

▲ **Def:** Necht'  $(K, \leq)$  je úplně uspořádaná množina tzv. *klíčů*,  $V$  je libovolná množina tzv. doplňujících údajů. Necht'  $U = K \times V$  je množina dvojic  $(k, v)$ , v níž se každý klíč  $k$  vyskytuje nejvýše jednou (tj. každý záznam z množiny  $U$  je určen jednoznačně svým klíčem). Problému nalézt k danému klíči  $k$  záznam  $(k, v) \in U$  se říká *vyhledávací problém*.

▲ Mějme pole hodnot  $A$ . Hodnoty jsou v poli  $A$  rozmístěny náhodně. Budeme-li předpokládat, že v tomto poli budeme hledat pouze  $k$ -krát, kde  $k$  je malé přirozené číslo (například 3), jak byste postupovali při hledání prvku v tomto poli? Jak byste postupovali v případě, že by bylo vyžadováno časté hledání prvků v tomto poli?

▲ Algoritmus binárního vyhledávání:

```
type Elem = Integer;
   Pole = array [1..999] of Elem;

function bSearch(k:Elem; var D: Pole; n: Integer): Integer;
{Posl. D je rostoucí. Když D[i]=k, tak bSearch(k)=i, jinak bSearch(k)=-1}
  function bs (l, r: Integer):Integer;
  begin
    if l>r then bs := -1 {nenalezeno}
      else
        begin
          m := (l+r) div 2;
          if k < D[m] then bs := bs(l,m-1);
            else if k > D[m] then bs := bs(m+1,r);
              else {k=D[m]} bs := m;
        end
      end
  end
begin bSearch := bs (1,n); end
```

▲ Jaká je složitost binárního vyhledávání?

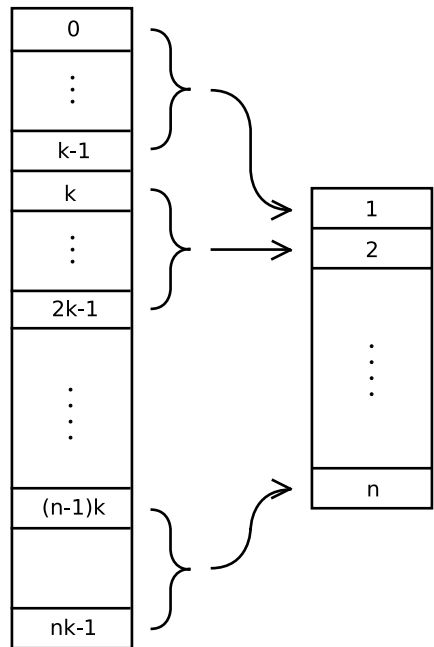
▲ V poli  $A = [1, 4, 5, 6, 11, 13, 17, 18]$ , nalezněte binárním vyhledáváním index prvku s hodnotou klíče 17. Ve stejném poli nalezněte index prvku s hodnotou klíče 2.

▲ Hašovací tabulka je datová struktura, pomocí níž lze prakticky efektivně realizovat "slovníkové" operace vyhledání, přidání a zrušení položky.

Prakticky efektivně znamená, že operace mají příznivou průměrnou časovou složitost (tedy ne nutně časovou složitost v nejhorším případě).

Hašovací tabulka je jednorozměrné pole  $H$  indexované čísly  $1, \dots, n$ . Převod klíčů na čísla realizuje hašovací funkce  $h : K \rightarrow \{1, \dots, n\}$ . Výpočet hodnot hašovací funkce musí být efektivní, nejlépe složitosti  $\Theta(1)$ .

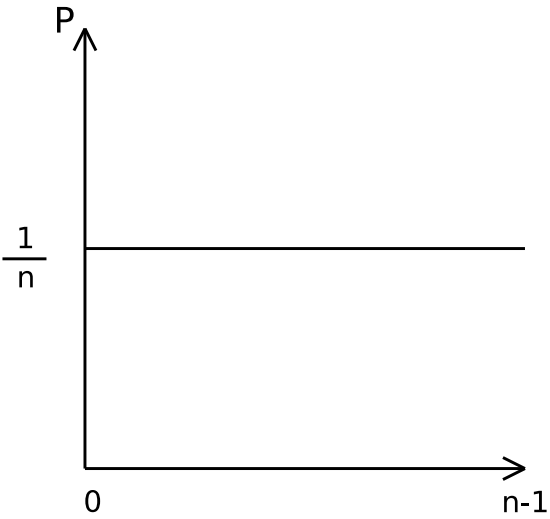
Příklad:



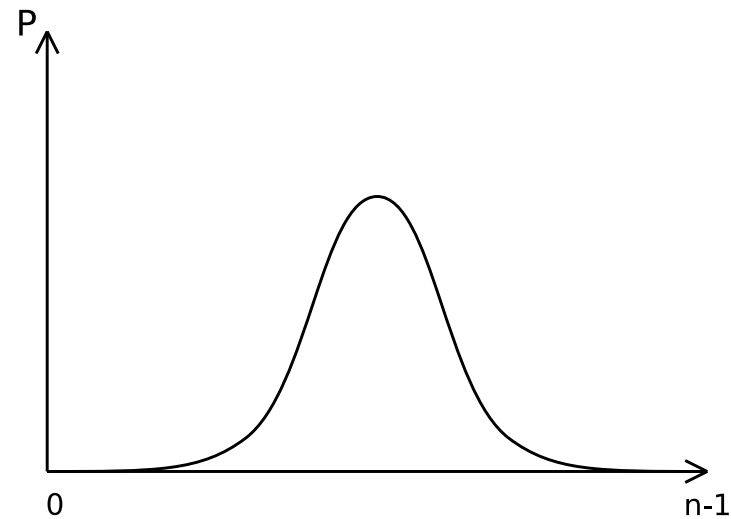
▲ Jaké hašovací funkce jsou použity v následujících případech?

- Vyhledávání v telefonním seznamu.
- Psaní písmen na mobilním telefonu.
- Rozřazování mužstva s pomocí pravidla "první, druhý, první, druhý, ...".

▲ Navrhněte hašovací funkci pro data, která mají pravděpodobnost výskytu svých prvků danou níže uvedenými funkcemi. Funkci navrhněte tak, aby hodnoty byly v hašovací tabulce distribuovány rovnoměrně.



(a) Uniformní



(b) Gaussián

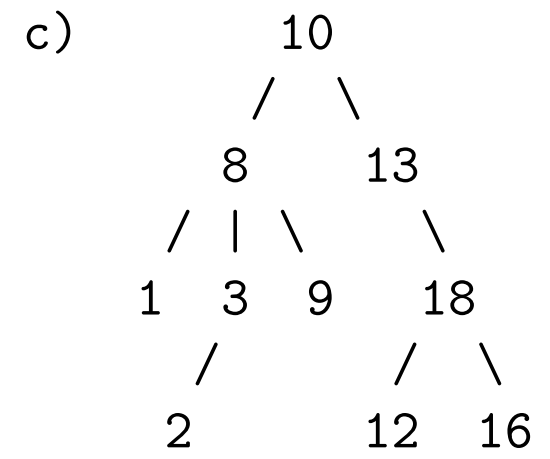
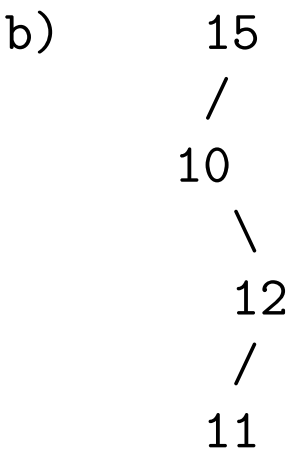
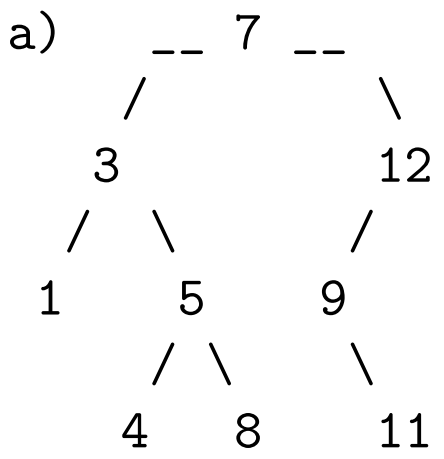
▲ Mějme uniformní hašovací funkci, která mapuje interval  $0..19$  na  $1, \dots, 80..99$  na 5. Vložte do hašovací tabulky postupně čísla  $[57, 60, 74, 35, 16, 61, 7, 49, 86, 98]$ . Pro řešení kolizí použijte jednosměrně zřetěžený seznam.

▲ Předpokládejme hašovací funkci  $h$ , která mapuje  $n$  různých hodnot do pole  $T$  délky  $m$ . Jaký je předpokládaný počet kolizí uvažujeme-li uniformní hašování? To jest, u kolika z uložených  $n$  prvků lze očekávat, že budou na stejné pozici s jinými prvky?

▲ **Def:** *Binární vyhledávací strom* (BVS) je binární strom nad úplně uspořádanou množinou (tzv. klíčů)  $(K, \leq)$  takový, že pro každý jeho podstrom  $t$  platí:

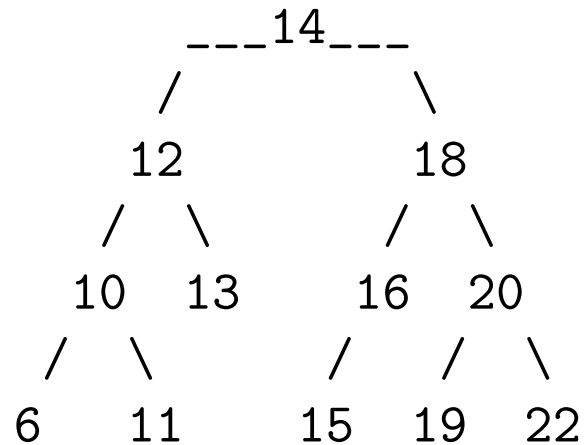
- hodnoty uzlů v podstromu  $\text{left}(t)$  jsou menší než  $\text{rootval}(t)$  a hodnoty uzlů v podstromu  $\text{right}(t)$  jsou větší než  $\text{rootval}(t)$

Rozhoděte, zda následující stromy jsou BVS. Odpověď zdůvodněte.



▲ Při praktické implementaci BVS lze každý jeho uzel reprezentovat pomocí struktury Node obsahující čtyři atributy: hodnotu klíče Node.key, ukazatel na rodiče Node.parent, ukazatel na levého Node.left a pravého Node.right potomka.

```
Node = record
  key : Integer
  left, right : ^Node
  parent : ^Node
end
```



Nechť Node20 označuje ukazatel na uzel s hodnotou klíče 20. V daném BVS určete čemu se rovnají následující výrazy:

- $((\text{Node20.parent}).\text{left}).\text{left}).\text{key}$
- $((\text{Node13.parent}).\text{parent}).\text{parent}$
- $((\text{Node14.left}).\text{left}).\text{right}$
- $((((\text{Node12.parent}).\text{right}).\text{right}).\text{left}).\text{key}$



▲ Zkonstruuje BVS postupným vkládáním uzlů 16, 5, 9, 28, 2, 20, 18, 29, 24, 26, 22. Poté postupně odstraňte uzly 9, 5, 20. Nakonec vyhledejte uzly 24 a 9.

▲ Předpokládejme, že máme čísla mezi 1 a 1000 v BVS a hledáme číslo 363. Které z následujících sekvencí nemůžou být sekvencemi uzlů při hledání této hodnoty?

a) 2, 252, 401, 398, 330, 344, 397, 363

b) 924, 220, 911, 244, 898, 258, 362, 363

c) 925, 202, 911, 240, 912, 245, 363

d) 2, 399, 387, 219, 266, 382, 381, 278, 363

e) 935, 278, 347, 621, 299, 392, 358, 363

▲ Mějme definován binární vyhledávací strom a operace nad ním následujícím způsobem:

```
Node = record      |      function Init(T)
  key : Integer    |      begin
  left,right : ^Node |      T.root = nil
  parent : ^Node   |      end
end                |
```

---

```
                                |      function Search(T,k)
BVS = record                    |      begin
  root : ^Node                  |      x := T.head
end                              |      while (x <> nil) AND (x.key <> k) do
                                |      if (k < x.key) then
                                |      x := x.left
                                |      else
                                |      x:= x.right
                                |      return x
                                |      end
                                |
```

function Minimum(z)		function Successor(z)
begin		begin
while (z.left <> nil) do		if (z.right <> nil) then
z := z.left		return Minimum(z.right)
return z		y := z.parent
end		
		while (y <> nil AND
function Maximum(z)		z = y.right) do
begin		z := y
while (z.right <> nil) do		y := y.parent
z := z.right		
		return y
return z		end
end		

```

function Delete(T,z)
begin
    if (z.left = nil OR
        z.right = nil) then
        y := z
    else
        y := Successor(T,z)

    if (y.left <> nil) then
        x := y.left
    else
        x := y.right

    if (x <> nil) then
        x.parent := y.parent

    if (y.parent = nil) then
        T.root := x
    else
        if (y = (y.parent).left) then
            (y.parent).left := x
        else
            (y.parent).right := x

    if (y <> z) then
        z.key := y.key

    return y
end

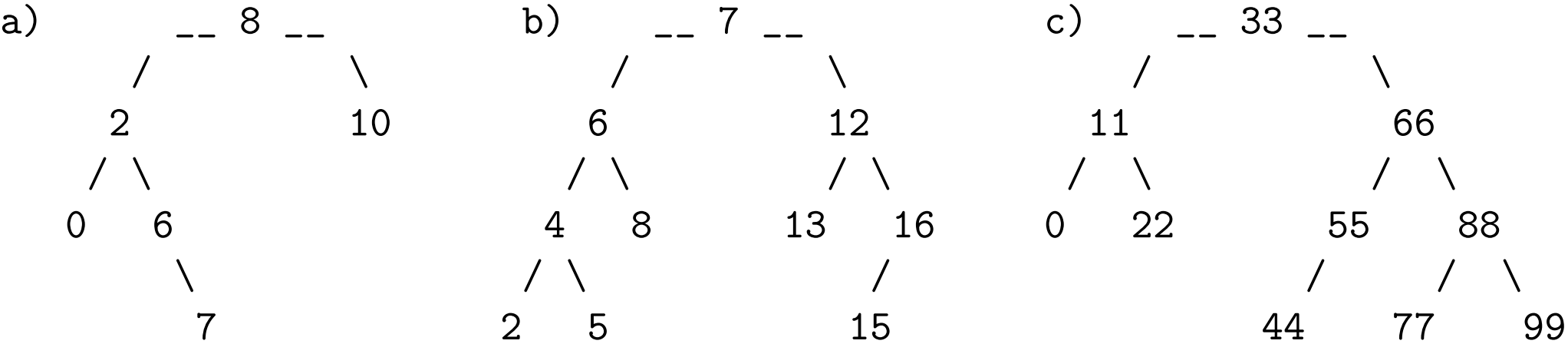
```

Navrhněte implementaci funkce `Insert(T,z)`, která vloží do stromu `T` uzel `z`.

▲ Určete časovou složitost jednotlivých operací nad BVS.

▲ Def.: AVL strom je binární vyhledávací strom, u kterého se hloubka levého a pravého podstromu libovolného uzlu liší nejvýše o jedničku. Příkladem AVL stromu je Fibonacciho strom. Vyhledávání v AVL stromu se neliší od vyhledávání v běžném vyhledávacím stromu. Při přidávání položky do AVL stromu se musí kontrolovat vyváženost. Podobně při rušení položky.

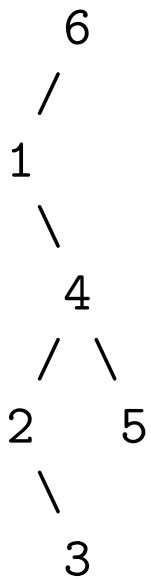
▲ Rozhodněte, zda jsou následující stromy AVL.



▲ Postupným vkládáním klíčů 8, 3, 5, 0, 2, 4, 1, 6, 9, 7 vybudujte AVL strom.

▲ Z výsledného AVL stromu z předchozího příkladu odeberte postupně uzly s hodnotami 7, 5, 6 a 1.

▲ Následující binární vyhledávací strom upravte tak, aby byl AVL stromem.

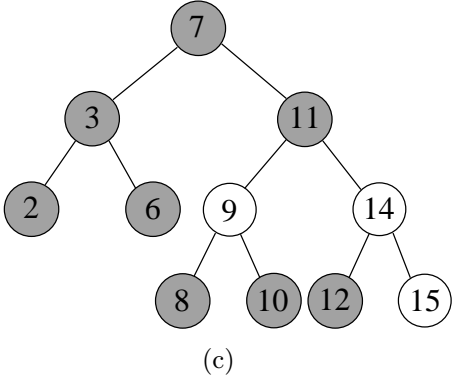
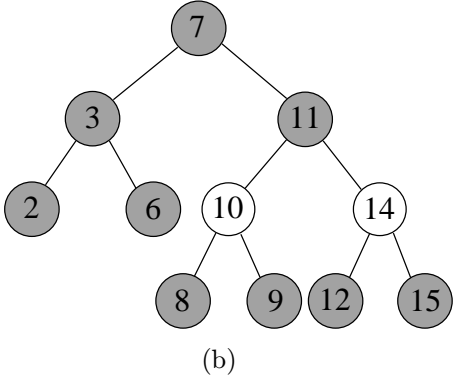
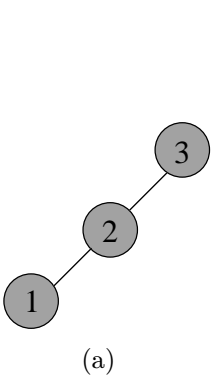


▲ Jaká je složitost operací vyhledání, přidání a odebrání uzlu v AVL stromu?

▲ **Def:** *Červenočerný strom* je binární vyhledávací strom, jehož každý uzel je obarven černou nebo červenou barvou. Musí splňovat tyto podmínky:

- kořen stromu je černý
- je-li vnitřní uzel červený, jeho následníci (pokud existují) jsou černí
- všechny větve obsahují stejný počet černých uzlů

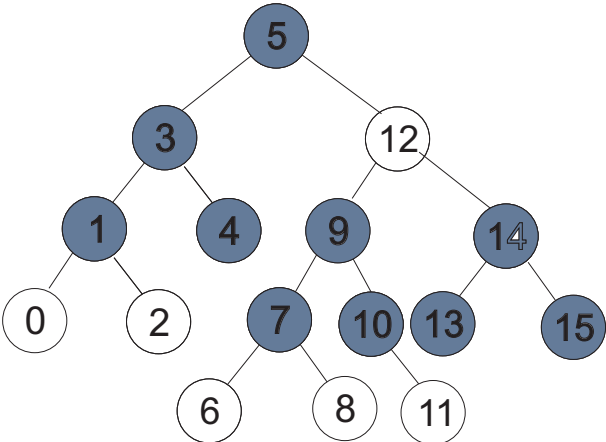
▲ Rozhodněte, zda jsou následující stromy červenočerné stromy.



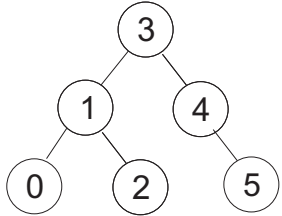
▲ Zkonstruuje červenočerný strom postupným vkládáním uzlů 12, 5, 9, 18, 2, 15, 13, 19, 17. Poté postupně odstraňte uzly 9, 5, 15. Nakonec vyhledejte uzly 17 a 9.



▲ Z následujícího červenočerného stromu odstraňte uzly s hodnotou 10, 13.



▲ Kolika způsoby je možné obarvit následující BVS na červenočerný?



▲ Jaká je složitost jednotlivých operací nad BVS, pokud uvažujeme červenočerné stromy?