Prioritný rad

- každý prvok má prioritu
- prioritný rad - rad zoradený podľa priority
- nie FIFO, ale vyberie sa prvok s najvyššou prioritou
- príklady:
  - súbory na tlač čakajúce v rade
  - procesy čakajúce na preprocesor

Prioritný rad pomocou spájaného zoznamu

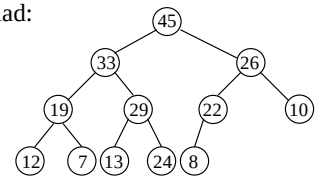
- Pridávanie prvkov na začiatok zoznamu O(1)
- Vymazávanie prvkov - nájdenie prvku s najväčšou prioritou, ten sa vymaže O(n)

Prioritný rad pomocou BVS

- Pridávanie prvkov - zaradenie do stromu podľa priority (priorita je kľúč)
- Vymazávanie prvkov - vymazanie prvku s najväčšou prioritou, t.j. najpravejší uzol
- Obe operácie - O(log n) - výhodnejšie ako pri spájanom zozname
- Nepotrebujeme všetky vlastnosti BVS len na to, aby sme našli prvok s najväčšou prioritou

Prioritný rad pomocou binárnej haldy

- Binárna halda je binárny strom, pre ktorý platí, že hodnota kľúča je väčšia alebo rovná hodnotám kľúčov jeho synov
- Príklad:



Binárna halda

- Binárna halda má menej striktné pravidlá na umiestnenie prvkov ako BVS
- Neplatí, že ľavý podstrom obsahuje prvky s nižšími hodnotami kľúčov ako pravý podstrom
- koreň stromu má však vždy najväčšiu hodnotu (>= ako ostatné uzly): vymazanie koreňa stromu

Binárna halda - implenentácia poľom

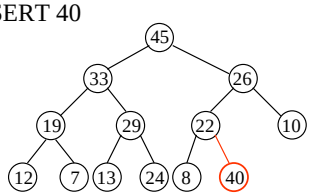
- Koreň stromu na 0-tej pozícii heap[0]
- Deti uzla na i-tej pozícii poľa, ak existujú:
  - left(i) = 2 * i
  - right(i) = 2 * i + 1
- heap[i..j], kde i>=0, je binárna halda práve vtedy, keď každý prvok nie je menší ako jeho deti.

Pridanie prvku do binárnej haldy

- Vytvorí sa nový vrchol na najnižšej úrovni
- Ak hodnota kľúča nového uzla <= hodnota predchodcu - koniec
- Ak je väčší, vymení sa nový uzol so svojím predchodcom
- Ak je hodnota nového uzla väčšia ako nový predchodca, vymení sa aj s ním, … až pokým nie je strom opäť haldou

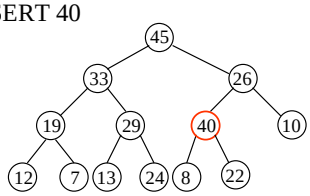
Pridanie prvku do binárnej haldy - príklad (1)

INSERT 40



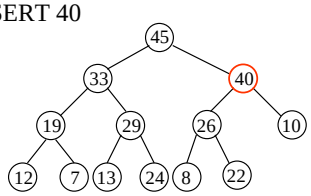
Pridanie prvku do binárnej haldy - príklad (2)

INSERT 40



Pridanie prvku do binárnej haldy - príklad (3)

INSERT 40



Pridanie prvku do binárnej haldy - implementácia

```
INSERT(heap, key)
  heap-size (heap) = heap-size(heap) + 1
  i = heap-size (heap)
  while i > 1 and heap[PARENT(i)] < key
      do heap[i] = heap[PARENT(i)]
          i = PARENT(i)
  heap[i] = key
```

Vymazanie najväčšieho prvku z binárnej haldy (1)

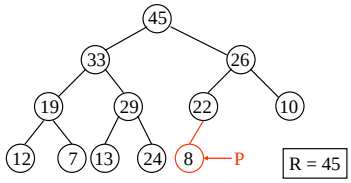
- Odstráni sa koreň haldy, hodnotu kľúča koreňa označíme R
- Odstráni sa najpravejší uzol na najnižšej úrovni (jeho hodnotu označme P)
- Pokúsime sa vyplniť hodnotu koreňa hodnotou P
- Ak hodnota P >= R, P sa zapíše do koreňa
- Inak presunieme potomka koreňa s väčšou hodnotou do koreňa,

Vymazanie najväčšieho prvku z binárnej haldy (2)

- R= hodnota presunutého uzla
- Vzniká voľné miesto, kam sa opäť pokúšame umiestniť P (ak hodnota P >= R)
- Takto pokračujeme až pokým nastane hodnota P >= R, kde R je hodnota posleného presunutého uzla, alebo posledný presunutý uzol je list - tam presunieme uzol P

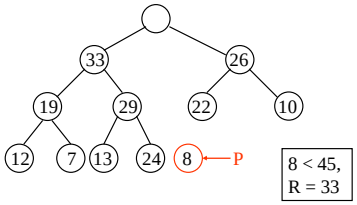
Vymazanie najväčšieho prvku z binárnej haldy - príklad (1)

EXTRACT-MAX



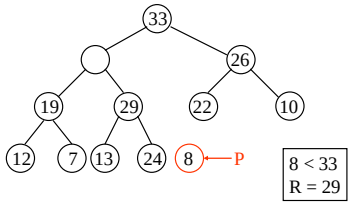
Vymazanie najväčšieho prvku z binárnej haldy - príklad (2)

EXTRACT-MAX



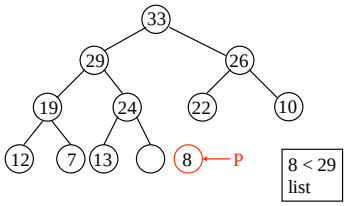
Vymazanie najväčšieho prvku z binárnej haldy - príklad (3)

EXTRACT-MAX



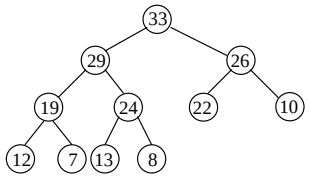
Vymazanie najväčšieho prvku z binárnej haldy - príklad (4)

EXTRACT-MAX



Vymazanie najväčšieho prvku z binárnej haldy - príklad (5)

EXTRACT-MAX



Vymazanie najväčšieho prvku z binárnej haldy - implementácia

```
EXTRACT-MAX(heap)
  if heap-size(heap) < 1
      then error
  max = heap[0]
  heap-size(heap) = heap-size(heap) -1
  HEAPIFY(heap, 0)
  return max
```

HEAPIFY - implementácia (1)

```
HEAPIFY(heap, i)
  l = left(i)
  r = right(i)
  if l <= heap-size(heap) and heap[l] > heap[i]
      then largest = l
      else largest = i
% pokračovanie
```

HEAPIFY - implementácia (2)

```
  if r <= heap-size(heap) and heap[r] > heap[i]
      then largest = r
  if largest <> i
      then exchange heap[i], heap[largest]
           HEAPIFY(heap, largest)
```

Vytvorenie haldy

```
BUILD-HEAP(heap)
  heap-size(heap) = length(heap)
  for i = ⌊length[heap] / 2⌋ downto 1
      do HEAPIFY(heap, i)
```

Cvičenie

- Implementujte prioritný rad (binárnu haldu) dynamicky.

  Pomôcka:

```
typedef struct prioritny_rad {
        int hodnota;
        struct prioritny_rad *lavy, *pravy;
} PRIORITNY_RAD;
```