NMIN102

Programování 2

B-stromy

RNDr. Michal Žemlička, Ph.D.

B-strom

B-strom (řádu m) je rozvětvený výškově vyvážený orientovaný strom splňující tato omezení:

1. kořen má nejméně dva potomky a nejvýše m potomků, není-li listem;
2. každý uzel kromě kořene a listů má nejméně $\lceil m/2 \rceil$ a nejvýše m potomků;
3. každý uzel má nejméně $\lceil m/2 \rceil - 1$ a nejvíce $m - 1$ datových záznamů; kořen má nejméně 1 datový záznam;
4. všechny větve jsou stejně dlouhé

B-strom – organizace dat v uzlu

$$[\ p_0, (k_1, p_1, d_1), (k_2, p_2, d_2), \ldots, (k_n, p_n, d_n), u\ ]$$

kde p_i jsou ukazatelé na potomky, k_i klíče, d_i data a u nevyužitý prostor

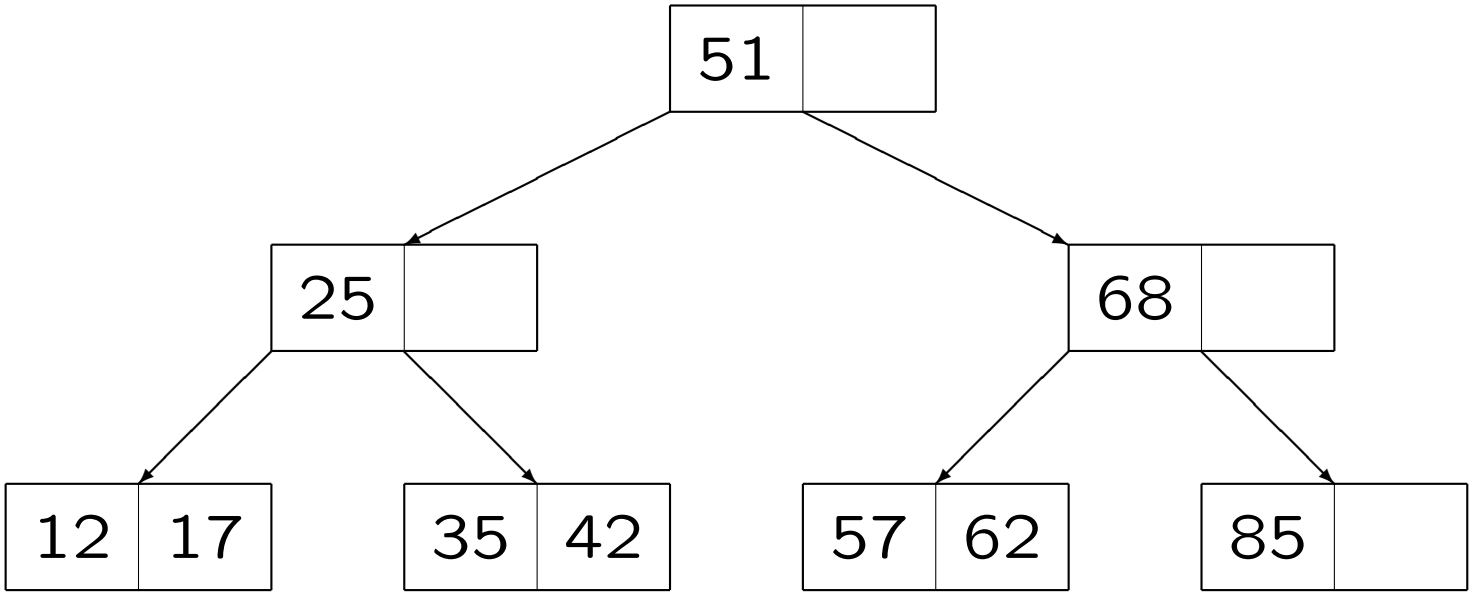
Záznamy (k_i, p_i, d_i) jsou uspořádány vzestupně dle klíčů.

B-strom – modifikace

Kořen podstromu může rozdělovat klíče v uzlech podstromu neostře (strom pak nazýváme *redundantní* – na původní verzi se budeme odkazovat jako na *neredundantní*), což dovolí i následující úpravy:

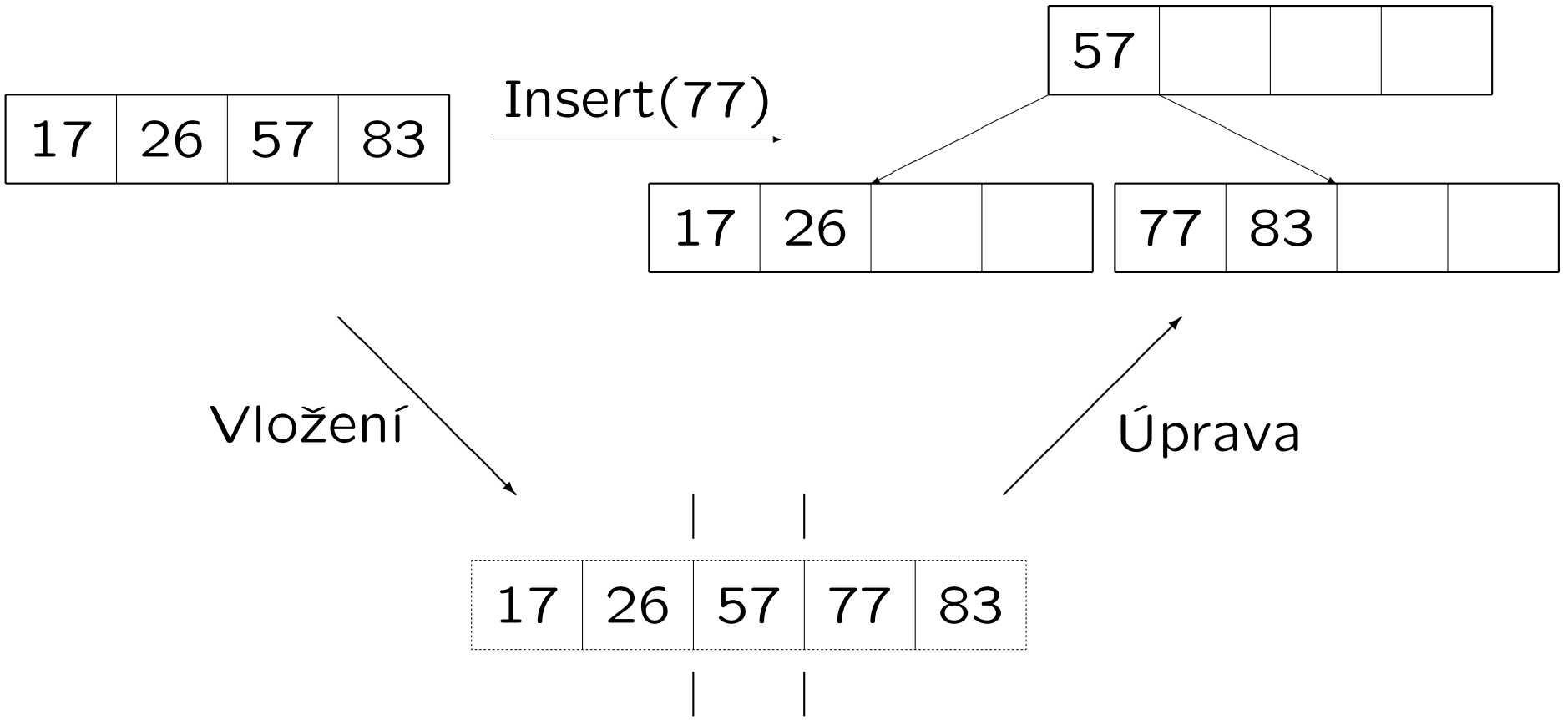
- Data mohou být uložena pouze v listech – v nelistových uzlech jsou pouze klíče.
- Data mohou být uložena odděleně – na poslední úrovni pak ukazatelé neukazují na další uzly stromu, ale na data.

B-strom – příklad



Neredundantní B-strom (2-3)-strom

B-strom – štěpení



B-strom – struktury

Pro neredundantní variantu s daty v uzlech můžeme použít tyto struktury:

```
TYPE
    rec = RECORD p:PgAdr; k:Key; d:Data END;
    page = RECORD
        m : Integer;          { zaplnění stránky }
        leaf : Boolean;       { je to list? }
        p0 : PgAdr;
        body : ARRAY [1..MaxM] OF rec;        END;
```

B-strom – struktury (2)

Pro rekurzivní variantu s daty v uzlech je třeba použít variantní strukturu:

```
TYPE
    irec = RECORD p:PgAdr; k:Key; END;
    drec = RECORD k:Key; d:Data END;
```

B-strom – struktury (3)

Také je třeba si pamatovat celou větev ve stromě (přinejmenším pro vkládání a vypouštění):

```
TYPE
    Branch = RECORD
        h : Integer;
        PgPtr: ARRAY [1..hmax] OF PgAdr;
        Ind: ARRAY [1..hmax] OF Integer;
        END;
```

B-strom – Access

```
PROCEDURE Access(root:PgAdr; k:Key; VAR s:Branch;
      VAR found:Boolean);
VAR pt:PgAdr; i:Integer;
BEGIN
      s.h:=0; pt:=root;
      IF root=NIL THEN BEGIN found:=false; Exit END;
      REPEAT
            s.h:=s.h+1; s.ptr[s.h]:=pt; GetPage(pt,pg);
            Search(pg,k,found,i,pt); s.ind[s.h]:=i
      UNTIL pg.leaf
END;
```

B-strom – Insert

```
PROCEDURE Insert(VAR root:PgAdr; r:rec; VAR ok:Boolean);
VAR         wr:rec; wk:Key; pt:PgAdr; p1,p2:page;
            found:Boolean; s:Branch; i:Integer;
BEGIN
      IF root=NIL THEN InitRoot(root,s)
      ELSE BEGIN Access(root,r.k,s,found);
            IF found THEN BEGIN ok:=false; Exit END
      END;
      ok:=true; wr:=r; GetPage(s.ptr[s.h],p1);
```

B-strom – Insert (2)

```
        FOR i:=s.h DOWNTO 1 DO
            WITH p1 DO BEGIN
                IF m < MaxM THEN
                    BEGIN MemIns(p1,wr,s.ind[i]); Exit END;
                Split(p1,wr,p2,pt);
                IF i=1 THEN
                    BEGIN NewRoot(p1,p2,root,pt); Exit END;
                wk:=body[m].k; GetPage(s.ptr[i-1],p1); wr.p:=pt;
                body[s.ind[i-1]].k:=wk; wr.k:=p2.ibody[p2.m].k
            END;
END;
```