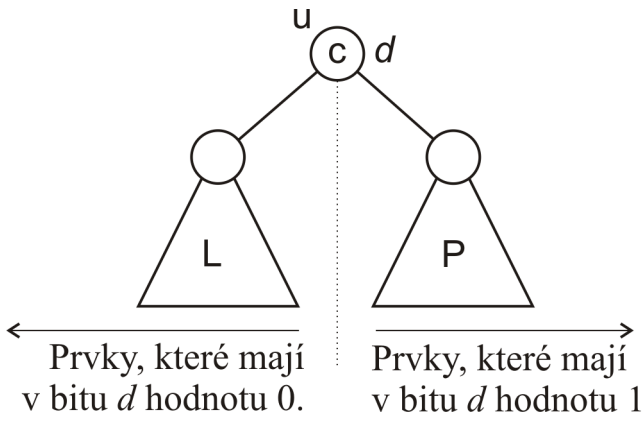
Číslicové vyhledávací stromy (*Digital Search Trees*)

Číslicové vyhledávací stromy lze použít pro prvky, které jsou reprezentovány v binárním tvaru (celá čísla, řetězce).

Nejjednodušší způsob číslicového vyhledávání vychází z obecných binárních vyhledávacích stromů. Větvení v číslicovém vyhledávacím stromu je založeno na hodnotě bitu, jehož pořadí v prvku odpovídá úrovni (hloubce) daného uzlu ve stromu. Větvení v prvním (kořenovém) uzlu je dle prvního bitu prvku. Levý následník obsahuje prvek, který v prvním bitu (bity jsou brány zleva) má hodnotu 0, zatímco pravý následník obsahuje prvek, který v prvním bitu má hodnotu 1. Větvení v uzlu, který je následníkem kořenového uzlu, je dle druhého bitu prvku atd.



Ve stromu opět platí, že v každém uzlu je právě jeden prvek.

Vyhledání prvku

1. Počáteční krok

   Uzel, který je v daném okamžiku vyhledávání aktuální, budeme označovat *u*. Na začátku jím bude kořen stromu.

   Nechť hledaný prvek je *x*.

   Aktuální index (pořadí) bitu označíme *d*, první bit má index 0.

2. Průběžný krok

   Vezmeme prvek obsažený v aktuálním uzlu *u*, označme ho *c*.

   - Nejprve srovnáme, zda je $x = c$.
     Pokud ano, hledaný prvek je nalezen a vyhledávání tím úspěšné končí.
   - Jestliže $x \neq c$, zjistíme hodnotu bitu *d* prvku *x*.
     Pokud je hodnota 0, je nutné v hledání pokračovat v levém podstromu. Jako nový aktuální uzel *u* položíme levého následníka současného aktuálního uzlu, zvýšíme hodnotu *d* o 1 a opět provedeme krok 2. Pokud uzel nemá

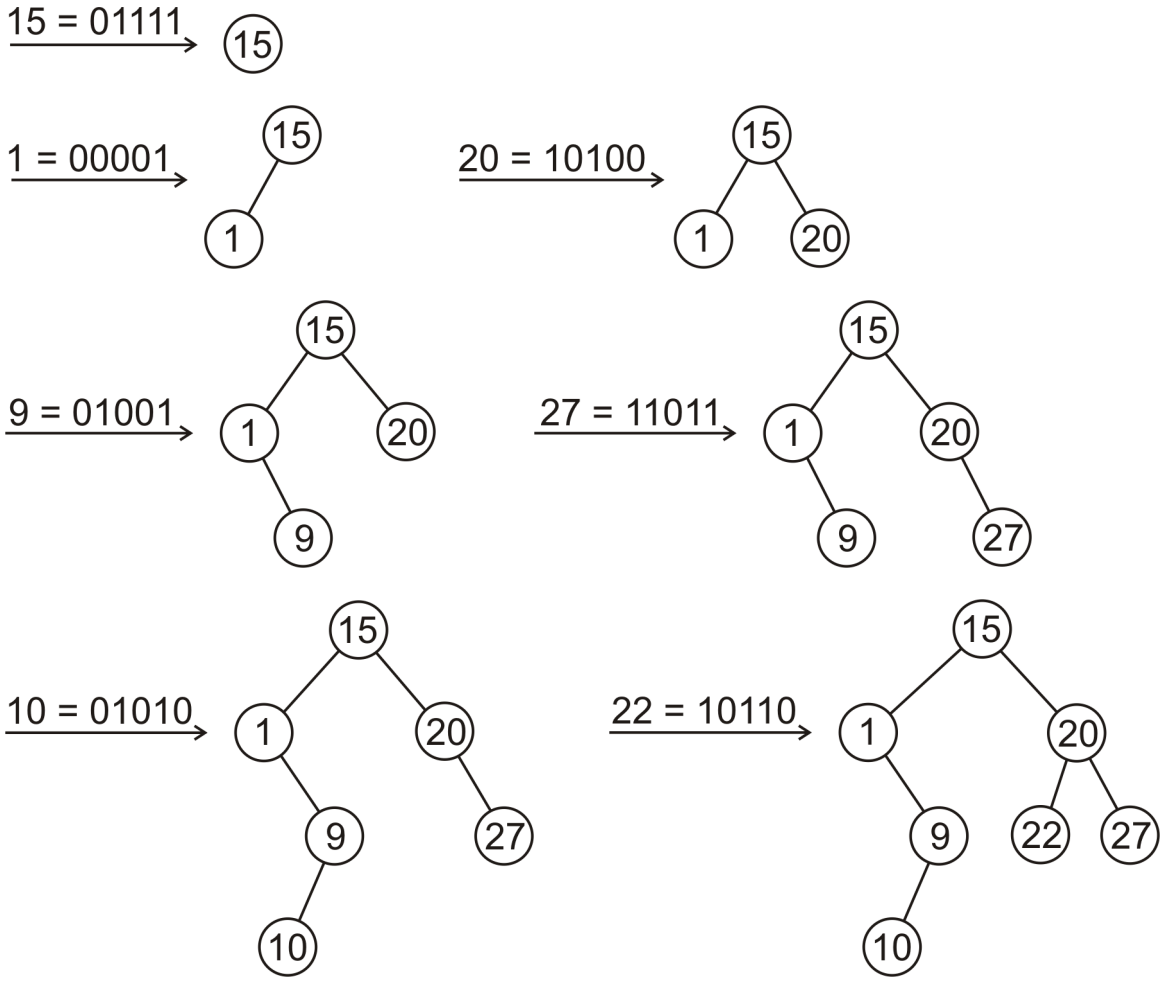
levého následníka, vyhledávání končí – hledaný prvek není ve stromu obsažen.

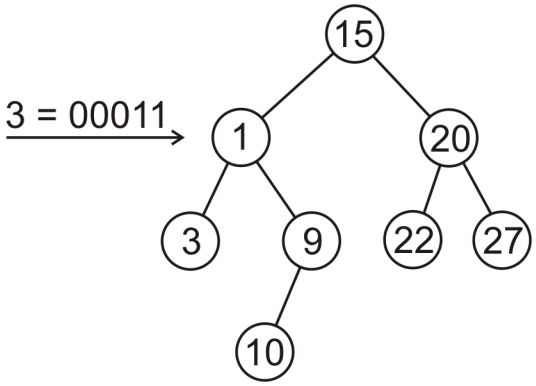
- Je-li hodnota bitu *d* rovna 1, pokračujeme obdobně ve vyhledávání v pravém podstromu, pokud aktuální uzel *u* má pravého následovníka.

Přidání prvku

Přidání prvku probíhá obdobně jako v binárním vyhledávacím stromu. Přidávaný prvek *x* nejprve vyhledáme. Pokud nebyl nalezen, vytvoříme příslušného následovníka v uzlu, ve kterém vyhledávání skončilo, a přidávaný prvek *x* do něho vložíme.

Příklad. Do stromu budeme ukládat pětibitová čísla bez znaménka (rozsah 0..31).





Pseudokód vyhledání:

```
Search(T, x)
  u ← T.root
  d ← 0
  while u ≠ NIL
    if x = u.item
      return u
    if Bit(x,d) = 0
      u ← u.left
    else
      u ← u.right
    d ← d+1
  return NIL
```

Pseudokód přidání:

```
NewNode(x)
  u ← new Node
  u.item ← x
  u.left ← u.right ← NIL
  return u

Insert(T, x)
  if T.root = NIL              // strom je prázdný, nemá kořen
    T.root ← NewNode(x)        // vytvoříme kořen
    return true
  u ← T.root
  d ← 0
  while true
```

```
    if x = u.item
      return false
    if Bit(x,d) = 0
      if u.left = NIL
        u.left ← NewNode(x)
        return true
      u ← u.left
    else
      if u.right = NIL
        u.right ← NewNode(x)
        return true
      u ← u.right
    d ← d+1
```

Odstranění prvku

Prvek nejprve vyhledáme. Pokud byl nalezen, pak se rozlišují dva případy:

- Prvek je v uzlu, který je list. Tento uzel odstraníme.
- Prvek je v uzlu, který není list – je nahrazen prvkem z kteréhokoliv listu z levého nebo pravého podstromu.

Pseudokód:

```
Delete(T, x)
  u ← T.root
  if u = NIL
    return false
  if u.item = x
    T.root ← DeleteNode(u)
    return true
  d ← 0
  while true
    if Bit(x,d) = 0
      if u.left = NIL
        return false
      if u.left.item = x
        u.left ← DeleteNode(u.left)
        return true
```

```
      u ← u.left
    else
      if u.right = NIL
        return false
      if u.right.item = x
        u.right ← DeleteNode(u.right)
        return true
      u ← u.right
    d ← d+1
IsLeaf(u)
  return u.left=NIL and u.right=NIL
DeleteNode(u)
  if IsLeaf(u)
    return NIL
  v ← u
  while true
    if v.left ≠ NIL
      if IsLeaf(v.left)
        u.item ← v.left.item
        v.left ← NIL
        return u
      v ← v.left
    else
      if IsLeaf(v.right)
        u.item ← v.right.item
        v.right ← NIL
        return u
      v ← v.right
```