B stromy

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZADS - Základní algoritmy a datové struktury

B stromy

- Strom parametrizujeme přirozeným číslem $t > 2$.
- B strom je definován následujícími podmínkami:
  1. **Počet klíčů ve vrcholech**. V každém vrcholu stromu je maximálně $2t - 1$ klíčů a minimálně $t - 1$ klíčů. Výjimkou je kořen, kde může být méně než $t - 1$ klíčů.
  2. **Počet potomků.** Pokud vrchol obsahuje n klíčů, má 0 (pak je to list) nebo $n + 1$ potomků.
  3. **Hloubka listů.** Všechny listy jsou ve stejné hloubce.
  4. **Podmínka uspořádání.** Klíče jsou ve vrcholu uspořádány vzestupně. Označíme-li klíče $k_0 < k_1 < \ldots k_{n-1}$ a podstromy $\alpha_0, \alpha_1, \ldots, \alpha_n$, pak
     - pro $0 \leqslant i < n - 1$ jsou klíče v podstromu α_i menší než k_i,
     - pro $0 < i \leqslant n$ jsou klíče v podstromu α_i větší než k_{i-1}.

Uzel

```
def make_node():
    k=[None for x in range(2*T-1)]
    ch=[None for x in range(2*T)]
    return {"keys" : k, "children" : ch,"parent" : None, "n" : int,"leaf":None}

t = {"root":None}
```

Vložení uzlu do stromu

Funkce insert

```
def insert(tr, k):
    target = find_insertion_vertex(tr["root"], k)
    insert_with_subtrees(tr, target, k, None, None)

def find_insertion_vertex(r, k):
    if r==None:
        return r
    i = 0
    while i < r["n"] and k > r["keys"][i]:
        i += 1
    if r["leaf"]==True:
        return r
    else:
        return find_insertion_vertex(r["children"][i], k)
```

Funkce insert_with_subtrees

```
def insert_with_subtrees(tr, x, k, left, right):
    if x == None: # potrebujeme novy koren
        z = make_node()
        z["keys"][0]=k
        if left==None:
            z["leaf"] = True
        else:
            z["leaf"] = False
        z["n"] = 1
        z["parent"] = None
        set_as_child(z, 0, left)
        set_as_child(z, 1, right)
        tr["root"] = z
```

Funkce insert_with_subtrees

```
    elif x["n"] == 2*T - 1:
        p = x["parent"]
        m, l, r = split_node(x)
        if k<m:
            target=l
        else:
            target=r
        insert_with_subtrees(tr, target, k, left, right)
        insert_with_subtrees(tr, p, m, l, r)
```

Funkce insert_with_subtrees

```
    else:
        i = 0
        while i < x["n"] and k > x["keys"][i]:
            i += 1
        x["n"] += 1
        shift_right(x, i)
        x["keys"][i] = k
        set_as_child(x, i, left)
        set_as_child(x, i+1, right)
```

Vyhledávání

Vyhledávání

```
def search(x, k):
    i = 0
    while i < x["n"] and k > x["keys"][i]:
        i += 1
    if i < x["n"] and k == x["keys"][i]:
        return x, k
    elif x["leaf"]==True:
        return None, i
    else:
        return search(x["children"][i], k)
```

Úkol

1. Použijte B strom pro ukládání údajů o maximální teplotě v jednotlivých dnech roku.
2. Napište funkci:
   - `najdi_teplotu(s,den)` – která vrátí maximální teplotu pro daný den (pokud je k dispozici).

- Příklad:
  - Pro data vložená do stromu:

```
data = [{"den":"1.1.","teplota":1.1},
[{"den":"12.12.","teplota":2.3},
[{"den":"8.7.","teplota":20.1},
[{"den":"14.3.","teplota":4.3},
[{"den":"17.11.","teplota":8.1},
[{"den":"12.4.","teplota":6.2},
[{"den":"17.11.","teplota":3.8},
[{"den":"19.12.","teplota":1.8},
[{"den":"1.9.","teplota":12.3},
[{"den":"13.8.","teplota":19.6},
[{"den":"14.2.","teplota":2.2},
[{"den":"16.5.","teplota":10.7},
[{"den":"14.5.","teplota":10.5},
[{"den":"5.10.","teplota":6.2},
[{"den":"18.12.","teplota":0.4},
[{"den":"12.6.","teplota":13.1},]
```

  - Příkazy:

```
najdi_teplotu(t,"5.10.")
najdi_teplotu(t,"31.12.")
```

  - Vypíší:

```
Maximální teplota 5.10. byla 6.2°C.
Maximální teplota 31.12. nebyla nalezena.
```