CV 23.11.'10

PRIORITNÍ FRONTA

prvek ve frontě = e
prvek má klíč = k, podle kterého z fronty vybíráme

Q.build($\{e_1, e_2\}$) // inicializace fronty
Q.insert(e) // vložení prvku e do fronty
Q.min() // vrátí nejmenší prvek
Q.removeMin() // vrátí a vyřadí prvek

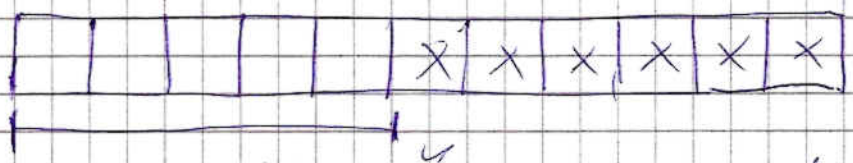
rozšíření - prvku e přiřadíme nějaký objekt ID, který prvek jednoznačně identifikuje

$e \begin{cases} \nearrow ID \text{ (= číslo, řetězec, ... cokoli)} \\ \searrow k \end{cases}$

Q.remove(ID) // vyřazení prvku dle ID, tj. bez ohledu na pořadí nebo prioritu

HW 6.4, str. 133

Implementace fronty - obyčejná fronta



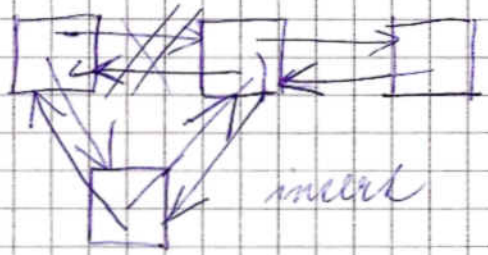
W => může vložit max. W prvků

n prvků vložím -- mohou být setř. nebo nesetř.

Časová náročnost

	nesetříděné	setříděné
Q. build ($\{e_1, e_2\}$)	$O(n)$	$n \log n + n$ $O(n \log n)$
Q. insert (e)	$O(1)$	$n \log n + n)$ $O(n \log n)$
Q. min ()	$O(n)$	$O(1)$
Q. removeMin ()	$n + n$ $O(n)$	$O(n)$

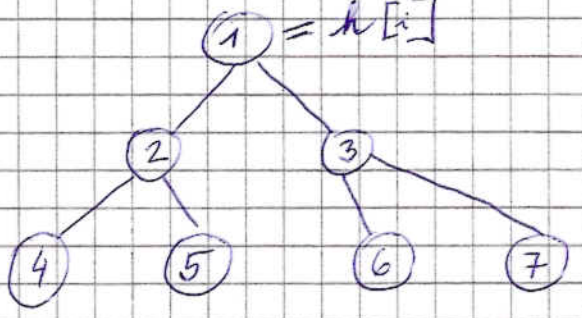
Implementace - spojový seznam

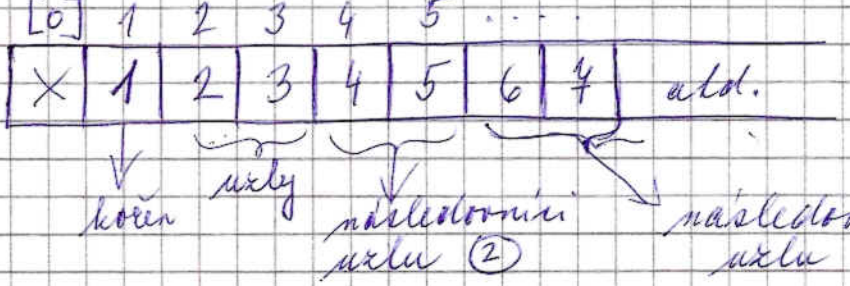


	nesetříděné	setříděné
Q. build ($\{e_1, e_2\}$)	$O(n)$	$O(n)$ $n \cdot \log n + n$
Q. insert (e)	$O(1)$	$\log n + 1$ $O(\log n)$
Q. min()	$O(n)$	$O(1)$
Q. removeMin()	$O(n)$	$O(1)$

Halda

= způsob, jak reprezentovat binární vyhledávací strom pomocí pole





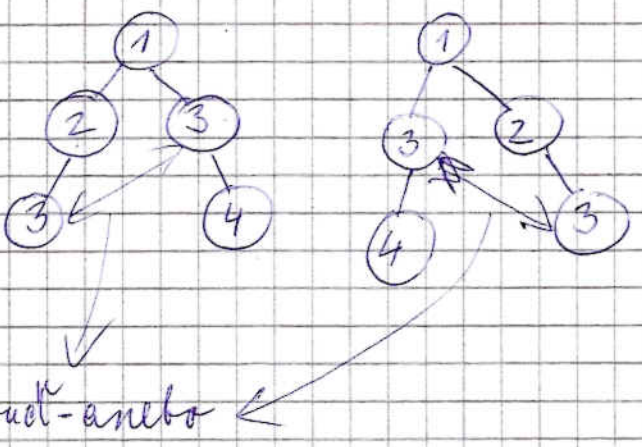
~~h[i] h[2i]~~ $i \leftarrow 2i$, $2i+1$

$h[i] \leq h[2i]$

$h[i] \geq h[2i+1]$

(PŘ) 1, 2, 3, 4 – možná uspořádání

1	2	3	4
	3	4	3
	4		



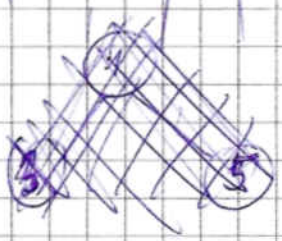
Jak seřídat haldu?

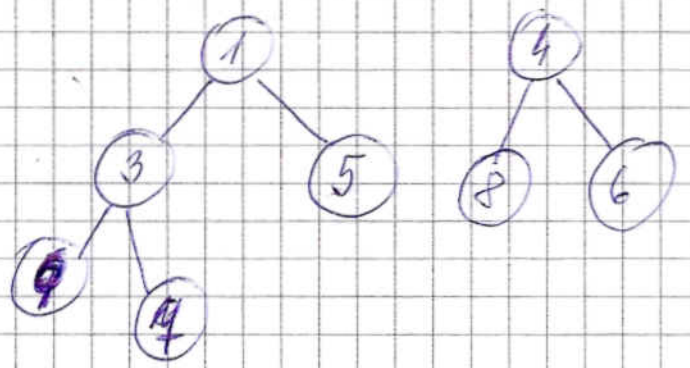
20	1	4	3	5	8	6	9	7

ind. že chceme zařadit do haldy

porovnáváme se potupně s hodnotou uzlů.

1	20							1. krok
	3	4	20					2. krok
			4	5	8	6	9	20 3. krok

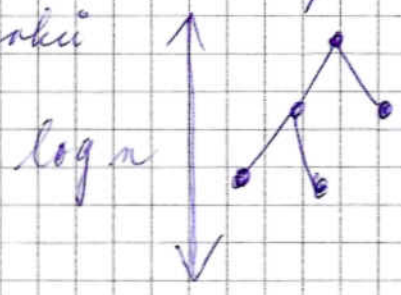




složitost výpočtu porovnání ~~na 1. krok~~

na 1. krok ~ 2 porovnání

počet poschí



⇒ složitost je $2 \log n$

lze snížit na $\log n + \log(\log n)$

~~for siftDown (i) // kořenový prvek do haldy~~

~~if (2i ≥ n) return~~

~~if (2i+1 ≥ n)~~

binární skládání 1 prvku = $\log(\log n)$

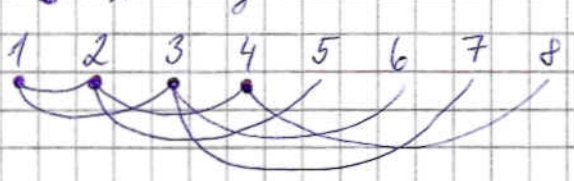
binární skládání n prvků

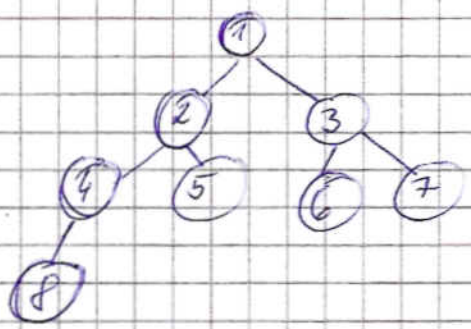
```
pro buildHeapBack() {
    for (i = ⌈n/2⌉ ... 1
        siftDown(i)          // pro siftDown()
                             // kácíme prvek
                             // do haldy
```

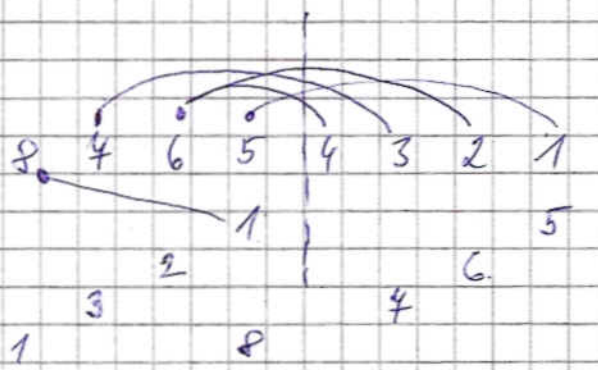
```
siftDown(i)
    if (2i ≥ n) { return }
    if (2i+1 ≥ n) ∨ h[2i] ≤ h[2i+1]
        m = i
    } else {
        m = 2i+1
    }
    if (h[m] ≤ h[i])
        swap(i, m)
        siftDown(m)
    }
```

---

1 - 8 = indexy







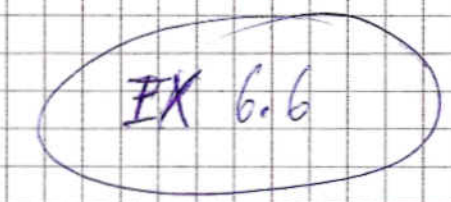
$\lfloor \frac{n}{2} \rfloor$ 2 log n

alg. - celkem 6 porovnání

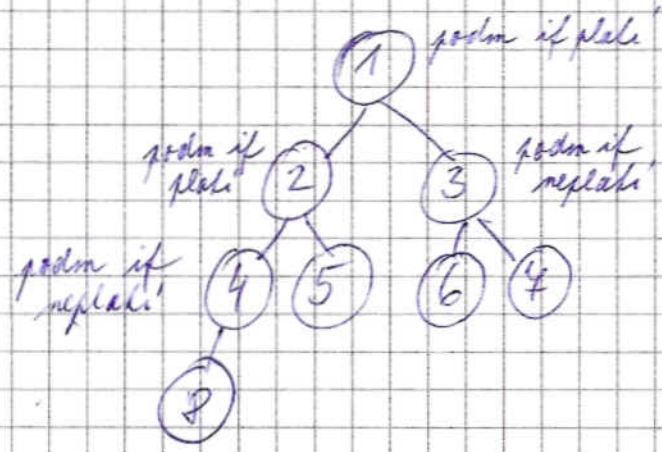
složitost $O(n \log n)$

1 4 2 5 8 3 6 7

```
buildHeapRead(i)                    // rekurzivní fce
  if (4i ≤ n){
    buildHeapRead(2i)
    buildHeapRead(2i+1)
  }
  siftDown(i)
}
```



1 2 3 4 5 6 7 8



všude, kde se volala met. HeapRead(), volá se siftDown()