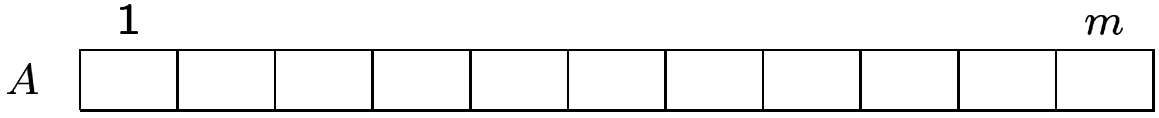
AULA 9

Filas com prioridades

CLRS 6.5

Filas com prioridade

Tipo abstrato de dados (= *abstract data type*)



Como organizar $A[1\,..\,m]$ de modo que as operações

consulte máximo
extraia máximo
aumenta valor de elemento
insira novo elemento

sejam eficientes?

Vetor sem ordem alguma? crescente? decrescente?

Implementações com max-heap:

HEAP-MAX (A, m)
1 devolva $A[1]$

Consome O(1)

HEAP-EXTRACT-MAX (A, m) $\triangleright$ $m \geq 1$

3 $max \leftarrow A[1]$

4 $A[1] \leftarrow A[m]$

5 $m \leftarrow m - 1$

6 MAX-HEAPIFY $(A, m, 1)$

7 devolva max

HEAP-INCREASE-KEY $(A, i, chave)$ $\triangleright$ $chave \geq A[i]$

3 $A[i] \leftarrow chave$

4 enquanto $i > 1$ e $A[\lfloor i/2 \rfloor] < A[i]$

5 faça $A[i] \leftrightarrow A[\lfloor i/2 \rfloor]$

6 $i \leftarrow \lfloor i/2 \rfloor$

MAX-HEAP-INSERT $(A, m, chave)$

1 $m \leftarrow m + 1$

2 $A[m] \leftarrow -\infty$

3 HEAP-INCREASE-KEY $(A, m, chave)$

Todos consomem $O(\lg m)$

TAREFA (AULA 9)

Exercício 9.A [CLRS 6.5-7]
Escreva uma implementação eficiente da operação MAX-HEAP-DELETE(A, m, i). Ela deve remover o nó i do max-heap $A[1\,..\,m]$ e armazenar os elementos restantes, em forma de max-heap, no vetor $A[1\,..\,m{-}1]$.

Exercício 9.B [CLRS 6-1, p.142]
O algoritmo abaixo faz a mesma coisa que o BUILD-HEAP?

```
B-H (A, n)
1  para m ← 2 até n faça
2       i ← m
3       enquanto i > 1 e A[⌊i/2⌋] < A[i] faça
4            A[⌊i/2⌋] ↔ A[i]     ▷ troca
5            i ← ⌊i/2⌋
```

Qual o invariante no início de cada iteração do bloco de linhas 2–5?
Qual o consumo de tempo do algoritmo?

Exercício 9.C [CLRS 6.5-5]
Prove que HEAP-INCREASE-KEY está correto. Use o seguinte invariante: no início de cada iteração, $A[1\,..\,m]$ é um max-heap exceto talvez pela violação da relação $A[\lfloor i/2 \rfloor] \geq A[i]$.