Indukcí dokážeme, že $A_k \geq 2^{\frac{k}{2}}$. První indukční krok jsme si už ukázali, teď pro $k \geq 2$ platí: $A_k = 1 + A_{k-1} + A_{k-2} > 2^{\frac{k-1}{2}} + 2^{\frac{k-2}{2}} = 2^{\frac{k}{2}} \cdot (2^{-\frac{1}{2}} + 2^{-1}) \cong 2^{\frac{k}{2}} \cdot 1.21 > 2^{\frac{k}{2}}$.

Tímto jsme dokázali, že na každé hladině je minimálně exponenciálně vrcholů, což nám zaručuje hloubku $\Theta(\log n)$. ♡

11. Vyhledávací stromy

Představme si následující problém: potřebujeme si udržovat určitá setříděná data, například slovník. Položkami jsou uspořádané dvojice (klíč, hodnota). Klíče jsou unikátní a jsou to prvky nějakého lineárně uspořádaného universa. Hodnoty mohou být libovolné.

Na našich datech budeme chtít provádět následující operace:

- *Insert* – vložit novou položku
- *Delete* – smazat položku
- *Find* – najít položku
- *Max* & *Min* – vybrat položku s největším, resp. nejmenším klíčem
- *Pred* & *Succ* – vybrat položku s klíčem nejbližším menším, resp. větším

Jak by vypadalo nejjednodušší řešení? Staticky bychom data mohli udržovat v setříděném poli. Takové pole se vyrobí v čase $\Theta(n \cdot \log n)$. Operace *Find* by trvala $\Theta(\log n)$ (vyhledávali bychom samozřejmě binárně). Ale problém by nastal s operacemi *Insert* a *Delete*, které by se takto implementovat nedaly, nebo by trvaly hodně dlouho (např. v případě *Insertu* bychom museli celé pole přestavět).

Pozorování: Proces binárního vyhledávání v setříděném poli se dá reprezentovat binárním vyhledávacím stromem.

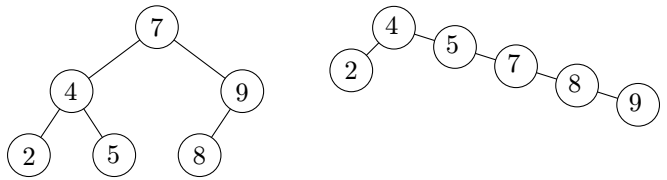
Definice: *Binární strom:* Strom je binární, pokud je zakořeněný a každý vrchol má nejvýše dva syny, u nichž rozlišujeme, který je levý a který pravý.

Definice: Pro vrchol v značíme:

- $l(v)$ a $p(v)$ – levý a pravý syn vrcholu v
- $L(v)$ a $P(v)$ – levý a pravý podstrom vrcholu v
- $S(v)$ – příslušný podstrom s kořenem v
- $h(v)$ – hloubka stromu $S(v)$, neboli délka nejdelší cesty z kořene do listu

Definice: *Binární vyhledávací strom* (BVS): Binární strom je vyhledávací, pokud v každém vrcholu je uložena dvojice (klíč, hodnota) [ztotožníme vrchol s klíčem] a pro všechny vrcholy platí: $(\forall u \in L(v) : u < v)$ & $(\forall u \in P(v) : u > v)$.

Příklady binárních vyhledávacích stromů:



Jak budou tedy vypadat operace *Find*, *Insert* a *Delete* na binárním vyhledávacím stromu?

Find(v, x)**:**

1. Pokud $v = \emptyset \Rightarrow$ vrátíme $\emptyset$.
2. Pokud $v = x \Rightarrow$ vrátíme v.
3. Pokud $v < x \Rightarrow$ vrátíme $Find(p(v), x)$.
4. Pokud $v > x \Rightarrow$ vrátíme $Find(l(v), x)$.

Insert(v, x)**:**

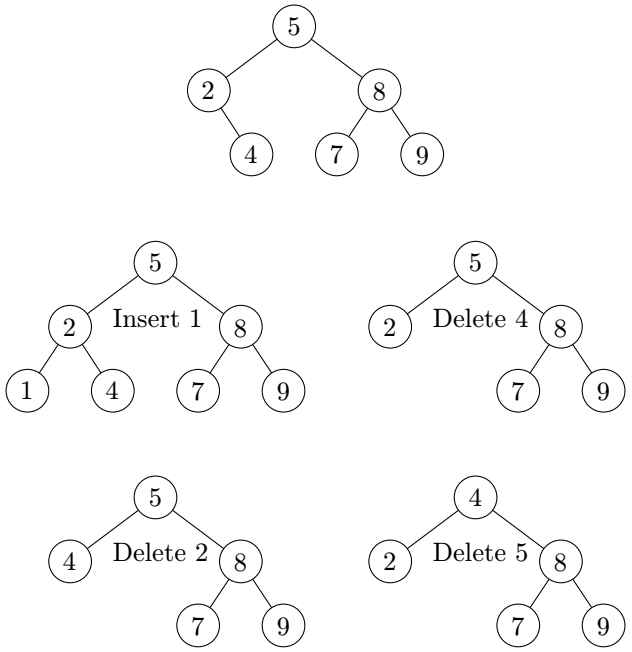
1. Jako *Find* a na konci přidáme list.

Delete(v)**:**

1. Pokud v je list $\Rightarrow$ jednoduše list utrhneme.
2. Pokud v má jednoho syna $\Rightarrow$ vrchol „vyřízneme“.
3. Jinak má v oba syny $\Rightarrow$ do vrcholu vložíme minimum z $P(v)$, což bude list, a ten utrhneme.

Poznámka: Pokud má vrchol v při operaci *Delete* oba syny, je vložení minima z $P(v)$ ekvivalentní s vložením maxima z $L(v)$.

Příklady operací *Insert* **a** *Delete* **na BVS:**



Časová složitost všech tří operací je $\Theta(\textit{hloubka stromu})$, což může být $\Theta(n)$, když budeme mít smůlu a strom bude (téměř) lineární spojový seznam, nebo $\Theta(\log n)$ když bude strom pěkně vyváženě vystavěný. Vidíme tedy, že složitost operací stojí a padá s hloubkou stromu. Proto by se nám líbilo, aby měl náš strom vždy hloubku $\Theta(\log n)$. Podívejme se tedy, jak se dá navrhnout binární vyhledávací strom, aby tuto podmínku splňoval ...

Vyvážené binární vyhledávací stromy

Definice: *Dokonalá vyváženost:* Strom je dokonale vyvážený, pokud pro všechny jeho vrcholy platí: $||L(v)| - |P(v)|| \leq 1$.

Takto definovaný binární strom bude mít určitě logaritmickou hloubku. Jak takový strom ale konstruovat? To se nám podaří buď staticky, nebo na něm budou operace dražší než $\Theta(\log n)$.

Statická konstrukce dokonale vyváženého BVS: Vybereme prostřední prvek ze setříděného pole (tedy medián posloupnosti) a dáme ho do kořene stromu. Jeho syny pak vystavíme rekurzivně z levé a pravé půlky pole. Celá konstrukce tedy trvá $\mathcal{O}(n)$.

Vidíme tedy, že to náš problém příliš neřeší. Potřebovali bychom, aby se strom dal také efektivně udržovat. Zkusíme proto slabší podmínku:

Definice: *Hloubková vyváženost:* Strom je hloubkově vyvážený, pokud pro všechny jeho vrcholy platí: $|h(L(v)) - h(P(v))| \leq 1$.

Stromům s hloubkovým vyvážením se říká *AVL stromy* (objeviteli je ruští matematikové G. M. Adelson-Velsky a E. M. Landis, podle nich jsou také pojmenovány) a platí o nich následující lemma:

Lemma: AVL strom na n vrcholech má hloubku $\Theta(\log n)$.

Důkaz: Uvažme posloupnost $A_k =$ minimální počet vrcholů AVL stromů hloubky k. Stačí ukázat, že A_k roste exponenciálně. Podívejme se na minimální AVL stromy:

$$\begin{aligned} A_0 &= 1 \\ A_1 &= 2 \\ A_2 &= 4 \\ A_3 &= 7 \\ &\vdots \\ A_k &= 1 + A_{k-1} + A_{k-2}. \end{aligned}$$

Rekurentní vzorec jsme dostali rekurzivním stavěním stromu hloubky k: nový kořen a 2 podstromy o hloubkách $k-1$ a $k-2$.

Vidíme tedy, že $A_n = F_{n+2} - 1$. (Můžeme dokázat např. indukcí.) Teď nám již stačí dokázat, že posloupnost A_k roste exponenciálně S výhodou můžeme využít toho, že na první přednášce jsme si již dokázali, že Fibonacciho čísla rostou exponenciálně. Nicméně pro zapomnětlivé můžeme důkaz ve stručnosti zopakovat: