Algoritmos e Estrutura de Dados II
(ESTCMP011)
2do PERIODO 2018

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Filas de prioridades.

Prof. Luis Cuevas Rodríguez, PhD
E-mail: lcuevasrodriguez@gmail.com / lrodriguez@uea.edu.br
Celular: 9298154648

EST
Escola Superior de Tecnologia da UEA

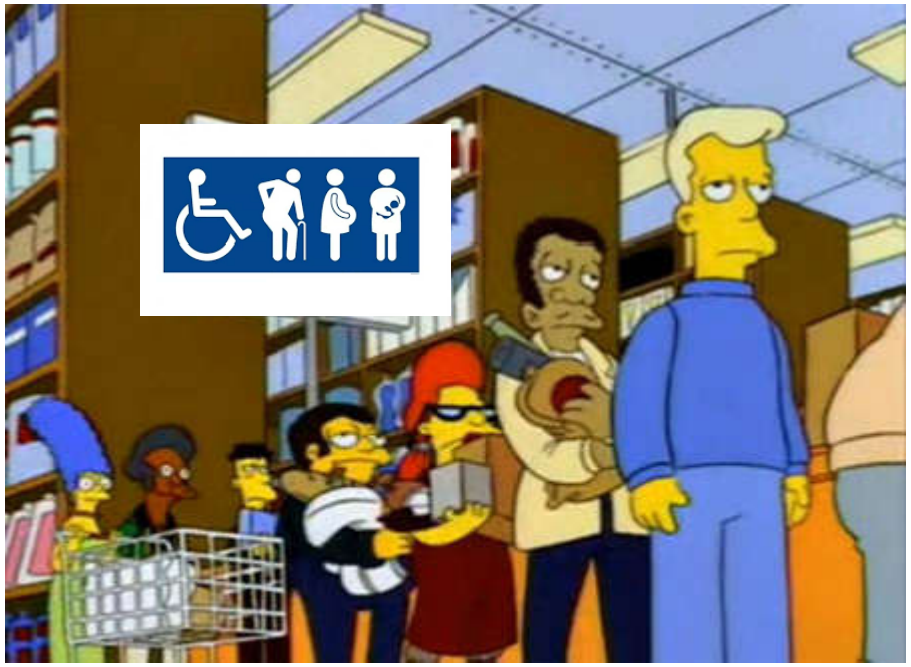
NUCOMP
NÚCLEO DE COMPUTAÇÃO

Conteúdo

- Fila de prioridade
  - Definição
  - Implementação
  - Complexidade

Fila de prioridade

- Filas (FIFO): primeiro em entrar primeiro em sair
- **Fila de prioridade:** estrutura de dado que mantém uma coleção de elementos, cada um com uma **prioridade associada**.



Fila de prioridade- Exemplos

- Fila de pacientes esperando transplante
- A fila de prioridade de processos aguardando o processador para execução num Sistemas Operacionais (os processos mais prioritários são executados antes dos outros)

Fila de prioridade - Implementação

- Inserir um elemento novo na fila de prioridade.
- Remover o elemento de maior prioridade da fila de prioridade.

Fila de prioridade - Operações

- Mantendo a ordem total.
- Forma aleatória sem nenhuma ordenação.
- Com ordem parcial usando um “heap”

Mantendo a ordem total

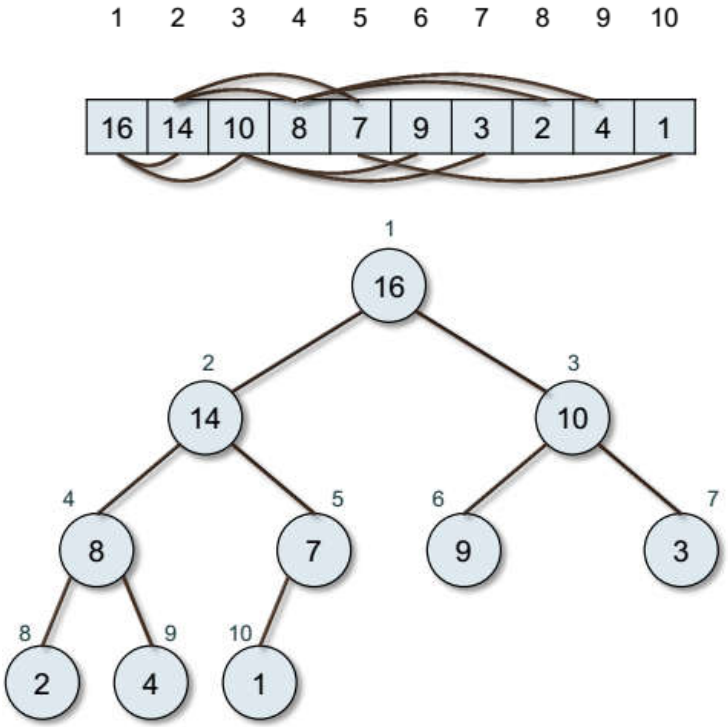
- Manter uma **lista linear ligada** ou encadeada em que os elementos estão ordenados por prioridades decrescentes.
  - Remover um elemento da fila de prioridade: remover o primeiro elemento (tempo constante)
  - Inserir um novo elemento: o pior caso é O(n), onde n é o número de elementos na fila de prioridade

Forma aleatória sem nenhuma ordenação

- Armazenar de forma aleatória os elementos em uma **lista linear seqüencial**, sem nenhuma ordem.
  - Para inserir um novo elemento: inserir em qualquer lugar, por exemplo no final da lista: Tempo O(1).
  - Para remover um elemento da fila de prioridade: é preciso percorrer a lista para obter o elemento com a maior prioridade. Remove-se este elemento, colocando no seu lugar um outro qualquer, por exemplo aquele no final da lista. Tempo O(n), onde n é o número de elementos na fila.

Com ordem parcial usando um “heap”

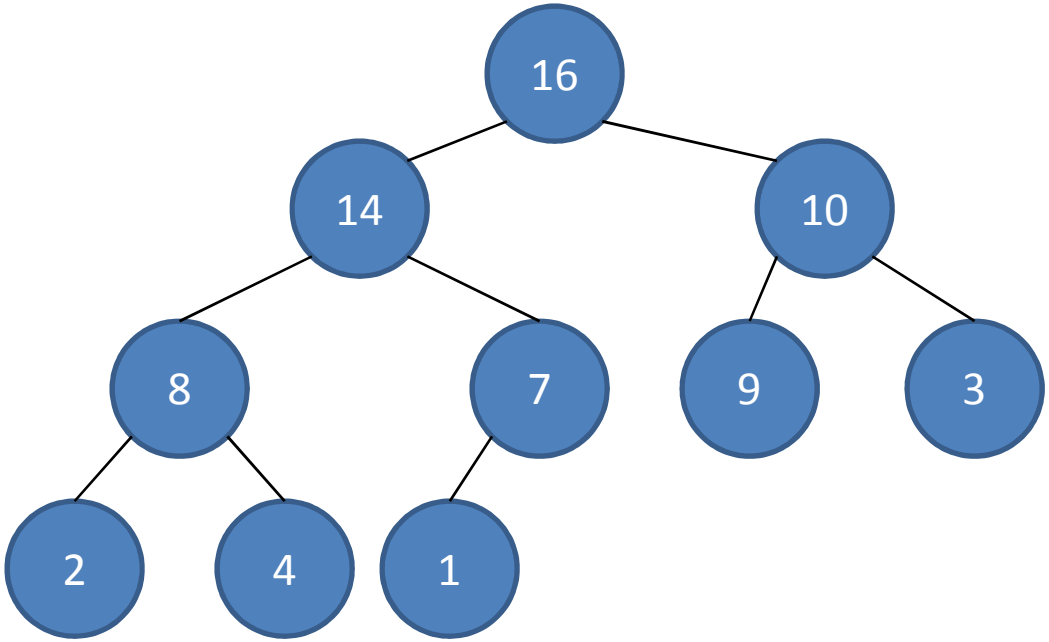
- Usar um **Heap máximo** de prioridade (árvore binária, cada nó tem uma prioridade maior ou igual à prioridade de seus filhos)



Com ordem parcial usando um “heap”

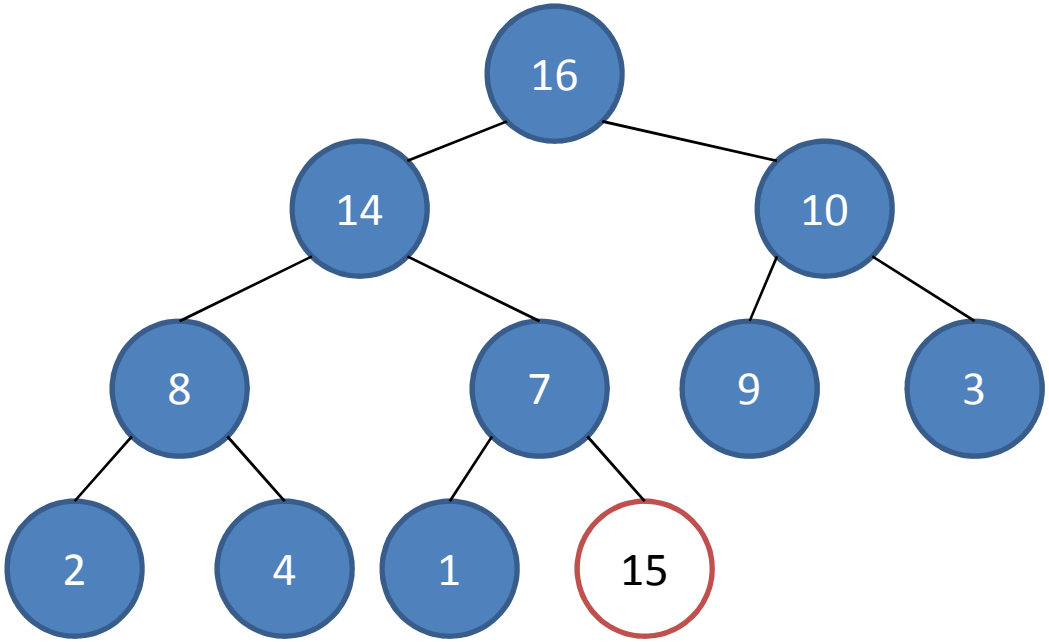
- **Inserir um novo elemento** com prioridade x:
  - cria-se um novo elemento no fim do vetor para receber x (isso pode perturbar a propriedade do “heap”)
  - Para consertar isso: se x for maior que seu pai, então os dois trocam de lugar. Essa operação é repetida até que x encontre o seu lugar correto na árvore. O exemplo insere 16 no heap.

Com ordem parcial usando um “heap”



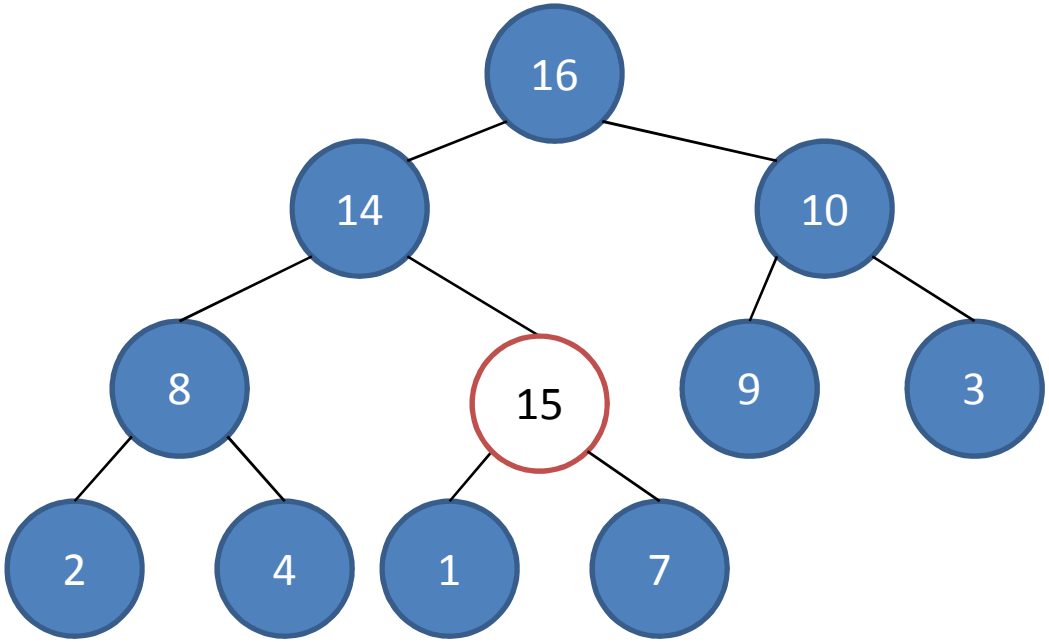
16	14	10	8	7	6	3	2	4	1	**15**			

Com ordem parcial usando um “heap”



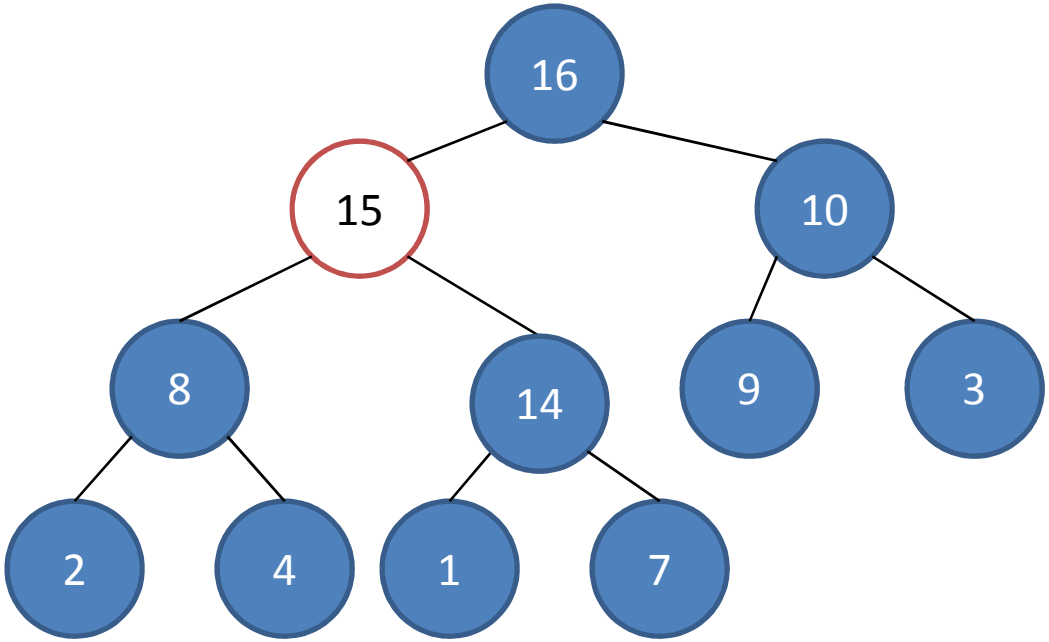
16	14	10	8	7	6	3	2	4	1	**15**			

Com ordem parcial usando um “heap”



16	14	10	8	**15**	6	3	2	4	1	7			

Com ordem parcial usando um “heap”



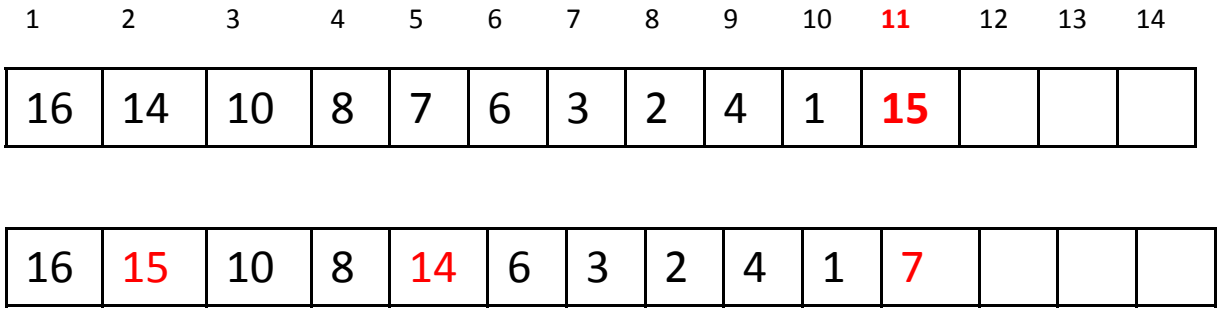
ult: ultimo elemento do vetor A

16	15	10	8	14	6	3	2	4	1	7			

Com ordem parcial usando um “heap”

ult: ultimo elemento do vetor A; x: prioridade do novo elemento

```
ult ← ult +1
k ← ult
while (|k/2|) and x > A[|k/2|]
      A[k]←A[|k/2|]
      k ← |k/2|
end while
A[k]←x
```



O(log n)

Com ordem parcial usando um “heap”

- Remover o elemento com maior prioridade:
  - elemento de maior prioridade é A[1], remover A[1]
  - Por em A[1] o elemento da última posição.
  - manter o heap desde A[1], função **Heapify** (O(log n)
- Complexidade de remover O(log n)

Formas de implementar	Inserir	Remover
Mantendo a ordem total	O(n)	O(1)
Forma aleatória sem nenhuma ordenação	O(1)	O(n)
Com ordem parcial usando um “heap”	O(log n)	O(log n)

Outras operações

Formas de implementar	Seleção	Alterar
Mantendo a ordem total	O(1)	O(n)
Forma aleatória sem nenhuma ordenação	O(n)	O(n)
Com ordem parcial usando um “heap”	O(1)	O(log n)