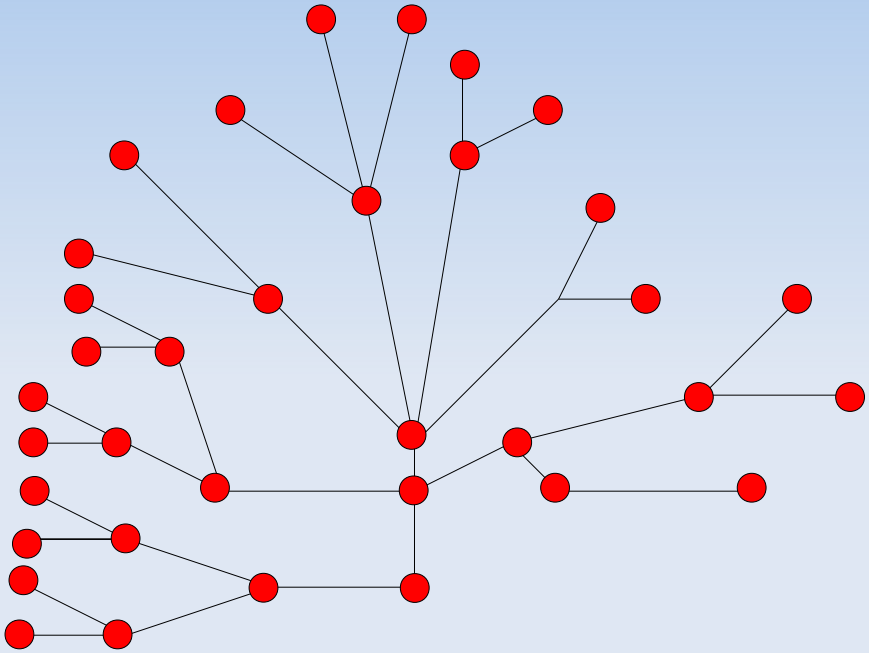
- Ako sa nestratiť v stromoch a nájsť tú správnu kostru
- Ako súvisia stromy a triedenie
- Ako sa chcel David Huffman vyhnúť skúške
- O artikulačných bodoch a dvojsúvislých grafoch

O čom si povieme dnes?

- Ako sa nestratiť v stromoch a nájsť tú správnu kostru
- Ako súvisia stromy a triedenie
- Ako sa chcel David Huffman vyhnúť skúške
- O artikulačných bodoch a dvojsúvislých grafoch

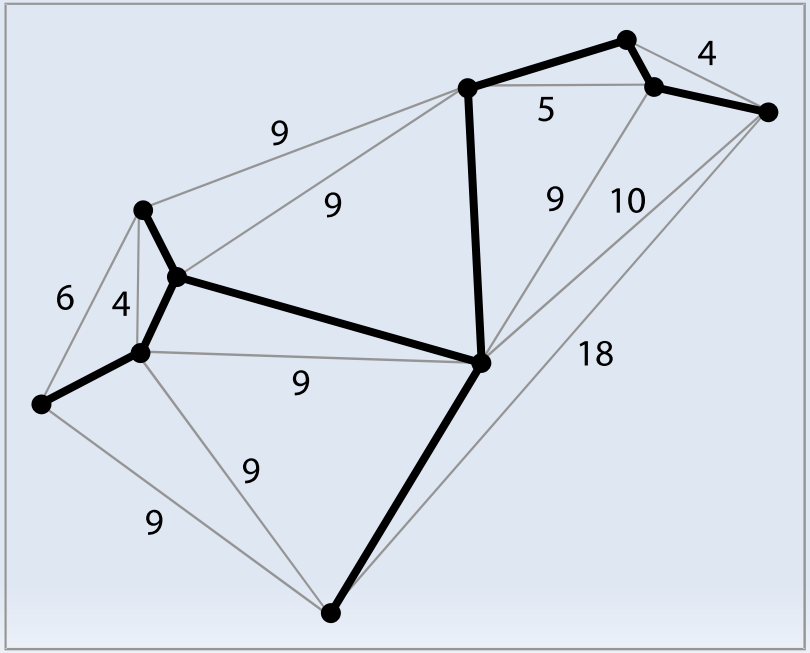
Ako nájsť kostru grafu?

Definícia 1:
Nech $G = (V, E)$ je neorientovaný graf bez slučiek. Potom sa graf G nazýva **strom**, ak G je súvislý a neobsahuje cyklus.



Základné pojmy, vety a dôsledky

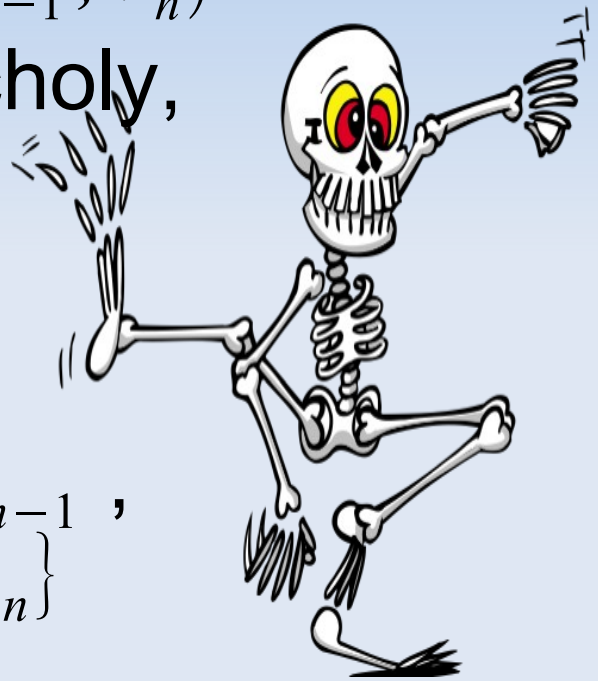
Pokrývajú ci strom grafu *G* je graf T,
ktorý je pokrývajúcim grafom grafu G,
ak je graf T strom.



Takýto graf nazývame aj **kostrou** grafu

Ako nájsť kostru grafu?

- Nájdem najdlhšiu cestu $(v_0, v_1, \ldots, v_{n-1}, v_n)$ v grafe G. Ak som pokryl všetky vrcholy, mám kostru $\longrightarrow$



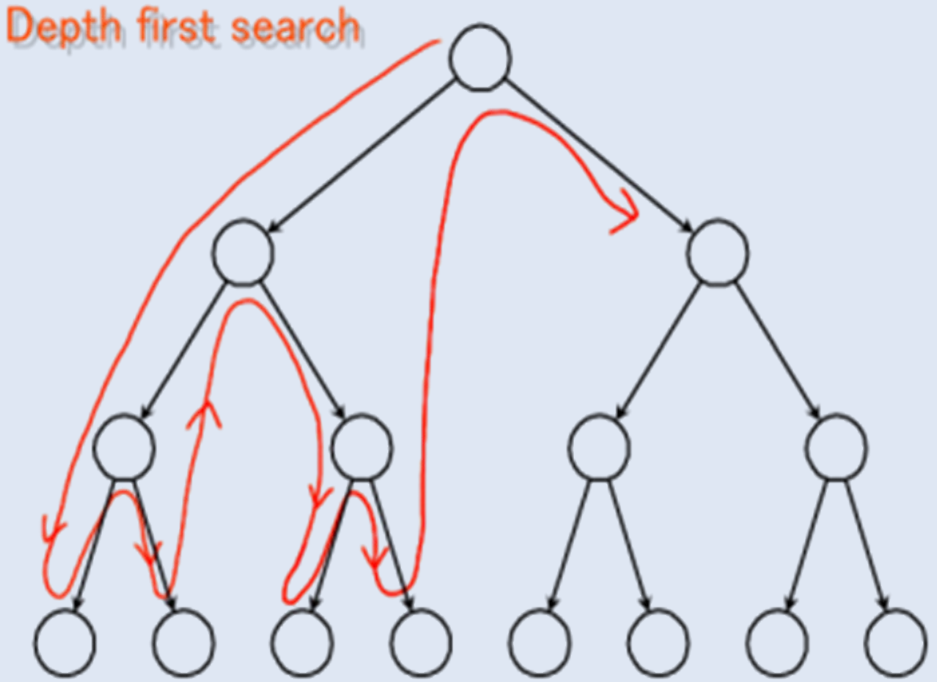
- Ak nie, vrátim sa do vrcholu v_{n-1} a nájdem najdlhšiu cestu z vrcholu v_{n-1}, ktorá neobsahuje vrcholy z $\{v_0, \ldots, v_n\}$ a pridám ich ku kostre

- *Backtracking* - postupne pokryjem celý graf G.

Depth-First Search

- *Prehľadávanie do hĺbky*
- Časová zložitosť - najhorší prípad $O(|V|+|E|)$
- Pamäťová zložitosť - $O(|V|)$

```
1  procedure DFS(G,v):
2      label v as explored
3      for all edges e in G.incidentEdges(v) do
4          if edge e is unexplored then
5              w ← G.opposite(v,e)
6              if vertex w is unexplored then
7                  label e as a discovery edge
8                  recursively call DFS(G,w)
9          else
10             label e as a back edge
```



Ako zistiť spojitosť grafu?

Majme graf daný ako zoznam jeho hrán, s priradenou váhou

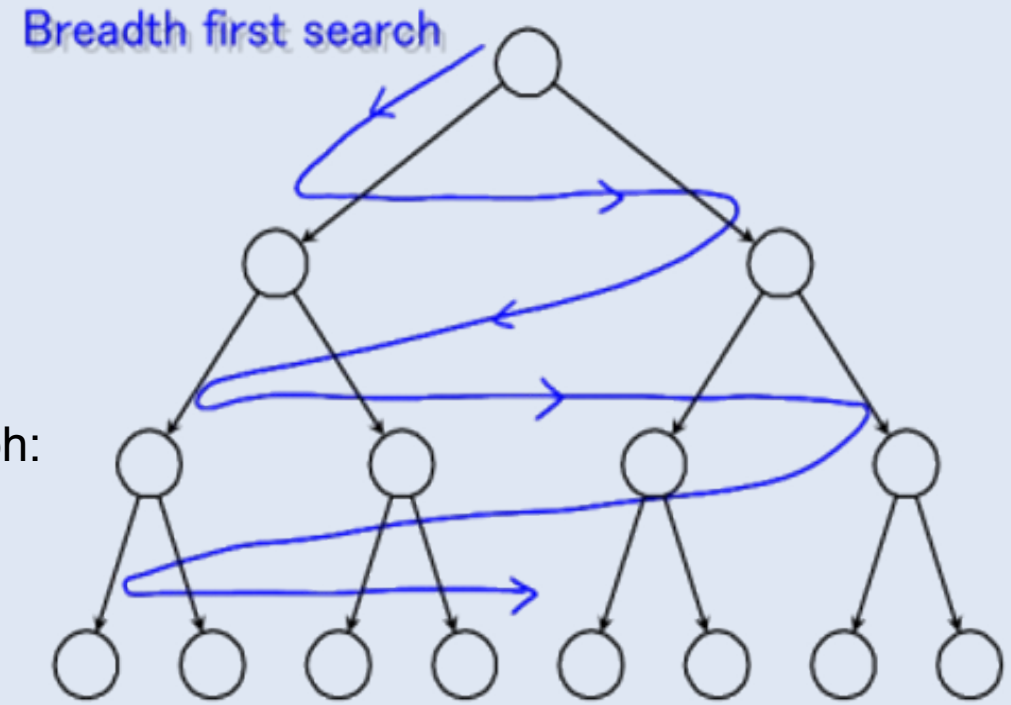
```
(a,c) - 2
(a,d) - 3
(b,c) - 1
(b,g) - 3
(c,i) - 1
(d,j) - 2
(e,f) - 1
(g,j) - 2
(f,h) - 2
(h,i) - 3
```

	a	b	c	d	e	f	g	h	i	j
a	0	0	2	3	0	0	0	0	0	0
b	0	0	1	0	0	0	3	0	0	0
c	2	1	0	0	3	0	0	0	1	0
d	3	0	0	0	0	0	0	0	0	2
e	0	0	3	0	0	1	0	0	0	0
f	0	0	0	0	1	0	0	2	0	0
g	0	3	0	0	0	0	0	0	0	2
h	0	0	0	0	0	2	0	0	3	0
i	0	0	1	0	0	0	0	3	0	0
j	0	0	0	2	0	0	2	0	0	0

Breath-First Search

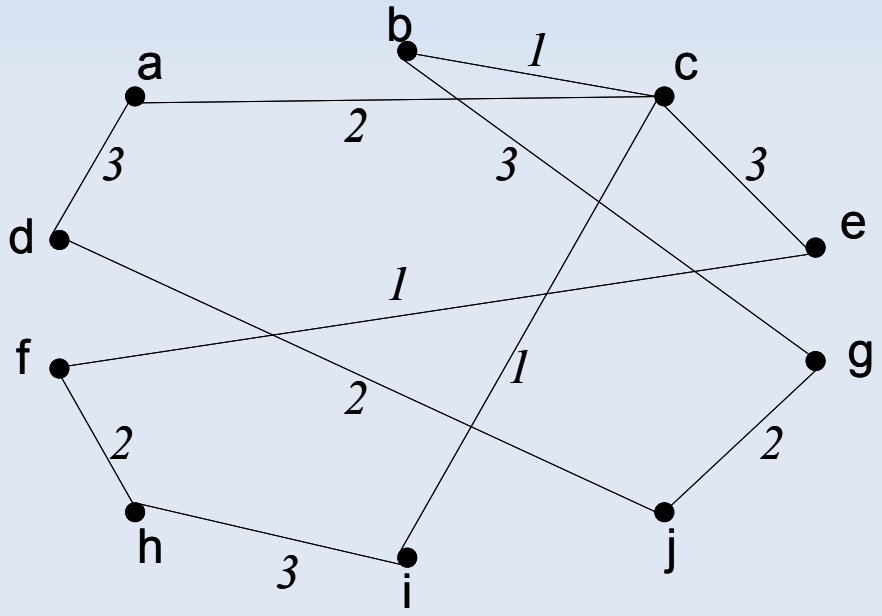
- *Prehľadávanie do šírky*
- Časová zložitosť - najhorší prípad $O(|V|+|E|)$
- Pamäťová zložitosť - $O(|V|)$

```
1   procedure BFS(Graph,source):
2       create a queue Q
3       enqueue source onto Q
4       mark source
5       while Q is not empty:
6           dequeue an item from Q into v
7           for each edge e incident on v in Graph:
8               let w be the other end of e
9               if w is not marked:
10                  mark w
11                  enqueue w onto Q
```

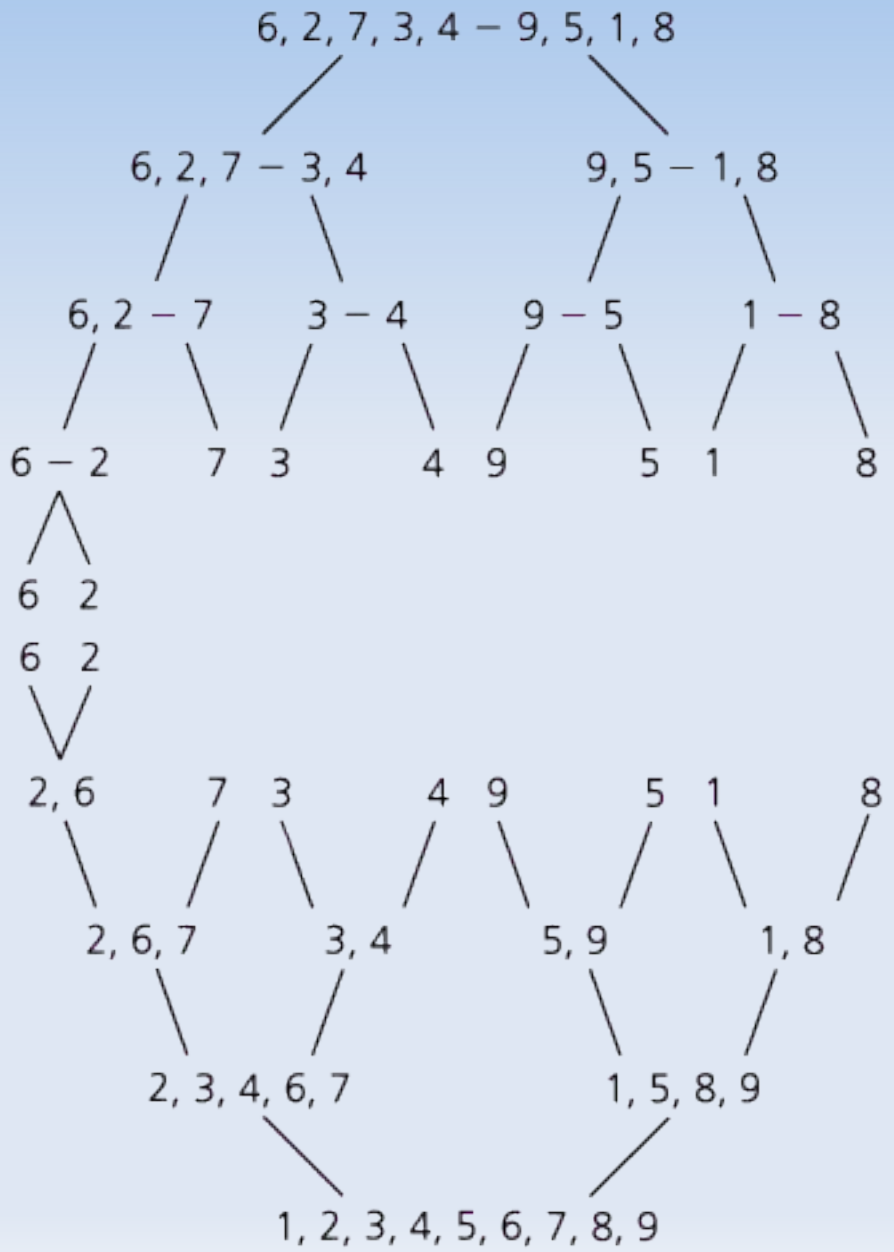


Ako zistiť spojitosť grafu?

	a	b	c	d	e	f	g	h	i	j
a	0	0	2	3	0	0	0	0	0	0
b	0	0	1	0	0	0	3	0	0	0
c	2	1	0	0	3	0	0	0	1	0
d	3	0	0	0	0	0	0	0	0	2
e	0	0	3	0	0	1	0	0	0	0
f	0	0	0	0	1	0	0	2	0	0
g	0	3	0	0	0	0	0	0	0	2
h	0	0	0	0	0	2	0	0	3	0
i	0	0	1	0	0	0	0	3	0	0
j	0	0	0	2	0	0	2	0	0	0



Usporiadanie



Usporiadanie

Merge Sort

```
function merge_sort(m)
    if length(m) ≤ 1
        return m
    var list left, right, result
    var integer middle = length(m) / 2
    for each x in m up to middle
         add x to left
    for each x in m after or equal middle
         add x to right
    left = merge_sort(left)
    right = merge_sort(right)
    result = merge(left, right)
    return result
```

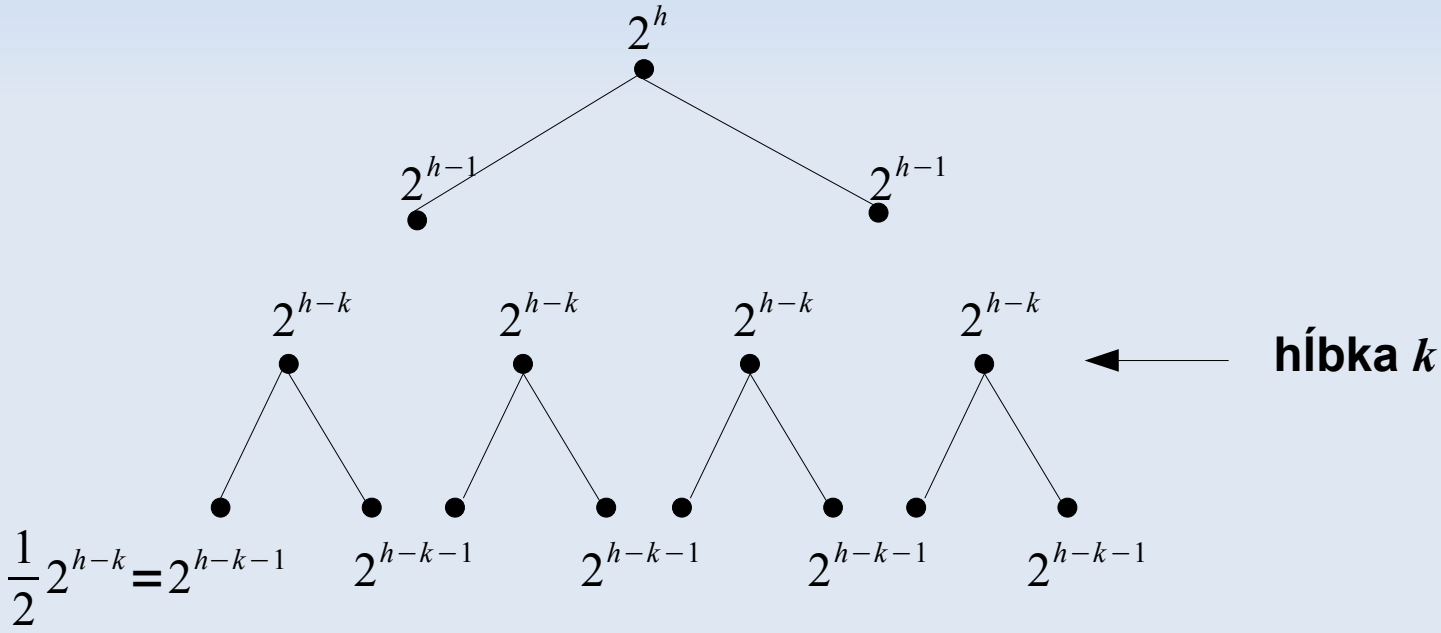
```
function merge(left,right)
    var list result
    while length(left) > 0 or length(right) > 0
        if length(left) > 0 and length(right) > 0
            if first(left) ≤ first(right)
                append first(left) to result
                left = rest(left)
            else
                append first(right) to result
                right = rest(right)
        else if length(left) > 0
            append first(left) to result
            left = rest(left)
        else if length(right) > 0
            append first(right) to result
            right = rest(right)
    end while
    return result
```

Ako rýchlo to Mergesort dokáže?

Lema 1: Nech L a K sú dva utriedené zoznamy s m a n prvkami, tak potom viem spojiť L a K do jedného utriedeného zoznamu, s použitím najviac $m+n-1$ porovnaní.

Ako rýchlo to Mergesort dokáže?

Majme zoznam 2^h prvkov. Ten musíme najprv musíme rozdeliť do stromu.



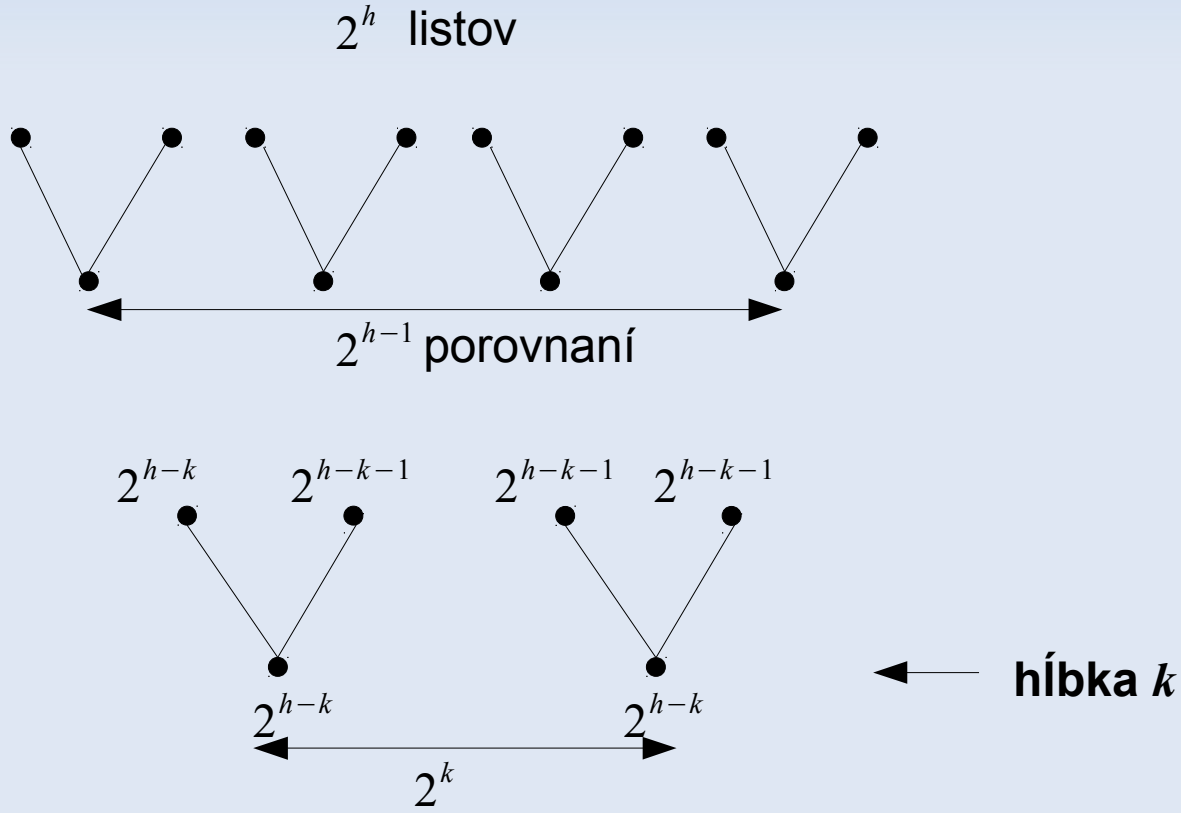
Ako rýchlo to Mergesort dokáže?

Na spojenie 2 zoznamov v hĺbke k (každý má 2^{h-k} prvkov). Z lemy 1 a dostaneme $2^{h-k}+2^{h-k}-1=2^{h-k+1}-1$ porovnaní pre zoznam v hĺbke k (bude mať 2^{h-k+1} prvkov).

Na spojenie 2^k zoznamov (2^{k-1} párov) v hĺbke k, potrebujeme

$$2^{k-1}\left(2^{h-k+1}-1\right)$$

porovnaní



Ako rýchlo to Mergesort dokáže?

$$\sum_{k=1}^{h} 2^{k-1}\left(2^{h-k+1}-1\right)=\sum_{k=0}^{h-1} 2^{k}\left(2^{h-k}-1\right)=\sum_{k=0}^{h-1} 2^{h}-\sum_{k=0}^{h-1} 2^{k}=h\,2^{h}-\left(2^{h}-1\right)$$

$$h\,2^{h}-\left(2^{h}-1\right)=n\log_2 n-\left(n-1\right)=n\log_2 n-n+1$$

- Časová zložitosť - najhorší prípad $O\left(n\log_2 n\right)$
- Pamäťová zložitosť - $O\left(n\right)$

Kódovanie

- Koľko bitov potrebujeme na zakódovanie 26 znakov abecedy?
- Ako zakódovať slovo ”mama” ?

$$2^4 < 26 < 2^5$$

m	a	m	a
00111	00001	00111	00111

Kódovanie

- V slovenčine sú najčastejšie a, o, e, i, s. Čo ak by sme pre ne použili kódy rôznych dĺžok
- Zakódujme teraz slovo ”šašo”
- Problém: tento kód kóduje napr. aj slová eio, sose, sae, ...

a: 01 o: 0 e:10 i:11 s:1

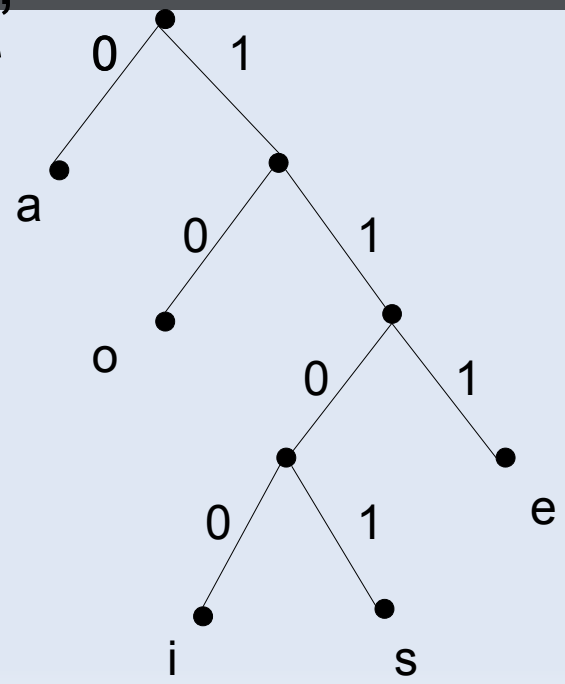
10110

Kódovanie

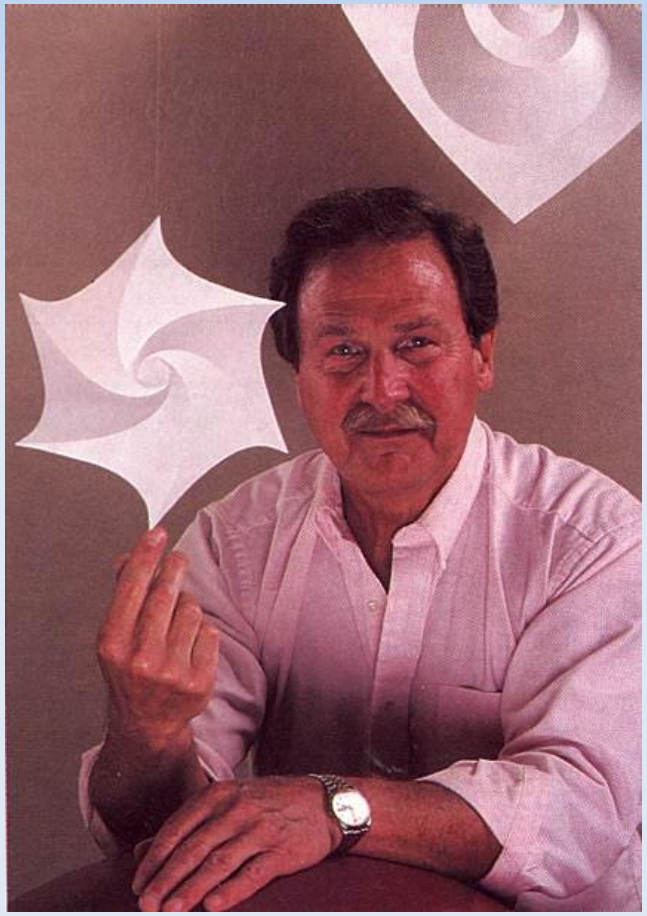
Vyskúšajme iné kódovanie.
a: 0 o: 10 e: 111 i:1100 s:1101

Prefixový kód: Kód žiadneho znaku nie je prefixom iného.

- Nenastane nejednoznačnosť
- Na dekódovanie môžeme použiť jednoduchý binárny strom.



David Huffman



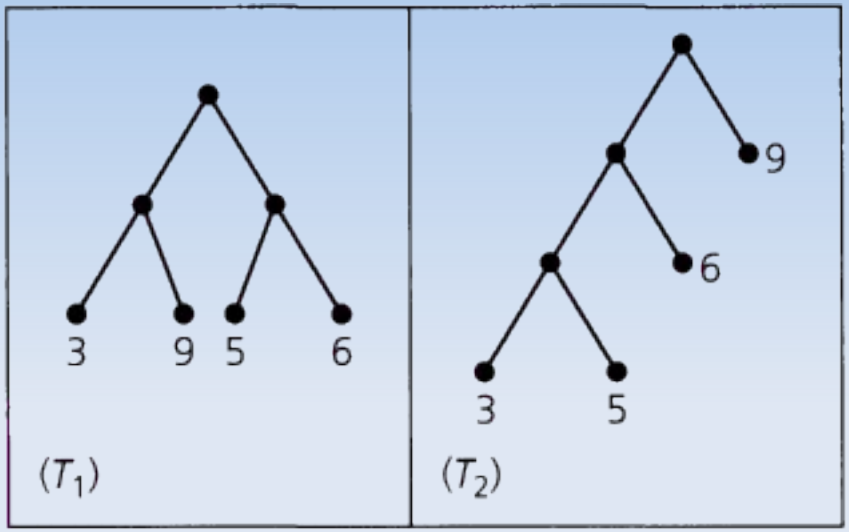
1925 – 1999

1951

O čo vlastne išlo?

- Majme váhy (kladné čísla) $w_1, w_2, \ldots w_n$, kde $w_1 \leq w_2 \leq \ldots \leq w_n$.
- Kompletný binárny strom pre váhy $w_1, w_2, \ldots w_n$ je taký strom s n listami, v ktorom každému listu priradíme niektorú z váh.
- Váha stromu T je definovaná ako $W(T) = \sum_{i=1} w_i l(w_i)$, kde $l(w_i)$ je hĺbka listu s priradenou váhou w_i
- Optimálnym nazveme takýto binárny strom vtedy, ak je jeho váha najmenšia možná.

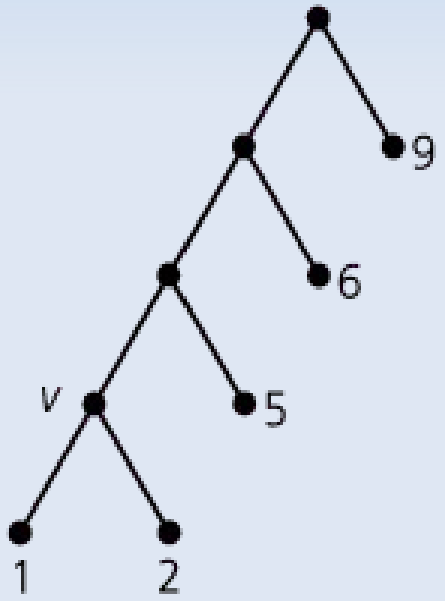
O čo vlastne išlo?



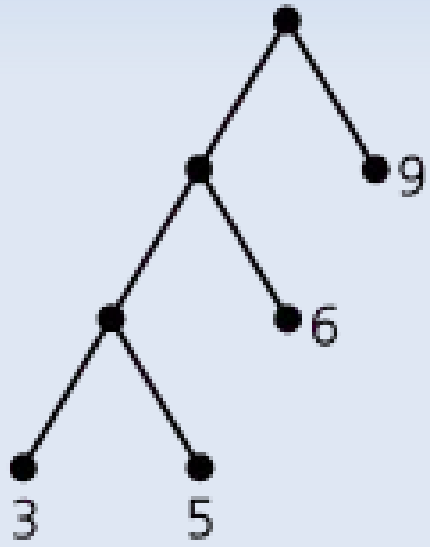
$$W(T_1)=46 \; W(T_1)=46$$

Hlavná myšlienka

Strom n váh $w_1, w_2, \ldots w_n$, zostrojíme zo stromu pre *n-1* váh $w_1 + w_2, \ldots w_n$.



1, 2, 5, 6, 9



1+2, 5, 6, 9

Huffmanov kód

Ak T je optimálny strom pre n váh w_1+ $w_{2,}\ldots,w_n$, kde $w_1 \leq w_2 \leq \ldots \leq w_n$ potom ak k vrchol s priradenou váhou w_1+ w_2 nahradím binárnym stromom hĺbky 1 s váhami listov $w_{1,}w_2$, dostanem optimálny strom T' pre váhy $w_{1,}w_{2,}\ldots,w_n$.

Huffmanov kód

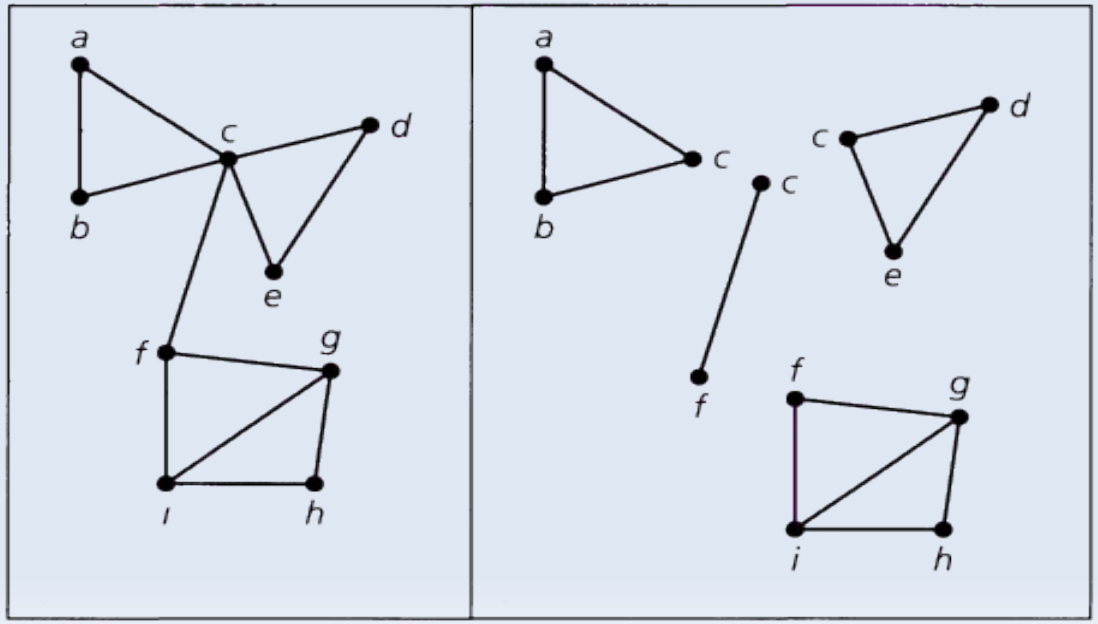
Ak T je optimálny strom pr n váh $w_1 \leq w_2 \leq \ldots \leq w_n$, potom existuje optimálny strom T' v ktorom sú listy s váhami w_1, w_2 súrodencami a majú najväčšiu hĺbku v strome T' .

Step 1. Create a parentless node for each symbol. Each node should include the symbol and its probability.
Step 2. Select the two parentless nodes with the lowest probabilities.
Step 3. Create a new node which is the parent of the two lowest probability nodes.
Step 4. Assign the new node a probability equal to the sum of its children's probabilities.
Step 5. Repeat from Step 2 until there is only one parentless node left.

Artikulačné body a dvjosúvislé komponenty

Vrchol v v neorientovanom grafe $G(V,E)$ bez slučiek sa nazýva **artikulačným bodom**, ak G-v má viac komponentov ako graf G.

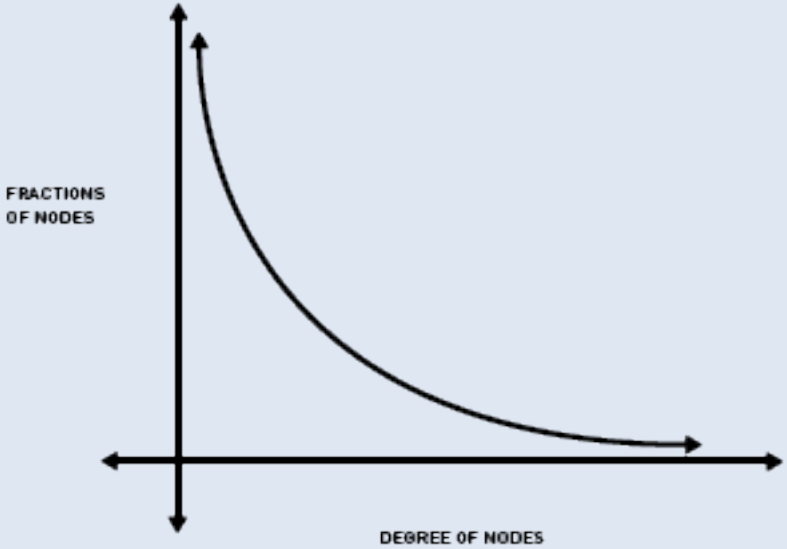
Ak graf nemá artikulačný bod, nazýva sa dvojsúvislý (spojitý) - biconnected

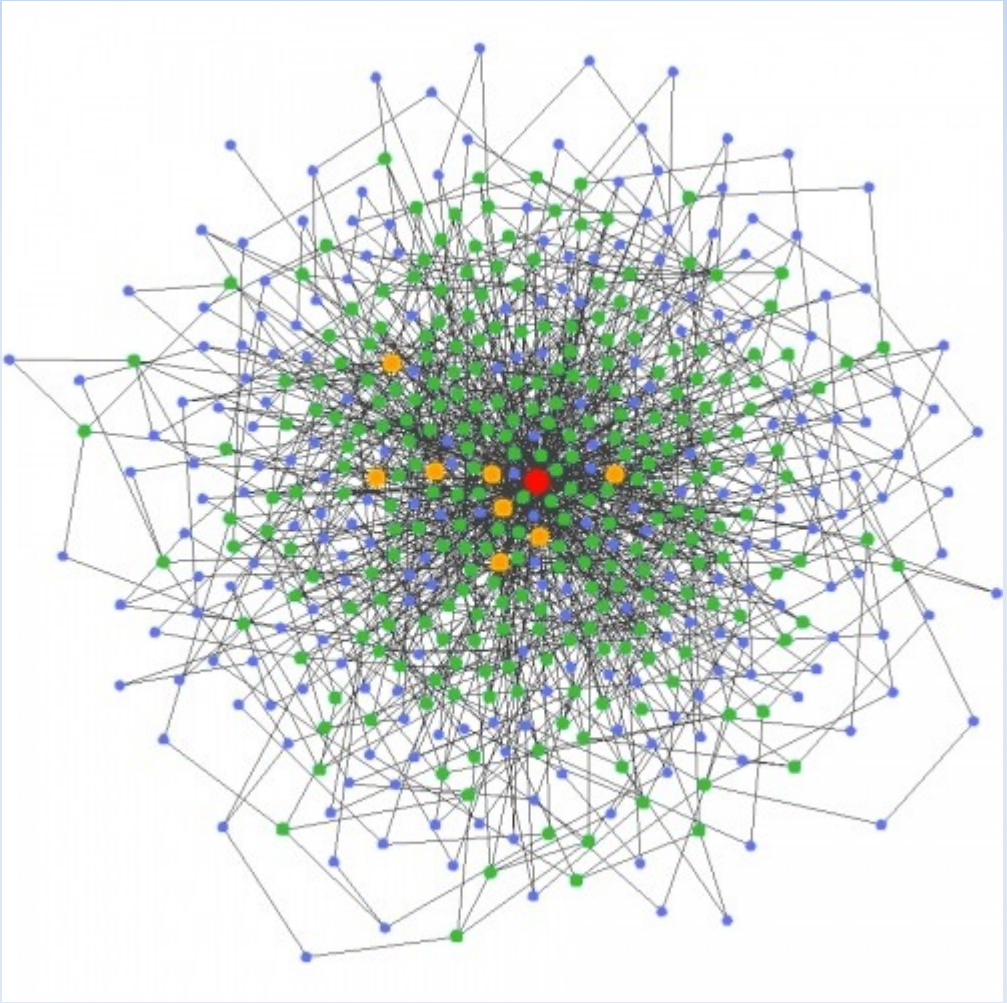


Artikulačné body a dvjosúvislé komponenty

Bezškálová sieť

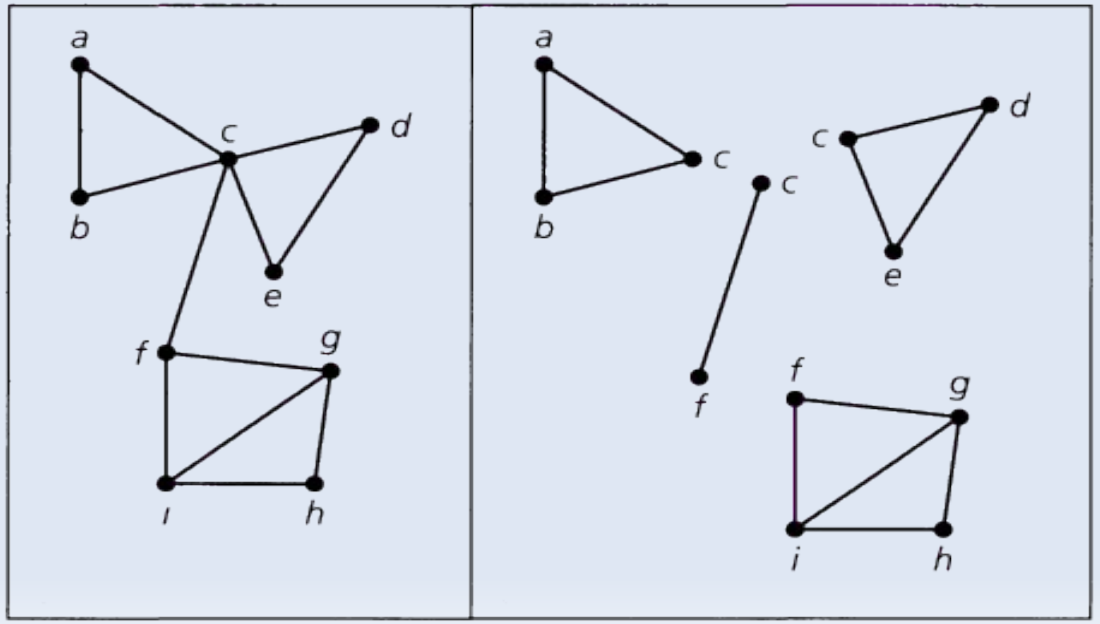
$$P(k) \sim k^{-\gamma}$$





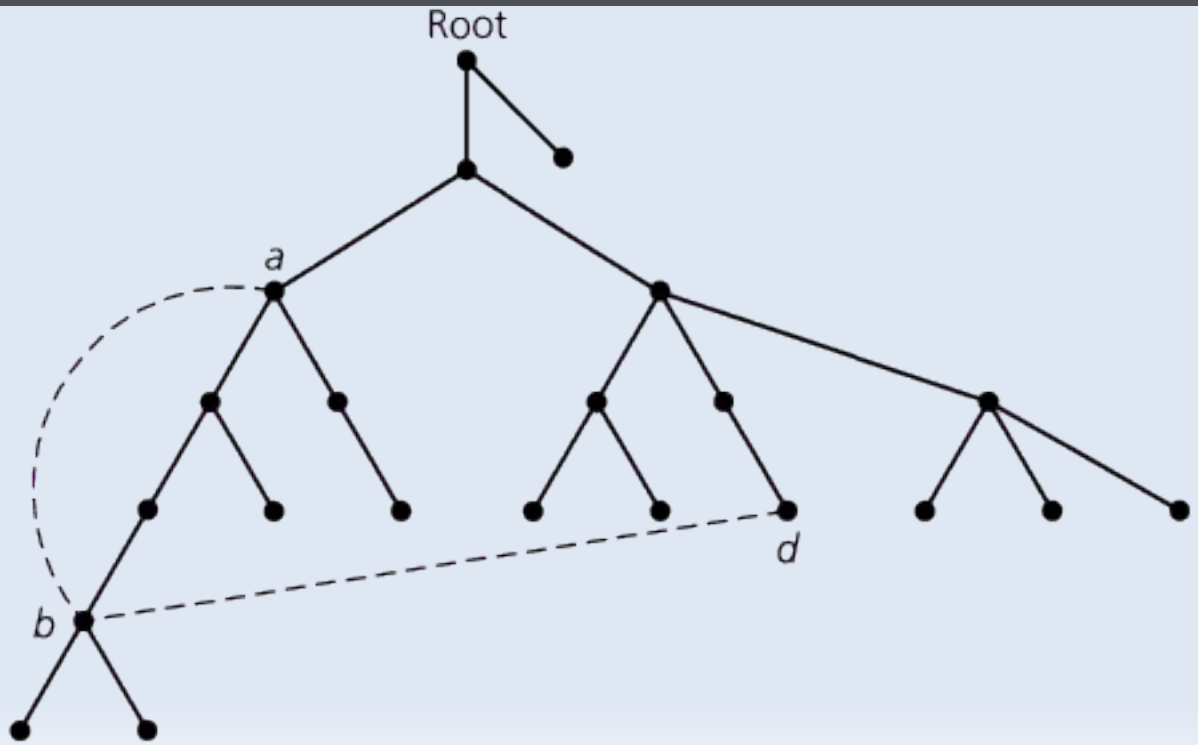
Artikulačné body a dvjosúvislé komponenty

Lema A1: Vrchol v je artikulačným vrcholom vtedy a len vtedy, ak v danom grafe existujú také dva vrcholy, že všetky cesty medzi nimi prechádzajú cez vrchol v.

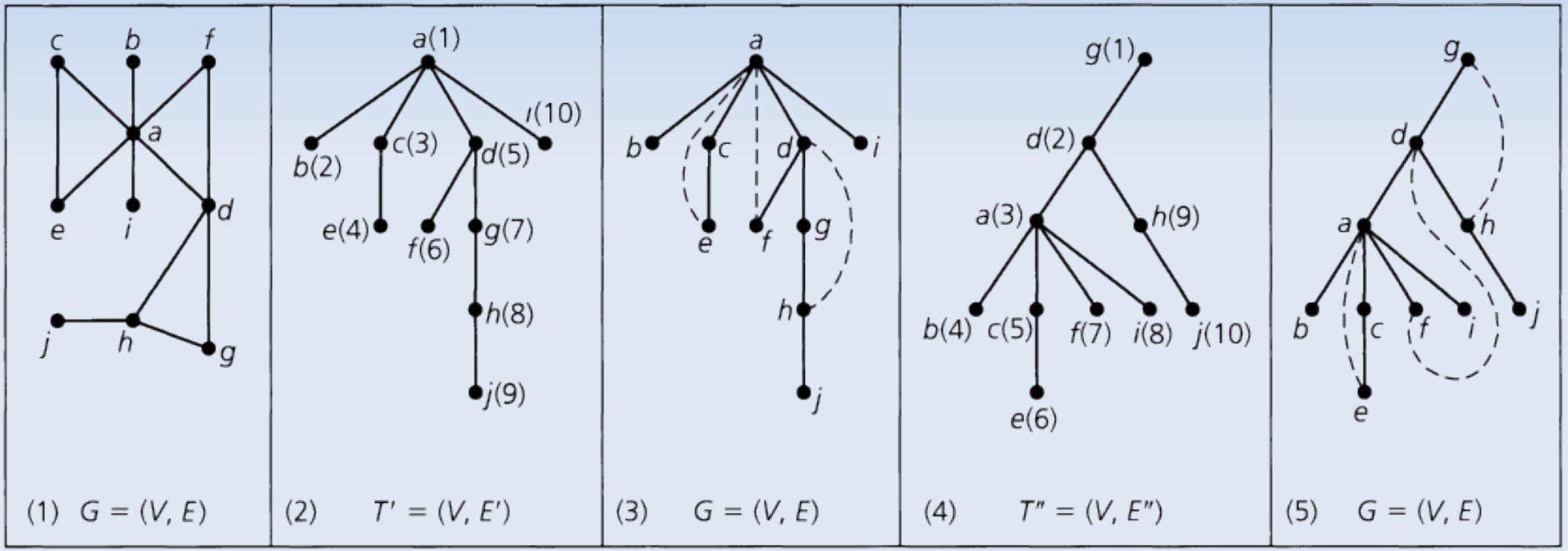


Artikulačné body a dvjosúvislé komponenty

Lema A2: Majme neorientovaný graf $G=(V,E)$ a jeho kostru (strom) $T=(V,E')$. Potom ak existuje hrana (a,b) v E, taká, že (a,b) nie je z E', potom a je predok alebo potomok b v strome T.

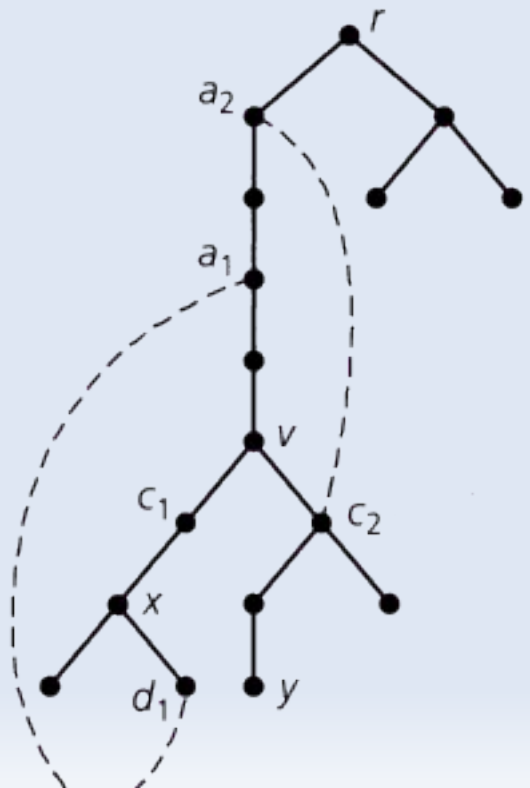


Artikulačné body a dvjosúvislé komponenty



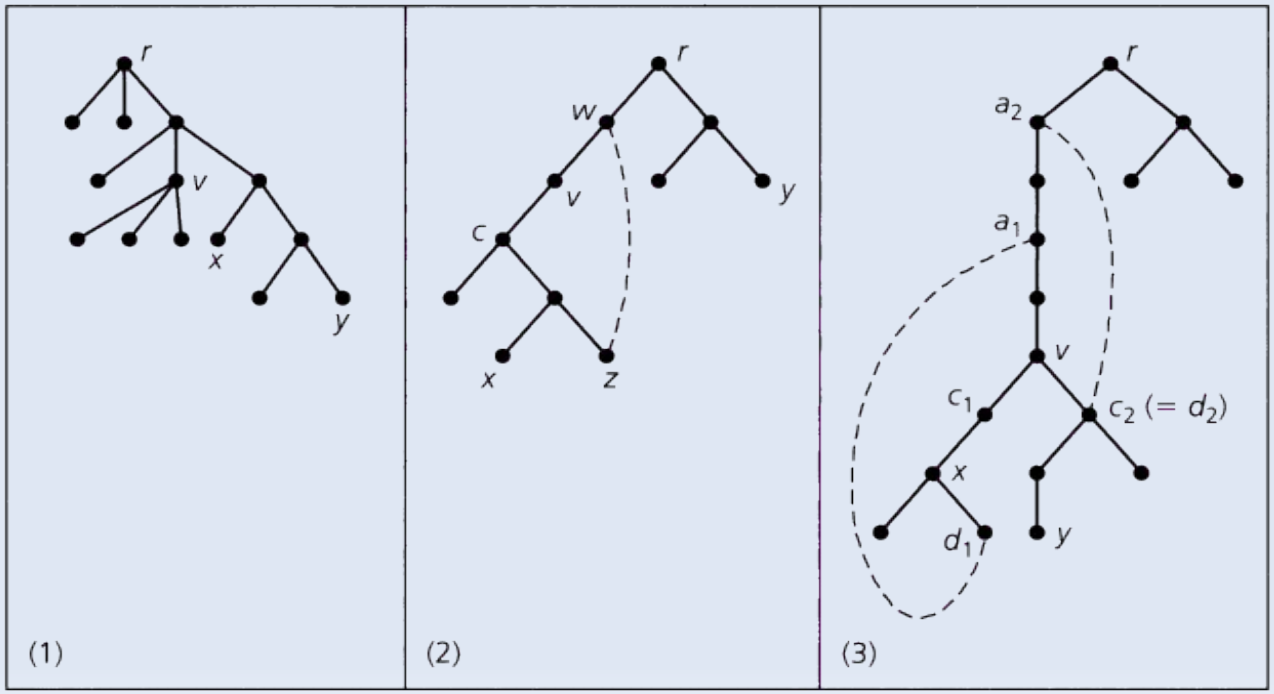
Artikulačné body a dvjosúvislé komponenty

Lema A3: Majme neorientovaný graf $G=(V,E)$ a jeho kostru (strom) $T=(V,E')$. Potom koreň r stromu T je artikulačný bod, vtedy a len vtedy ak má aspoň dve deti.



Artikulačné body a dvjosúvislé komponenty

Lema A4: Majme neorientovaný graf $G=(V,E)$ a jeho kostru (strom) $T=(V,E')$. Potom vrchol v, ktorý nie je koreňom, je artikulačným podom vtedy a len vtedy, ak má dieťa, z ktorého podstromu, neexistuje ”back-edge” do predka vrcholu v.



Artikulačné body a dvjosúvislé komponenty

Step 1: Find the depth-first spanning tree T for G according to a prescribed order. Let $x_1, x_2, \ldots, x_n$ be the vertices of G preordered by T. Then $\mathrm{dfi}(x_j) = j$ for all $1 \leq j \leq n$.

Step 2: Start with x_n and continue back to $x_{n-1}, x_{n-2}, \ldots, x_3, x_2, x_1$, determining $\mathrm{low}(x_j)$, for $j = n, n-1, n-2, \ldots, 3, 2, 1$, recursively, as follows:

a) $\mathrm{low}'(x_j) = \min\{\mathrm{dfi}(z) | z \text{ is adjacent in } G \text{ to } x_j\}$.

b) If $c_1, c_2, \ldots, c_m$ are the children of x_j, then $\mathrm{low}(x_j) = \min\{\mathrm{low}'(x_j), \mathrm{low}(c_1), \mathrm{low}(c_2), \ldots, \mathrm{low}(c_m)\}$. [No problem arises here, for the vertices are examined in the reverse order to the given preorder. Consequently, if c is a child of p, then $\mathrm{low}(c)$ is determined before $\mathrm{low}(p)$.]

Step 3: Let w_j be the parent of x_j in T. If $\mathrm{low}(x_j) = \mathrm{dfi}(w_j)$, then w_j is an articulation point of G, unless w_j is the root of T and w_j has no child in T other than x_j. Moreover, in either situation the subtree rooted at x_j together with the edge $\{w_j, x_j\}$ is part of a biconnected component of G.

Artikulačné body a dvjosúvislé komponenty