

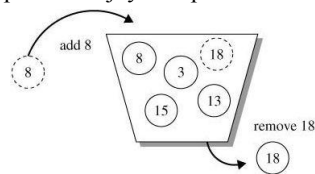
Prioritný front

binárna halda

1

Prioritný front

- Prioritný front je množina prvkov, ktorým je pridelená priorita – je ich možné porovnávať. Prvky je možné vkladať v akomkoľvek poradí s rôznou prioritou avšak pri výbere sa vyberá vždy prvok s najvyššou prioritou.



2

Prioritný front

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

3

Prioritný front - operácie

- Insert (S, x)
 - vloženie prvku x do množiny (prioritného frontu) S
- Maximum (S)
 - vrátenie prvku množiny S s najväčším kľúčom (prvku prioritného frontu s najväčšou prioritou)
- Extract-Max(S)
 - odstránenie a vrátenie prvku S s najväčším kľúčom

4

prioritný front - špecifikácia

- 1. create() vracia prázdny prioritný front
- 2. max(insert(v, create())) = v
- 3. extract-max(insert(v, create())) = create()
- 4. max(insert(v, insert(w, S))) =
 - if priorit(v) > priorit(max(insert(w, S)))
 - then v
 - else max(insert(w, S))
- 5. extract-max(insert(v, insert(w, S))) =
 - if priorit(v) > priorit(max(insert(w, S)))
 - then insert(w, S)
 - else insert(v, extract-max(insert(w, S)))

5

Prioritný front pomocou spájaného zoznamu

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

6

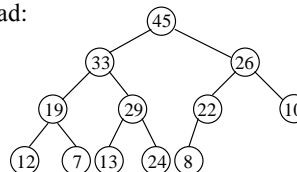
Prioritný front pomocou BVS

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

7

Prioritný front pomocou binárnej haldy

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



8

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
- platí haldová vlastnosť:
 - $A[\text{PARENT}(i)] \geq A[i]$ pre všetky vrcholy i okrem koreňa
- dôsledok: koreň stromu má však vždy najväčšiu hodnotu (\geq ako ostatné uzly)

9

Prioritný front pomocou binárnej haldy

- Insert (S, x)
 - Heap-Insert (A, x)
- Maximum (S)
 - Heap-Maximum (A)
- Extract-Max(S)
 - Heap-Extract-Max(A)

10

Binárna halda - implementácia vektorom

- Koreň stromu na 1. pozícii heap[1]
- Nasledovníky vrchola zapísaného na i -tej pozícii vektora, ak existujú:
 - left(i) = $2 * i$
 - right(i) = $2 * i + 1$
- heap[$i..j$], kde $i \geq 1$, je binárna halda práve vtedy, ak každý prvok nie je menší ako jeho nasledovníky.

11

počítanie indexov

- PARENT(i) // index predchodcu vrchola v i
 - return $\lfloor i / 2 \rfloor$ // floor($i/2$)
- LEFT(i) // index ľavého nasledovníka vrchola v i
 - return $2*i$
- RIGHT(i) // index pravého nasledovníka vrchola v i
 - return $2*i+1$

12

vrátenie prvku s najväčšou prioritou

- Heap-Maximum (A)
 - A[1]

$O(1)$

13

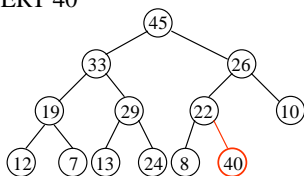
Pridanie prvku do binárnej haldy

- Vytvorí sa nový vrchol na najnižšej úrovni
- Ak hodnota kľúča nového vrchola \leq hodnota predchodcu - koniec
- Ak je väčšia, vymení sa nový vrchol so svojim predchodcom
- Ak je hodnota nového vrchola väčšia ako nový predchodca, vymení sa aj s ním, ... až pokiaľ nie je strom opäť haldou

14

Pridanie prvku do binárnej haldy - príklad (1)

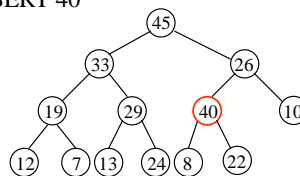
INSERT 40



15

Pridanie prvku do binárnej haldy - príklad (2)

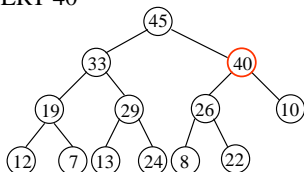
INSERT 40



16

Pridanie prvku do binárnej haldy - príklad (3)

INSERT 40



17

Pridanie prvku do binárnej haldy - implementácia

Heap-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

$O(\log n)$

18

Odstránenie najväčšieho prvku z binárnej haldy (1)

- Odstráni sa koreň haldy, hodnotu kľúča koreňa označíme R
- Odstráni sa najpravejší vrchol na najnižšej úrovni (jeho hodnotu označíme P)
- Pokúsime sa vyplniť hodnotu koreňa hodnotou P
- Ak hodnota $P \geq R$, P sa zapíše do koreňa
- Inak presunieme potomka koreňa s väčšou hodnotou do koreňa,

19

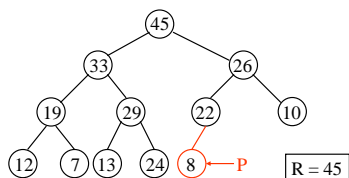
Odstránenie najväčšieho prvku z binárnej haldy (2)

- R = hodnota presunutého vrchola
- Vzniká voľné miesto, kam sa opäť pokúsime umiestniť P (ak hodnota $P \geq R$)
- Takto pokračujeme až pokým nastane hodnota $P \geq R$, kde R je hodnota posledného presunutého vrchola alebo posledný presunutý vrchol je list - tam presunieme P

20

Odstránenie najväčšieho prvku z binárnej haldy - príklad (1)

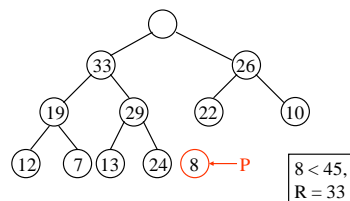
Heap-EXTRACT-MAX



21

Odstránenie najväčšieho prvku z binárnej haldy - príklad (2)

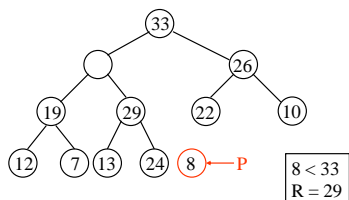
Heap-EXTRACT-MAX



22

Odstránenie najväčšieho prvku z binárnej haldy - príklad (3)

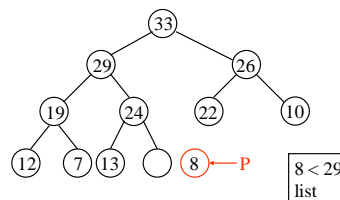
Heap-EXTRACT-MAX



23

Odstránenie najväčšieho prvku z binárnej haldy - príklad (4)

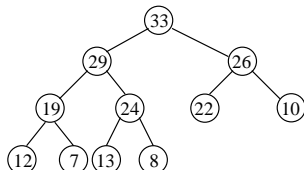
Heap-EXTRACT-MAX



24

Odstránenie najväčšieho prvku z binárnej haldy - príklad (5)

Heap-EXTRACT-MAX



25

Odstránenie najväčšieho prvku z binárnej haldy - implementácia

```

Heap-EXTRACT-MAX(heap)
  if heap-size(heap) < 1
    then error
  max = heap[1]
  heap[1] = heap[heap-size(heap)]
  heap-size(heap) = heap-size(heap) - 1
  HEAPIFY(heap, 1)
  return max
O(log n)

```

26

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

```

27

HEAPIFY - implementácia (2)

```

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

```

28

Vytvorenie haldy

z vektora heap[1..n], kde n=length(heap)
všetky prvky v podvektore heap[(⌊n/2⌋+1)..n] sú listy a teda
aj 1-prvkové haldy

```

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

```

29

Cvičenie

- Implementujte prioritný rad (binárnu haldu) dynamicky.

Pomôcka:

```

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

```

30