Binární vyhledávací stromy 1

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

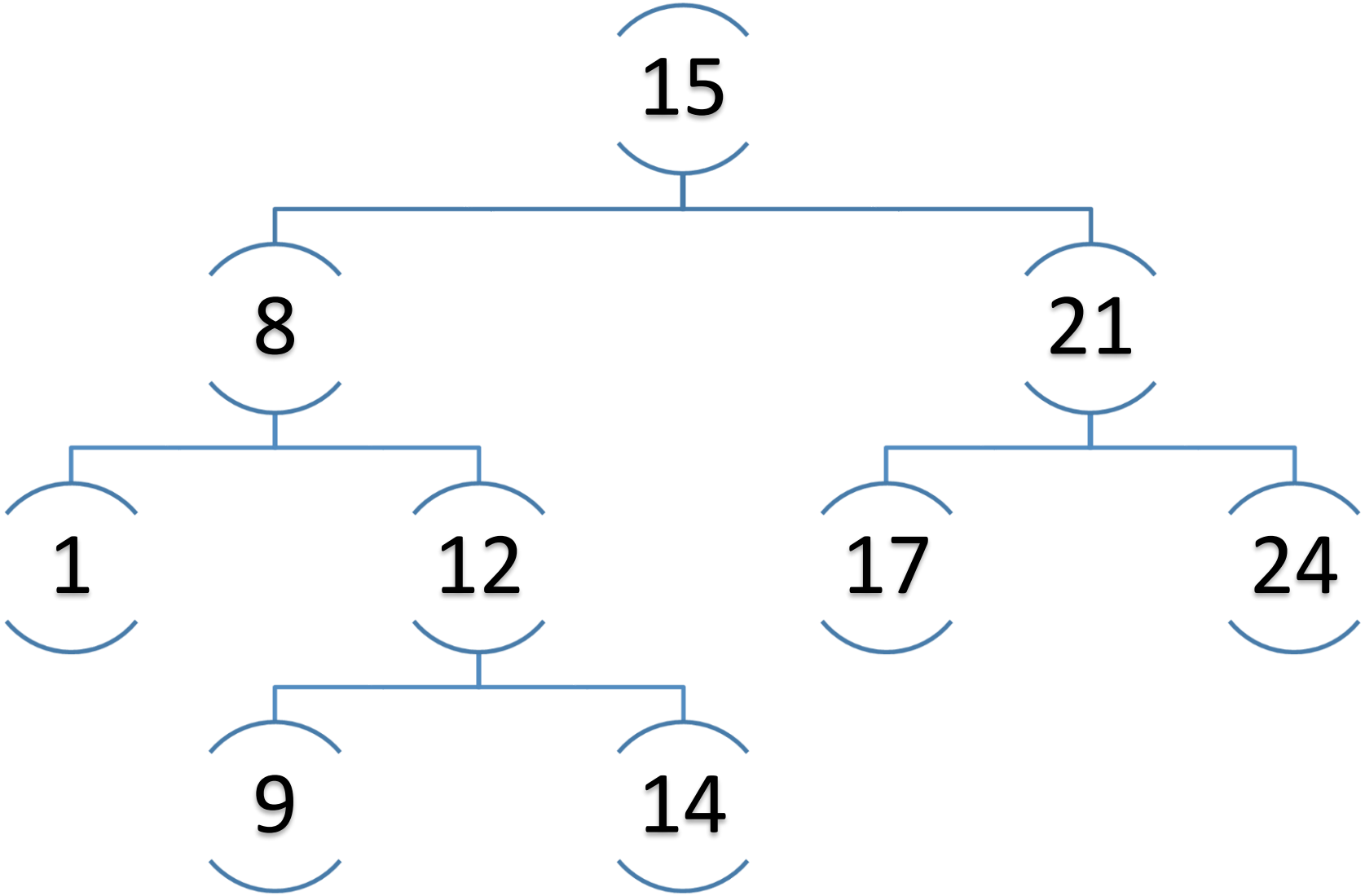
KMI/ZADS - Základní algoritmy a datové struktury

Uzel

```
{"left":levý potomek,
"right":pravý potomek,
"parent":rodič,
"key": klíč}
```

Vytvoření stromu

Příklad



Vložení prvku do stromu

```
def tree_insert(T, z):
    y = None
    x = T["root"]
    while x != None:
        y = x
        if z["key"] < x["key"]:
            x = x["left"]
        else:
            x=x["right"]
    z["parent"]=y
    if y == None:
        T["root"]=z
    else:
        if z["key"] < y["key"]:
            y["left"]=z
        else:
            y["right"]=z
```

Tisk stromu

```
def print_tree(x,i):
    if x != None:
        print("-"*(2*i),x["key"])
        i+=1
        print_tree(x["left"],i)
        print_tree(x["right"],i)
```

Vyhledávání ve stromu

Rekurzivní verze

```
def tree_search(x,k):
    if x == None or k == x["key"]:
        return x
    if k < x["key"]:
        return tree_search(x["left"], k)
    else:
        return tree_search(x["right"], k)
```

Iterativní verze

```
def tree_search_iterative(x,k):
    while x != None and k != x["key"]:
        if k < x["key"]:
            x=x["left"]
        else:
            x=x["right"]
    return x
```

Průchod stromem

Průchod stromem

```
def in_order_walk(x):
    if x != None:
        in_order_walk(x["left"])
        print(x["key"],end=" ")
        in_order_walk(x["right"])
```

Minimum a maximum

Hledání minimální hodnoty

```
def tree_min(x):
    while x["left"] != None:
        x=x["left"]
    return x
```

Hledání maximální hodnoty

```
def tree_max(x):
    while x["right"] != None:
        x=x["right"]
    return x
```

Odebírání vrcholu

Nahrazení podstromu jiným podstromem

```
def tree_swap(t,u,v):
    if t["root"] == u:
        t["root"] = v
        return
    y = u["parent"]
    if u==y["left"]:
        y["left"]=v
    if u==y["right"]:
        y["right"]=v
```

Nahrazení vrcholu jiným vrcholem

```
def node_swap(t,u,v):
    v["left"]=u["left"]
    v["right"]=u["right"]
    if t["root"]==u:
        t["root"]=v
        return
    y=u["parent"]
    if u==y["left"]:
        y["left"]=v
    if u==y["right"]:
        y["right"]=v
```

Vymazání uzlu

```
def tree_delete(t,z):
    if z["left"] == None:
        tree_swap(t,z,z["right"])
        return
    if z["right"] == None:
        tree_swap(t,z,z["left"])
        return
    y=tree_min(z["right"])
    tree_delete(t,y)
    node_swap(t,z,y)
```

Úkol

1. Použijte binární vyhledávací strom pro ukládání seznamu osob.
2. Klíčem bude jméno osoby.
3. Napište funkce:
   - `vloz_do_seznamu(s,jmeno)` – která vloží do seznamu osobu se jménem jmeno,
   - `tisk_seznamu(s)` – která vytiskne seznam osob v abecedním pořadí,
   - `odeber_ze_seznamu(s,jmeno)` – která odebere osobu se jménem jmeno ze seznamu.

- Příklad:
  - Příkazy:

```
seznam=["Pavel","Jitka","Alice","Karel","David"]
osoby={"root":None}
for i in seznam:
    vloz_do_seznamu(osoby,i)
tisk_seznamu(osoby)
odeber_ze_seznamu(osoby,"Alice")
tisk_seznamu(osoby)
```

  - Výpis:

```
Seznam osob: Alice David Jitka Karel Pavel
Seznam osob: David Jitka Karel Pavel
```