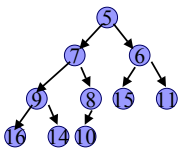
Haldy

ADT prioritní fronta

- množina M
- operace
  - **Přidej(M,x)** přidá prvek x do množiny M
  - **Odeber(M)** odeber z množiny M prvek, který „je na řadě“
- Zásobník (LIFO), Fronta (FIFO)
- Prioritní fronta:
  - **Přidej(M,x)** přidá prvek x do množiny M
  - **Min(M)** vrátí ukazatel na prvek v M s minimálním klíčem
  - **OdeberMin(M)** vrátí ukazatel na prvek s minimálním klíčem, a navíc odebere tento prvek z M

Binární halda

- implementace prioritní fronty ⇒ (binární) halda



5	7	6	9	8	15	11	16	14	10

Další operace nad haldou

- **SníženíKlíče(M,x,k)** sníží hodnotu klíče prvku x v M na k
- **Vymaž(M,x)** odstraní prvek x z M

- nová operace:
- **Sjednocení(M_1,M_2)** vytvoří novou množinu $M_1 \cup M_2$ množiny M_1 a M_2 jsou odstraněny

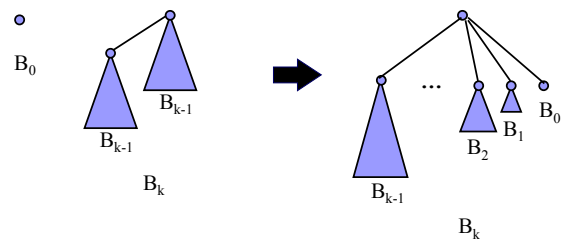
⇒ slučovatelná halda

Časová složitost jednotlivých operací

operace	binární halda (nejhorší případ)	binomická halda (nejhorší případ)	Fibonacciho halda (amortizovaná složitost)
Přidej	Θ(log n)	Θ(log n)	Θ(1)
Min	Θ(1)	Θ(log n)	Θ(1)
OdeberMin	Θ(log n)	Θ(log n)	Θ(log n)
Sjednocení	Θ(n)	Θ(log n)	Θ(1)
SníženíKlíče	Θ(log n)	Θ(log n)	Θ(1)
Vymaž	Θ(log n)	Θ(log n)	Θ(log n)

Binomické stromy

- Binomický strom B_k řádu k ($k \geq 0$)



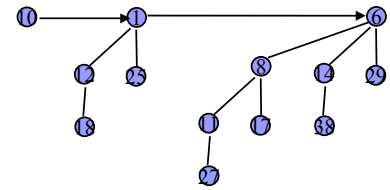
Vlastnosti binomického stromu

Věta: Binomický strom B_k

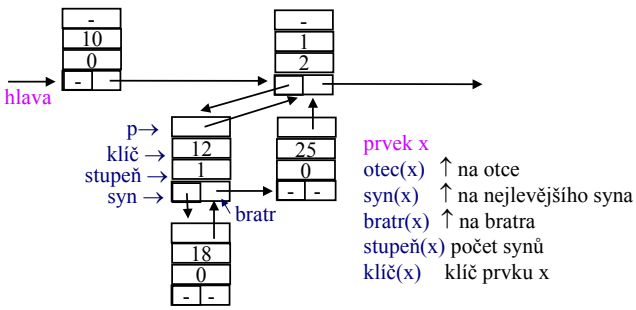
- obsahuje právě 2^k vrcholů ;
- má výšku k ;
- má právě $\binom{k}{i}$ vrcholů na i-té úrovni pro i = 0,1,…,k ;
- kořen má k synů, přičemž i-tý syn zprava (pro i=0,1,…,k-1) je kořenem podstromu B_i .

Binomická halda

- spojový seznam binomických stromů s ohodnocenými vrcholy splňující
  - je-li x otcem y, pak ohodnocení(x) ≤ ohodnocení(y)
  - pro každé k ≥ 0 se strom B_k vyskytuje v haldě nejvýše jedenkrát
  - stromy jsou uspořádány dle řádu (v rostoucí posloupnost)
- Příklad: M={1,6,8,10,11,12,14,17,18,25,27,29,38}



Implementace binomické haldy



Sjednocení

Sjednocení(H_1, H_2)

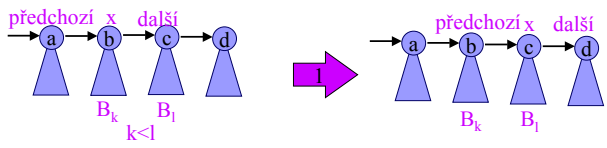
vytvoř prázdnou haldu H

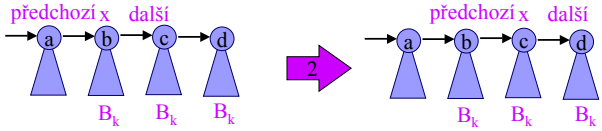
hlava(H) := HeapMerge(H_1,H_2)
(* slévání spojových seznamů H_1 a H_2 . Ve výsledném seznamu budou binomické stromy uspořádány v neklesající posloupnost dle řádu *)

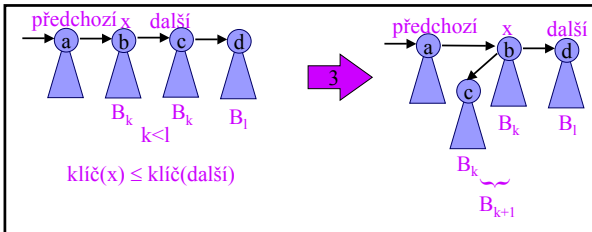
if hlava(H) = NIL **then return** H

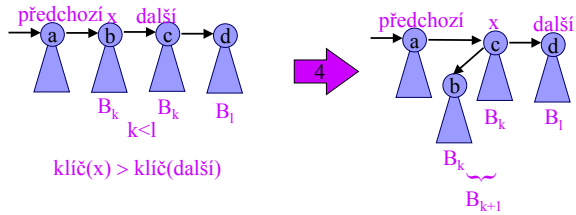
else předchozí := NIL; x := hlava(H) ; další := bratr(x) **fi**

while další ≠ NIL **do**









od.

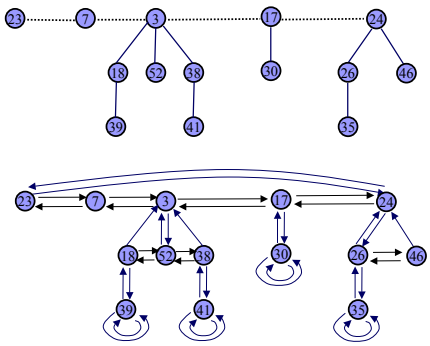
Operace Přidej a OdeberMin

- **Přidej(H,x)**
  vytvoř haldu H´ obsahující jediný prvek x
  H := Sjednocení(H,H´).
- **OdeberMin(H)**
  ve spojovém seznamu H najdi vrchol x s min. klíčem a vyjmi ho ze seznamu
  vytvoř prázdnou haldu H´
  ulož syny vrcholu x do seznamu v inverzním pořadí a hlavu seznamu ulož do hlava(H)
  H := Sjednocení(H,H´)
  return x .

Operace Min, SníženíKlíče a Vymaž

- **Min(H)**
  prohledej spojový seznam kořenů binomických stromů, na nějž ukazuje hlava(H)
  return prvek s minimálním klíčem.
- **SníženíKlíče(H,k,x)**
  if k > klíč(x) **then error**
  else klíč(x) := k ; y := x ; z := otec(y) ;
  while z ≠ NIL **and** klíč(y) < klíč(z)
  do y ↔z ; y := z ; z := otec(y) **od**
  fi .
- **Vymaž(H,x)**
  SníženíKlíče(H,x,-∞)
  OdeberMin(H) .

Fibonacciho halda



Implementace Fibonacciho haldy

- prvek x
- otec(x) ↑ na otce, syn(x) ↑ na syna
- levy(x), pravy(x) ↑ na bratra
- stupeň(x) počet synů
- znacka(x) Booleovská položka; určuje, zda x ztratil syna od chvíle, kdy se stal synem svého současného otce
- halda H
- min(H) ↑ na kořen stromu s min klíčem
- n(H) počet prvků v haldě
- Položme D(n) = max {stupeň(x) | x ∈ H , H je Fibonacciho halda o n prvcích}

Operace Přidej

VytvořFibHaldu(H)
min(H) := NIL; n(H) := 0.

Přidej(H,x)
stupeň(x) := 0;
otec(x):=syn(x):=NIL; levy(x):=pravy(x):=x;
znacka(x):=FALSE; uloz x do seznamu H ;
if min(H) = NIL or klíč(x) < klíč(min(H))
then min(H) := x **fi**
n(H)++ .

Sjednocení

Sjednocení(H_1,H_2)
VytvořFibHaldu(H);
min(H) := min(H_1) ;
zřetěz seznamy H_2 a H
if (min(H_1)=NIL) **or**
(min(H_2)<>NIL **and** min(H_2)<min(H_1))
then min(H):=min(H_2) **fi**
n(H):=n(H_1) + n(H_2) ;
dealokuj paměť pro H_1 a H_2
return H .

Odebrání minimálního prvku

OdeberMin(H)
z := min(H);
if *z*<>NIL **then forall** syny x vrcholu *z* **do** vlož *x* do seznamu H
otec(x) := NIL **od**
vyjmi *z* ze seznamu H
if *z*=pravy(*z*) **then** min(H):=NIL
else min(H):=pravy(*z*);
Konsolidace(H) **fi**
n(H)--
fi
return *z*.

Pomocná operace - spojení stromů

Link(H,y,x)
vyjmi *y* ze seznamu *H*
připoj *y* k vrcholu *x* jako nového syna; stupeň(*x*)++
znacka(*y*) := FALSE.

Konsolidace

Konsolidace(H)
for i:=0 **to** D(n(H)) **do** A[i]:=NIL **od**
forall vrchol *w* v seznamu H
do *x* := *w*; *d* := stupeň(*x*);
while A[*d*]<>NIL **do** *y* := A[*d*];
if klíč(*x*)>klíč(*y*) **then** *x* <-> *y* **fi**
Link(H,*y*,*x*);
A[*d*] := NIL; *d*++
od
A[*d*] := *x*
od

Konsolidace - dokončení

min(H) := NIL;
for i:=0 **to** D(n(H)) **do**
if A[i] <> NIL **then** přidej A[i] do seznamu
if min(H)=NIL **or**
klíč(A[i])<klíč(min(H))
then min(H):=A[i] **fi**
fi
od.

Snížení klíče

SníženíKlíče(H,x,k)
if k>klíč(*x*) **then error** "novy klic > aktualni klic" **fi**
klíč(*x*):=*k* ;
y := otec(x);
if *y*<>NIL **and** klíč(*x*) < klíč(*y*)
then Řez(*H,x,y*);
KaskádovýŘez (*H,y*) **fi**
if klíč(*x*) < klíč(min(*H*)) **then** min(*H*):=*x* **fi**.

Pomocné operace - řez & kaskádový řez

Řez(H,x,y)
odstraň *x* ze seznamu synů vrcholu *y*; stupeň(*y*)++;
přidej *x* do seznamu *H*
otec(*x*):=NIL; znacka(*x*):=FALSE.

KaskádovýŘez(H,y)
z:= otec(*y*);
if *z*<>NIL **then if** znacka(*y*)=FALSE **then** znacka(*y*):=TRUE
else Řez(*H,y,z*);
KaskádovýŘez(*H,z*)
fi
fi.

Literatura

- J.Vuillemin: *A data structure for manipulation priority queues.* Comm. ACM 21(1978), 309-315.
- M.L.Fredman, R.E.Tarjan: *Fibonacci heaps and their uses in improved network optimization algorithms.* J. ACM 34(1987), 596-615.