Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Jurij Mihelič, UniLj, FRI

Vrsta s prednostjo

- Vrsta s prednostjo (*priority queue*)
  - odvzemanje spredaj
    - odstranimo element z najmanjšo oz. največjo *prioriteto*
    - prioriteta je lahko tudi vrednost elementa oz. ključa
  - dodajanje s prioriteto
    - dodamo element v vrsto in pri tem podamo prioriteto

PriorityQueue

```
enqueue(p, x)
dequeue()

front()
```

oz. če je prioriteta del elementa

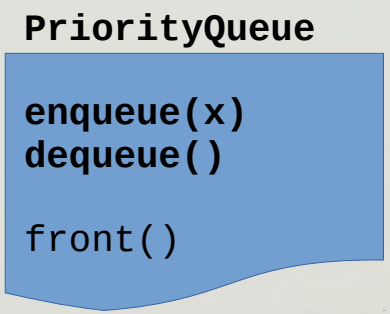
PriorityQueue

```
enqueue(x)
dequeue()

front()
```

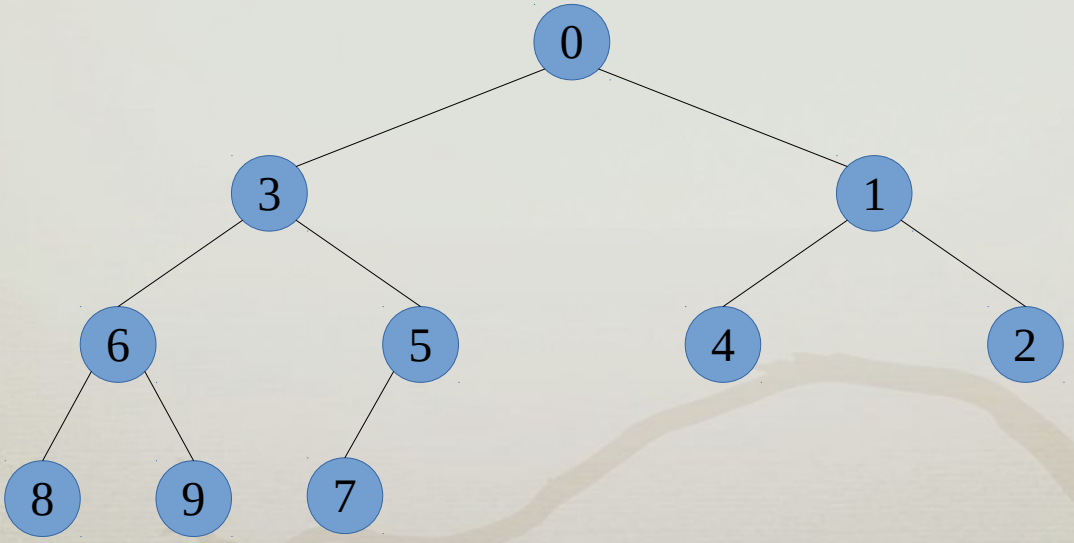
Vrsta s prednostjo

- Različne izvedbe
  - s poljem
  - z urejenim poljem
  - z urejenim povezanim seznamom
  - z uravnoteženim drevesom
  - kopica
  - itd.



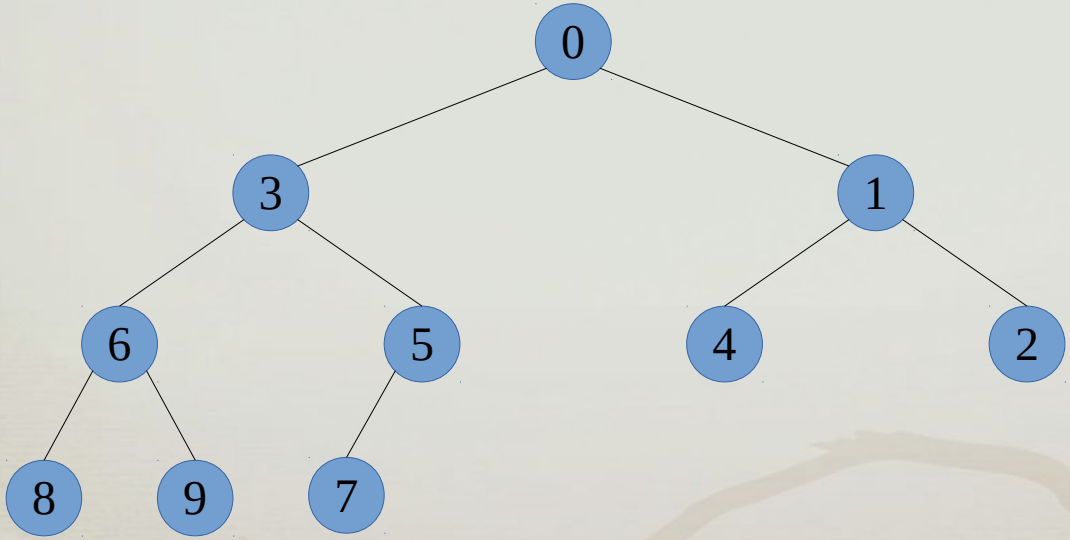
Kopica

- Definicija
  - celovito dvojiško drevo
    - učinkovita predstavitev v polju
  - delna urejenost vozlišč
    - urejenost med staršem in otroci



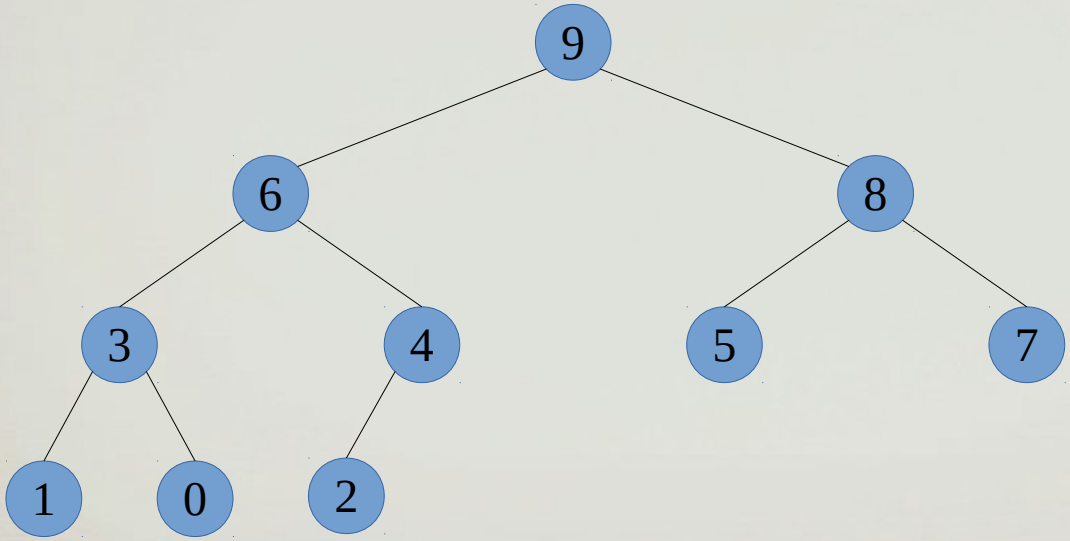
Kopica

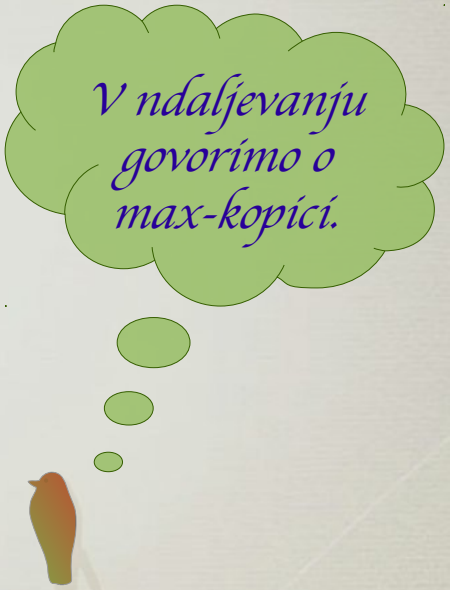
- Min-kopica
  - ključ starša ≤ ključi otrok
    - p.item ≤ min(p.left.item, p.right.item)
    - a[p] ≤ min(items[2*p+1], items[2*p+2])
  - v korenu je najmanjši element



Kopica

- Max-kopica
  - ključ starša ≥ ključi otrok
  - v korenu je največji element





Kopica

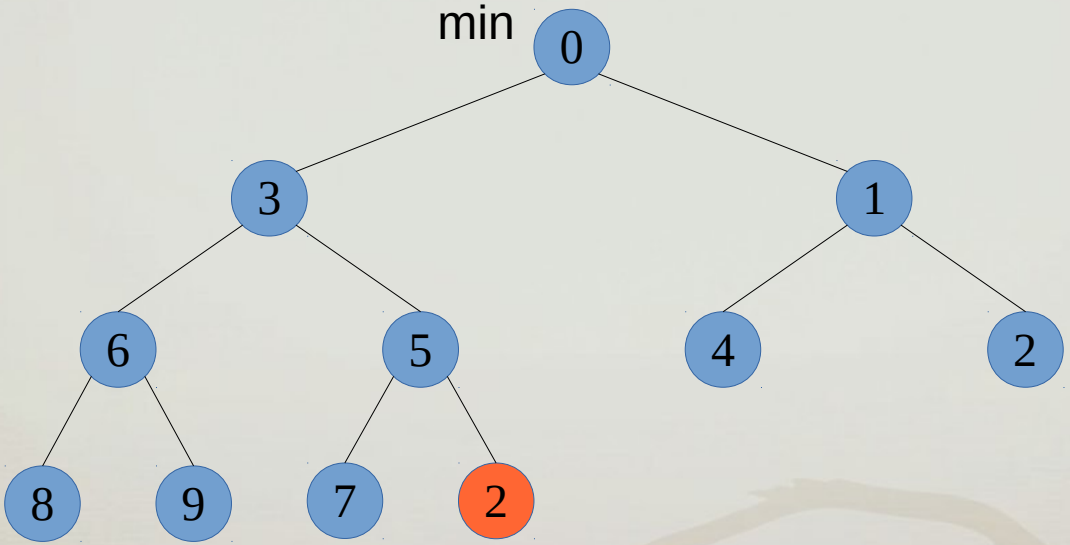
- Lastnosti
  - celovito drevo
  - višina kopice: $h = \lfloor \lg n \rfloor$
  - koren vedno vsebuje najmanjši / največji element
  - vsako poddrevo kopice je tudi kopica
  - učinkovita implicitna predstavitev celovitih dreves
    - otroka: $l = 2i+1$, $r = 2i+2$
    - starš: $p = \lfloor (i-1) / 2 \rfloor$
    - notranja vozlišča: prvih $\lfloor n/2 \rfloor$ elementov
    - listi: zadnjih $\lceil n/2 \rceil$ elementov

Kopica

- Dvigovanje elementa (*sift up*)
  - *skoraj kopica*, v kateri le en element
  - otrok kvari urejenost (glede na starša)
  - zamenjano ga z njegovim staršem
  - ponavljamo dokler gre

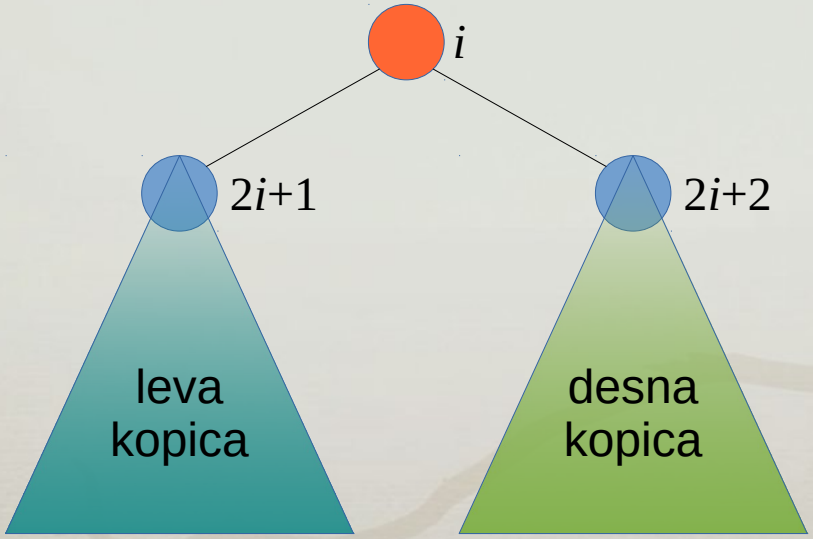
Kopica

- Vstavljanje elementa (*enqueue*)
  - dodamo element za konec kopice
  - velikost kopice povečamo za ena
  - ga dvignemo na ustrezno mesto



Kopica

- Ugrezanje elementa (*sift down*)
  - starš na indeksu i kvari urejenost (glede na otroke)
  - obe poddrevesi na 2i+1 in 2i+2 sta že kopici
  - zamenjamo ga z večjim (max-kopica) otrokom
    - zakaj ne smemo ugrezati v smeri manjšega?



Kopica

- Odvzemanje spredaj (*dequeue*)
  - vrnemo najmanjši / največji element
  - koren kopice
- Ideja algoritma
  - shrani koren in ga na koncu vrni
  - zamenjaj koren in zadnji element
  - zmanjšaj velikost kopice za ena
  - ugrezni koren

Kopica

- Gradnja kopice – 1. način (dvigovanje)
  - gradnja kopice iz zaporedja elementov
  - *vkopičenje (heapify*, *heapification*)
- Ideja algoritma
  - prazna kopica je kopica
  - zaporedoma vstavljamo elemente
- Online algoritem
  - ni nujno poznavanje celotnega zaporedja v naprej
  - elementi lahko prihajajo sproti

Kopica

- Gradnja kopice – 2. način (ugrezanje)
  - gradnja kopice iz zaporedja elementov
  - poznati moramo vse elemente v naprej
- Ideja algoritma
  - listi so kopice
  - ugrezanje notranjih vozlišč
    - notranja vozlišča: prvih $\lfloor n/2 \rfloor$ elementov
  - obiskovanje po višini

Ostale operacije

- Največji element
- Drugi največji element
- Iskanje elementa
- Povečevanje ključa elementa
- Zmanjševanje ključa elementa
- Spreminjanje ključa elementa
- Brisanje poljubnega elementa

Uporaba

- Razporejanje opravil
  - ko se opravilo zaključi, je naslednje na vrsti tisto z največjo prioriteto
- Urejanje s kopico
- Iskanje najkrajše poti v omrežju

Povzetek

operacija (max kopica)	zahtevnost
siftUp	$O(\lg n)$
siftDown	$O(\lg n)$
enqueue	$O(\lg n)$
dequeue	$O(\lg n)$
gradnja z dvigovanjem	$O(n \lg n)$
gradnja z spuščanjem	$\Theta(n)$
maksimum	$\Theta(1)$
drugi največji	$\Theta(1)$
iskanje elementa	$\Theta(n)$
večanje ključa elementa	$O(\lg n)$
zmanjševanje ključa elementa	$O(\lg n)$
odstranjevanje poljubnega elementa	$O(\lg n)$