Estruturas de Dados

Heapsort

Heapsort

- A estrutura de dados **heap** é um objeto arranjo que pode ser visto como uma árvore binária praticamente completa. Cada nó da árvore corresponde a um elemento do arranjo que armazena o valor no nó.

- A árvore está completamente preenchida em todos os níveis, exceto talvez no nível mais baixo, que é preenchido a partir da esquerda até certo ponto.

Heapsort

- Um arranjo A que representa um heap é um objeto com dois atributos: **comprimento[A]**, que é o número de elementos do arranjo e **tamanho-do-heap[A]**, o número de elementos no heap armazenado dentro do arranjo A.

- Ou seja, embora A[1..comprimento[A]] possa conter números válidos, nenhum elemento além de A[tamanho-do-heap[A]] onde tamanho-do-heap[A] ≤ comprimento[A], é um elemento do heap.

Heapsort

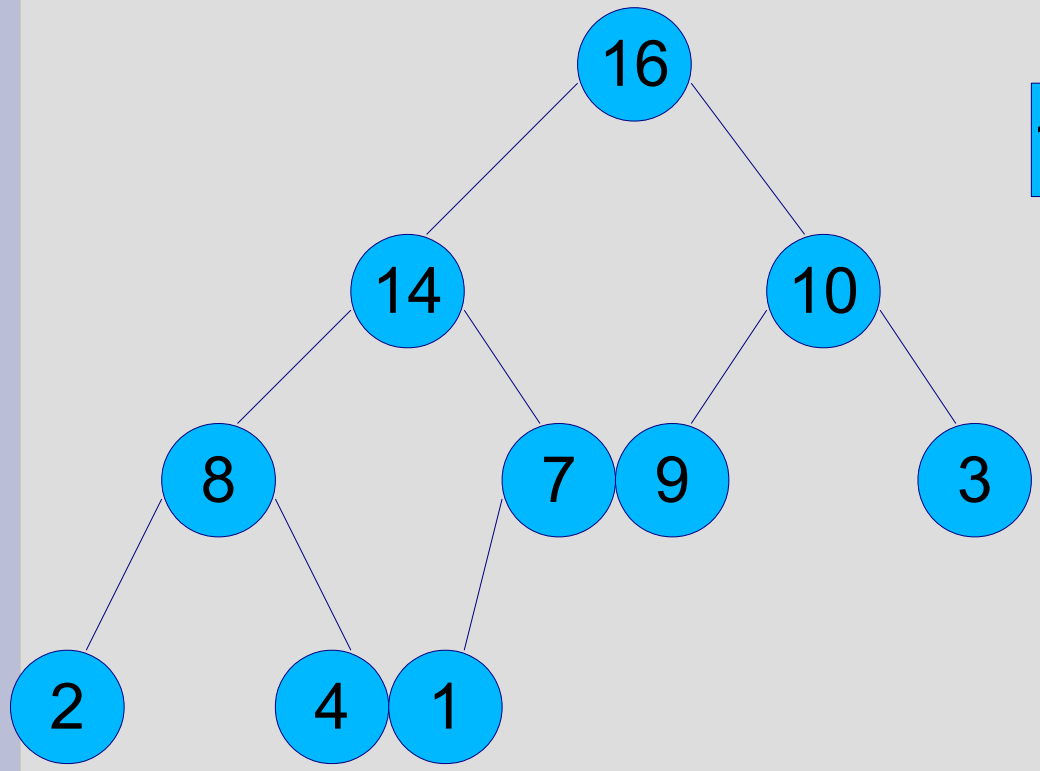
- A raiz da árvore é A[1] e, dado o índice i de um nó, os índices de seu pai PARENT(i), do filho da esquerda LEFT(i) e do filho da direita RIGHT(i) podem ser calculados de modo simples:

- Parent(i)
  return i

- Left(i)
  return 2i

  Right(i)
  return 2i + 1

Heapsort



16	14	10	8	7	9	3	2	4	1

Um heap máximo visto como uma árvore binária e um arranjo.

Heapsort

- Na maioria dos computadores, o procedimento LEFT pode calcular 2i em uma única instrução, simplesmente deslocando a representação binária de i uma posição de bit para a esquerda.

- De modo semelhante, o procedimento right pode calcular rapidamente 2i+1 deslocando a representação binária de i uma posição de bit para a esquerda e inserindo 1 como valor de bit de baixa ordem.

Heapsort

- O procedimento PARENT pode calcular $i/2$ deslocando i uma posição de bit para a direita.

- Existem dois tipos de heaps binários: heaps máximos e heaps mínimos. Em ambos os tipos, os valores nos nós satisfazem a uma propriedade de heap, cujos detalhes específicos dependem do tipo de heap.

Heapsort

- Em um heap máximo, a propriedade de heap máximo é que para todo nó i diferente da raiz

  $A[PARENT(i)] \geq A[i]$ isto é, o valor de um nó é no máximo o valor de seu pai.

- Um heap mínimo é organizado de modo oposto; a propriedade de heap mínimo é que, para todo nó i diferente da raiz
  $A[PARENT(i)] \leq A[i]$ o menor elemento em um heap mínimo está na raiz.

Heapsort

- Visualizando um heap como uma árvore, definimos a **altura** de um nó em um heap como o número de arestas no caminho descendente simples mais longo desde o nó até uma folha, e definimos a altura do heap como a altura de sua raiz.

- Tendo em vista que um heap de n elementos é baseado em uma árvore binária completa, sua altura é **O(lg n)**

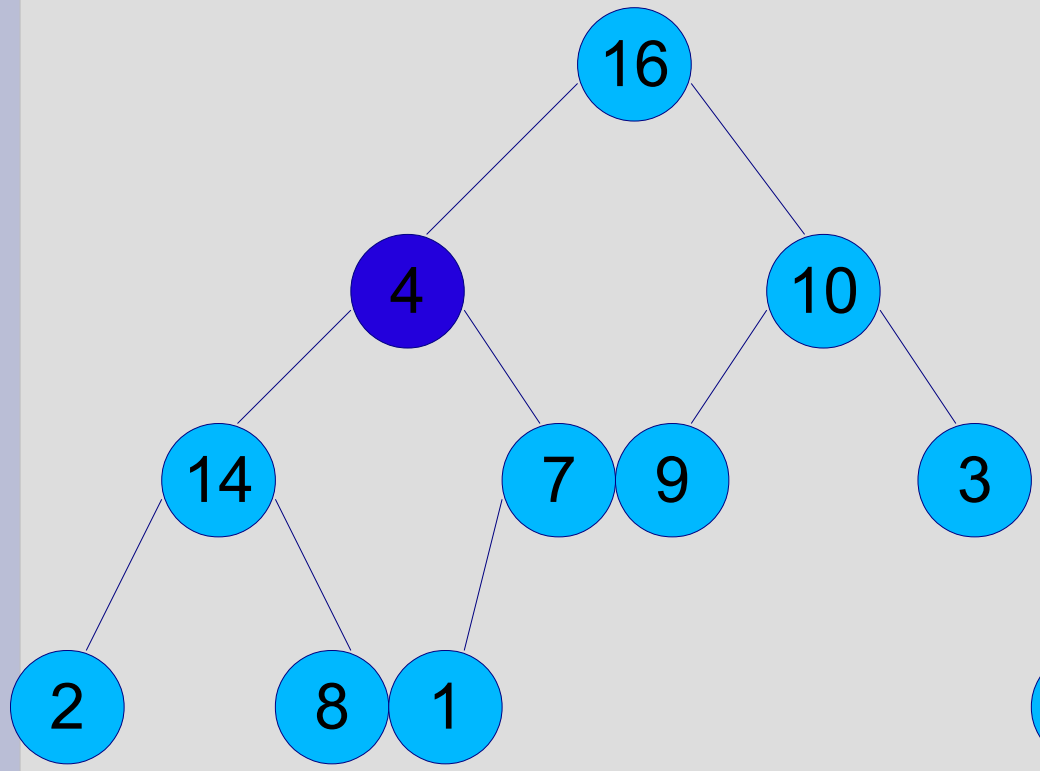
MAX-HEAPIFY

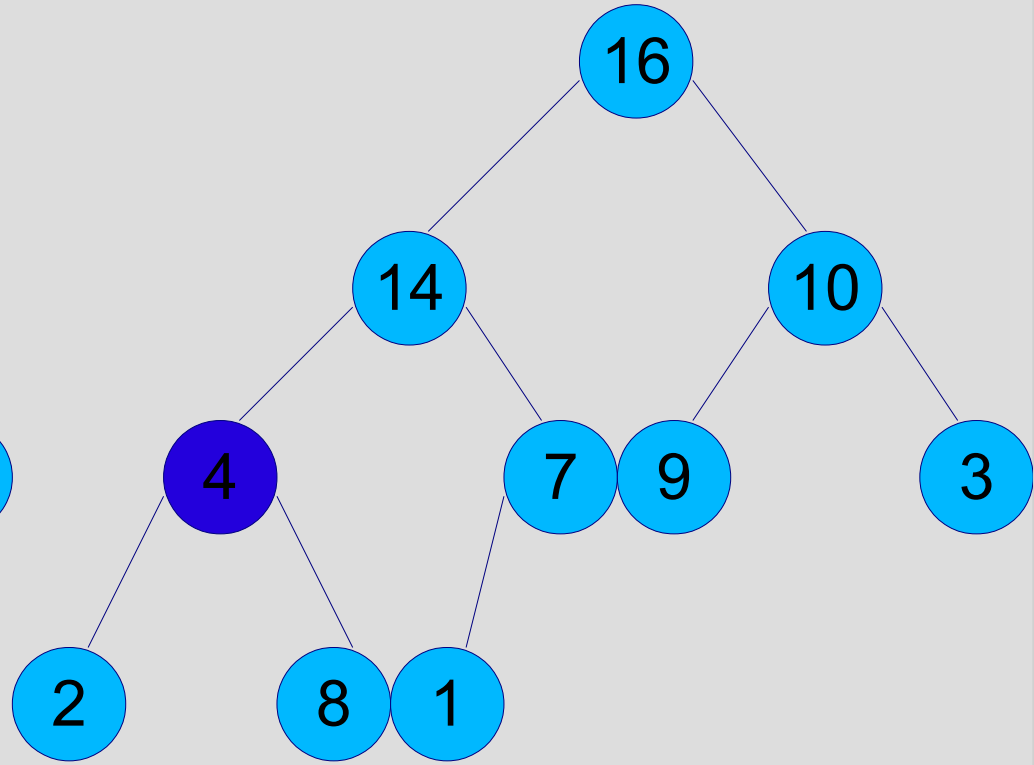
```
Max-Heapify(A,i)
    l ← Left(i)
    R ← Right(i)
    if l≤tamanho-do-heap[A] e A[l]>A[i]
       then maior←l
       else maior←i
    if r≤tamanho-do-heap[A] e A[r}>A[maior]
       then maior←r
    if maior≠i
       then trocar A[i]↔A[maior]
            MAX-HEAPIFY(A,maior)
```

Manutenção da propriedade de heap

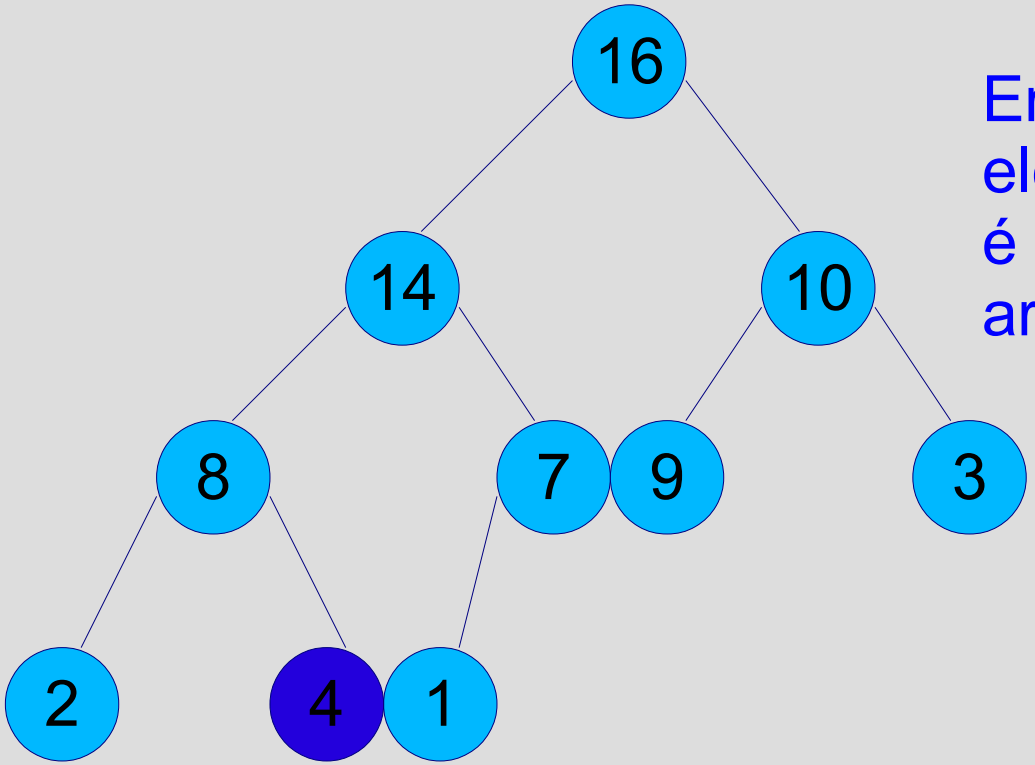
- MAX-HEAPIFY é uma sub-rotina importante para a manipulação de heaps máximos. Suas entradas são um arranjo A e um índice i para o arranjo. Quando MAX-HEAPiFY é chamado, supomos que as árvores binárias com raízes em LEFT(i) e RIGHT(i) são heaps máximos, mas que A[i] pode ser menor que seus filhos, violando assim a propriedade de heap máximo. A função de MAX-HEAPIFY é deixar que o valor em A[i] flutue para baixo no heap máximo, de tal forma que a subárvore com raiz no índice i se torne um heap máximo.

MAX-HEAPIFY





MAX-HEAPIFY



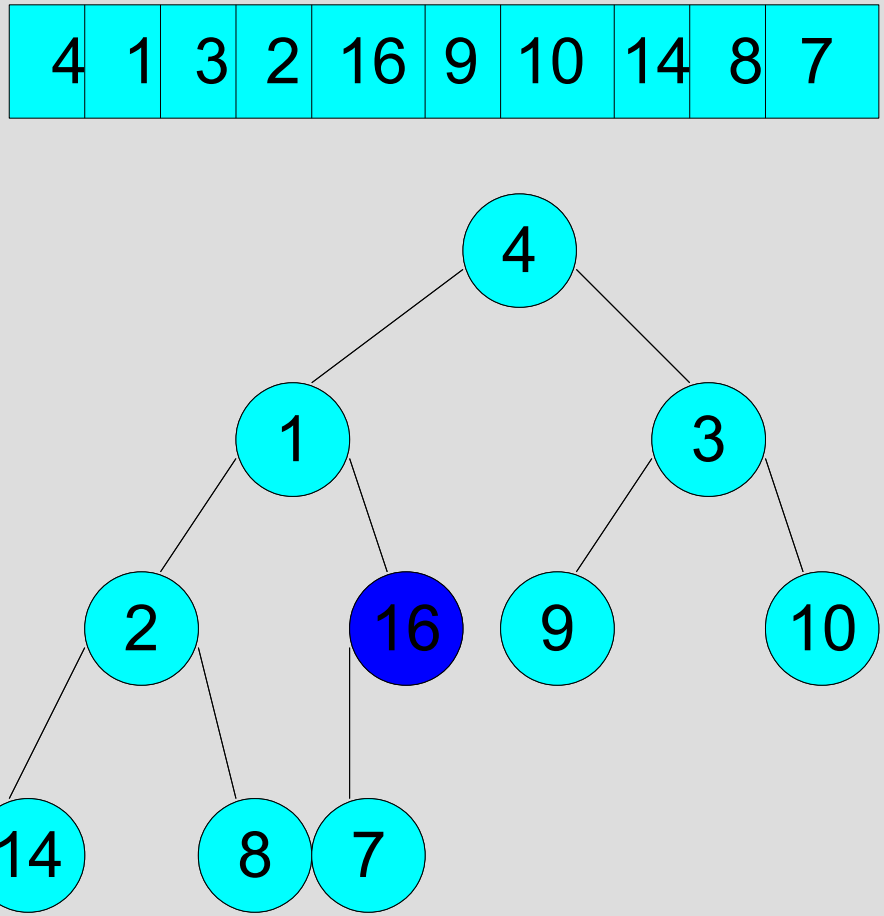
Em cada passo, o maior entre os elementos A[i], A[LEFT], A[RIGHT] é determinado e seu índice é armazenado em maior.

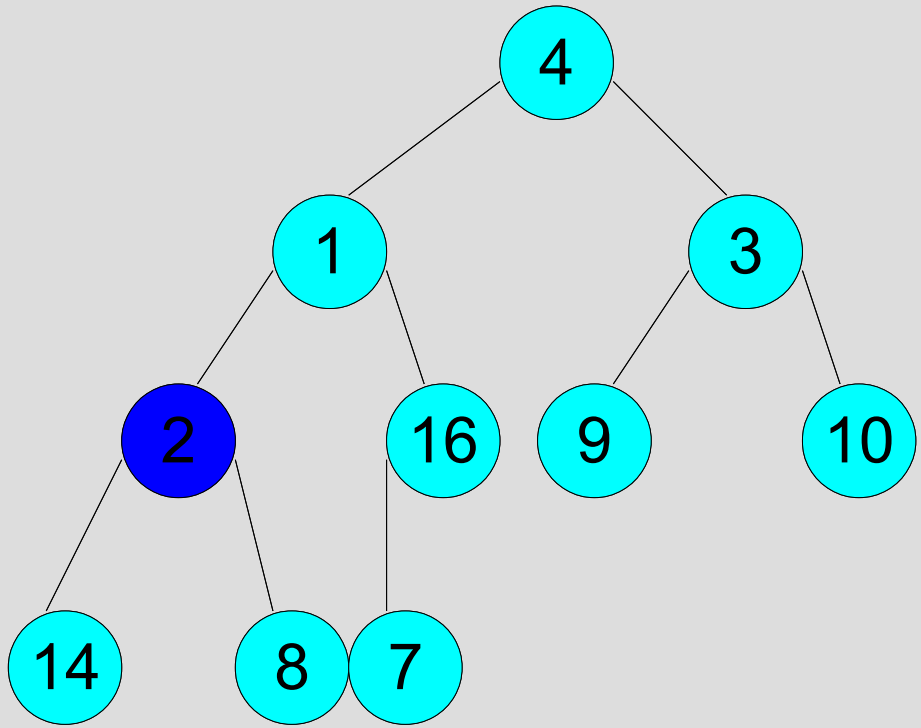
Construção de um Heap

- Podemos usar o procedimento MAX-HEAPIFY de baixo para cima, a fim de converter um arranjo A[1..n], onde *n*= *comprimento[A]* em um heap máximo.

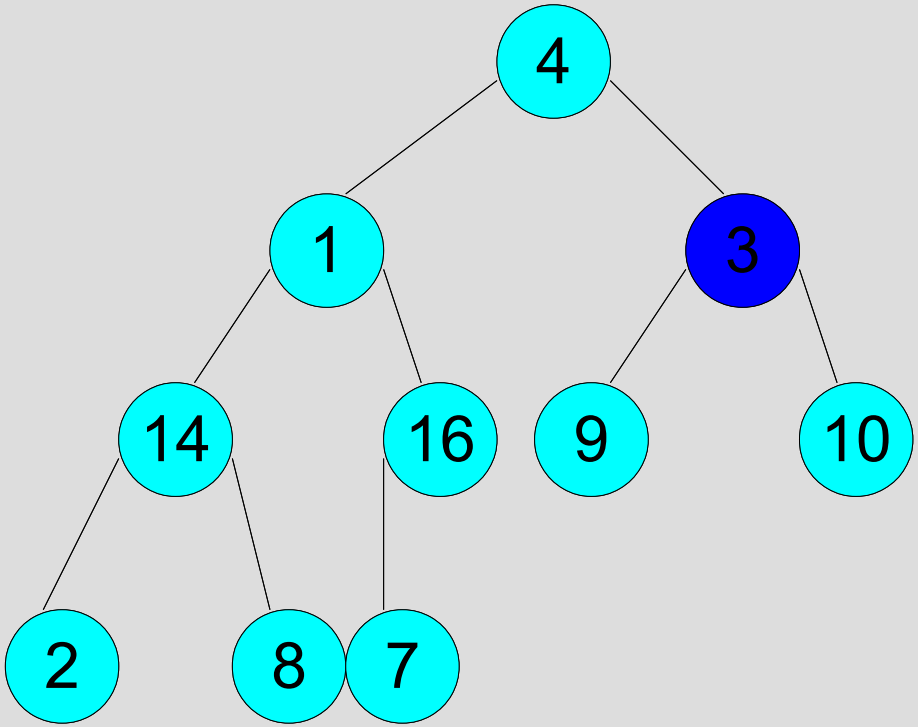
```
BUILD-MAX-HEAP(A)
 tamanho-do-heap[A]←comprimento[A]
 for i← comprimento[A/2] downto 1
        MAX-HEAPIFY(A,i)
```

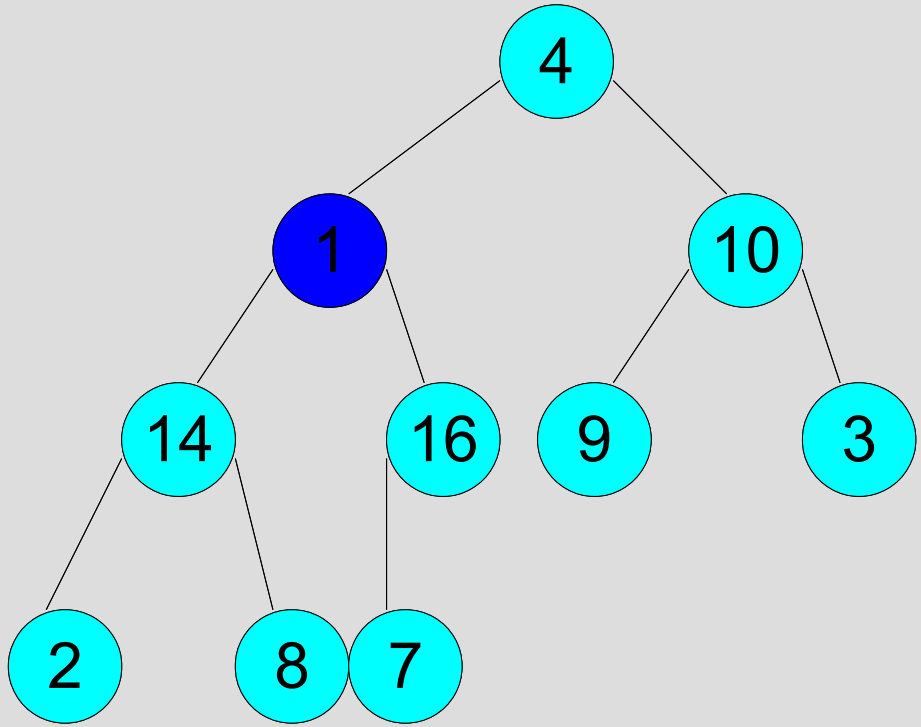
BUILD-MAX-HEAP



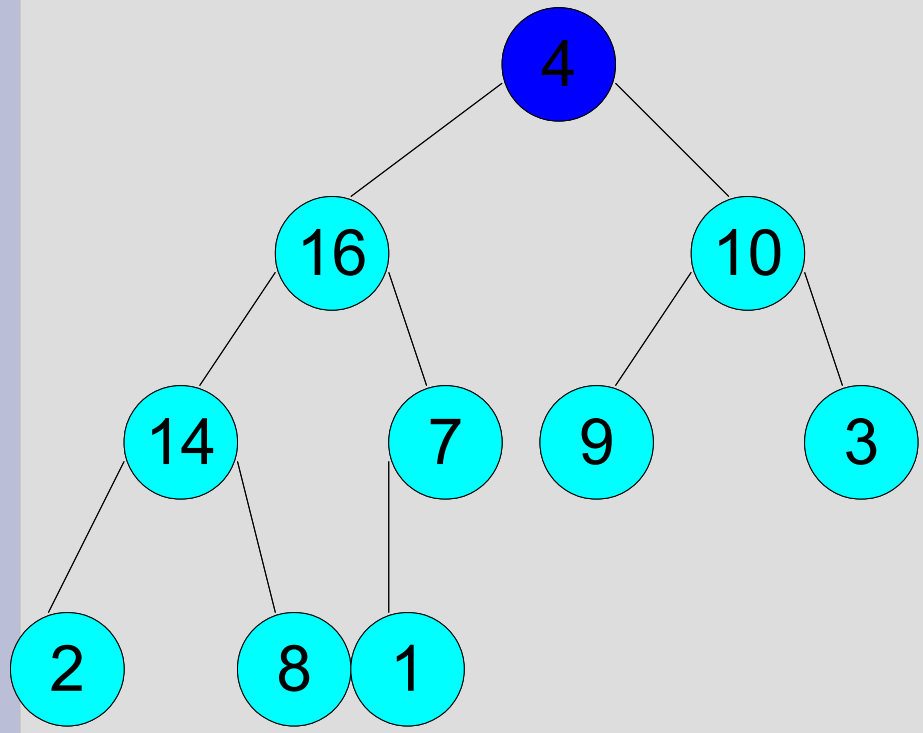


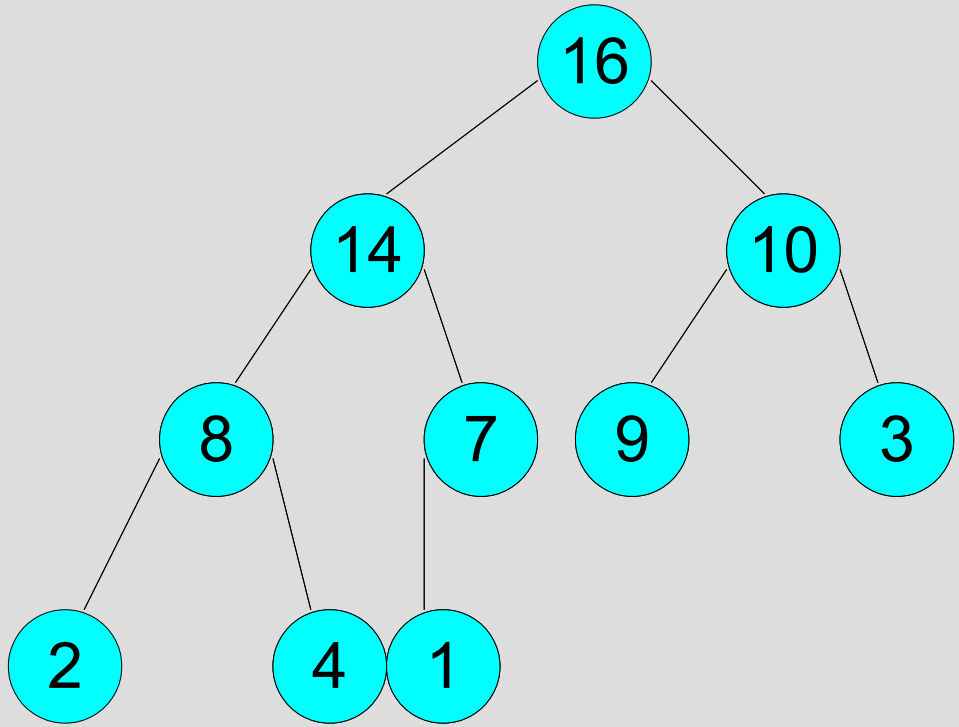
BUILD-MAX-HEAP





BUILD-MAX-HEAP





Heap-Sort

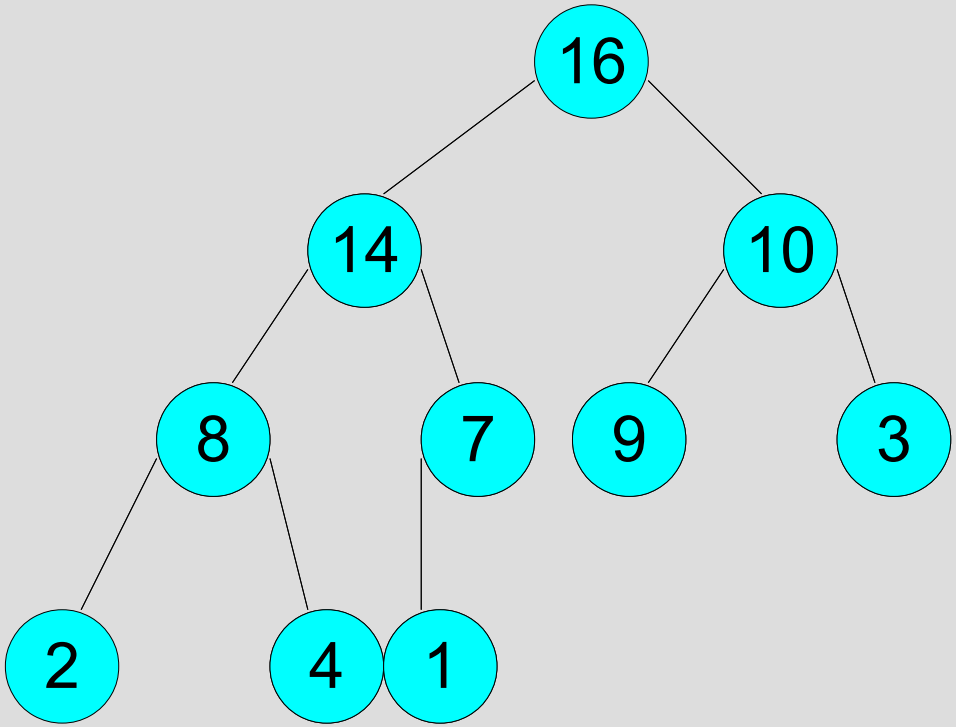
O algoritmo heapsort começa usando BUILD-MAX-HEAP para construir um heap no arranjo de entrada A[1..n], onde n = comprimento[A].

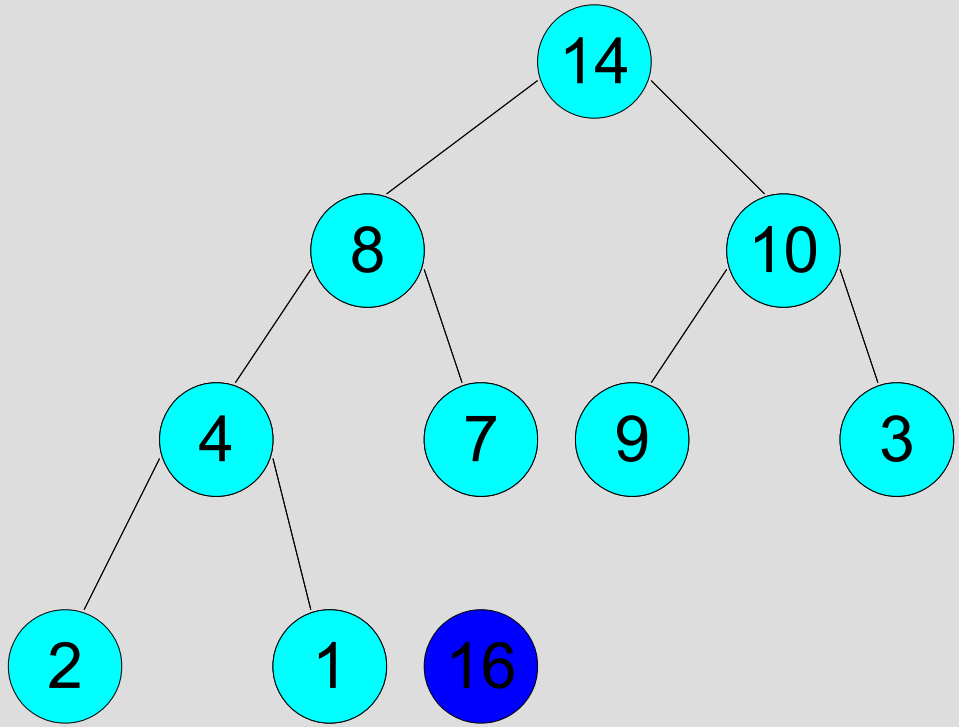
Tendo em vista que o elemento máximo do arranjo está armazenado na raiz A[1], ele pode ser colocado em sua posição final correta, trocando-se esse elemento por A[n].

Heap-Sort

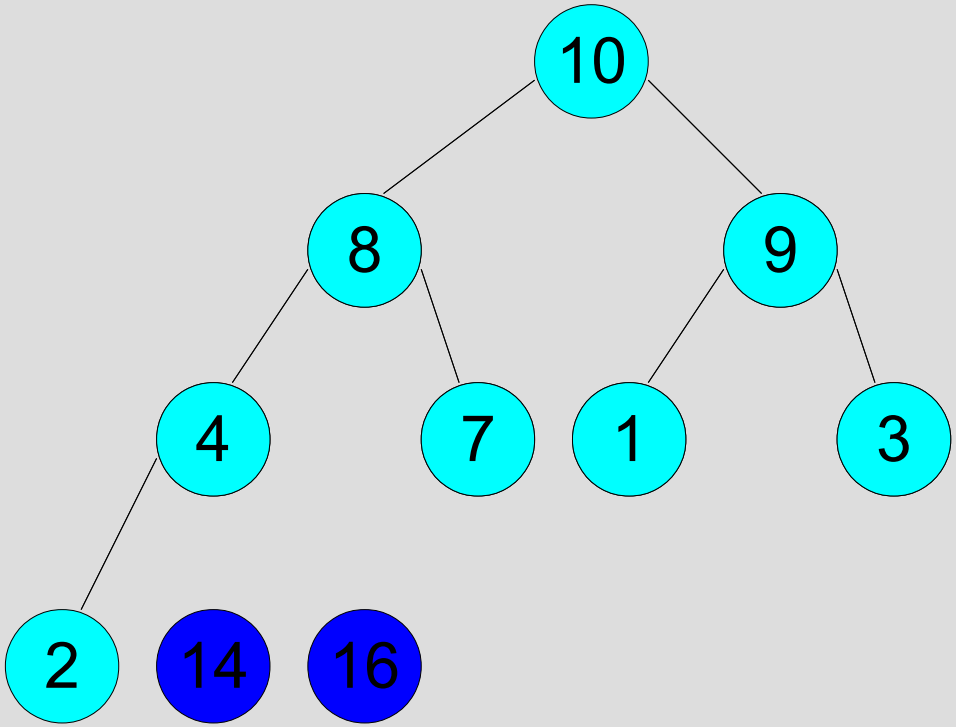
```
HEAPSORT(A)
 BUILD-MAX-HEAP(A)
 for i← comprimento[A] downto 2
     trocar A[1]↔A[i]
     tamanho-do-heap[A]←tamanho-do-heap[A]-1
     MAX-HEAPIFY(A,1)
```

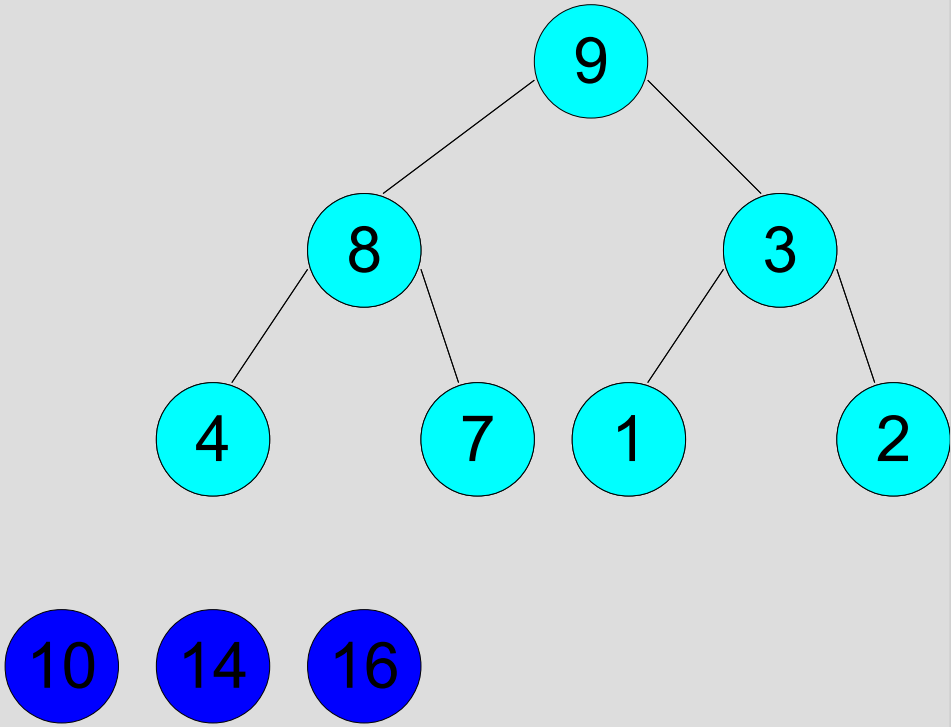
Heap-Sort



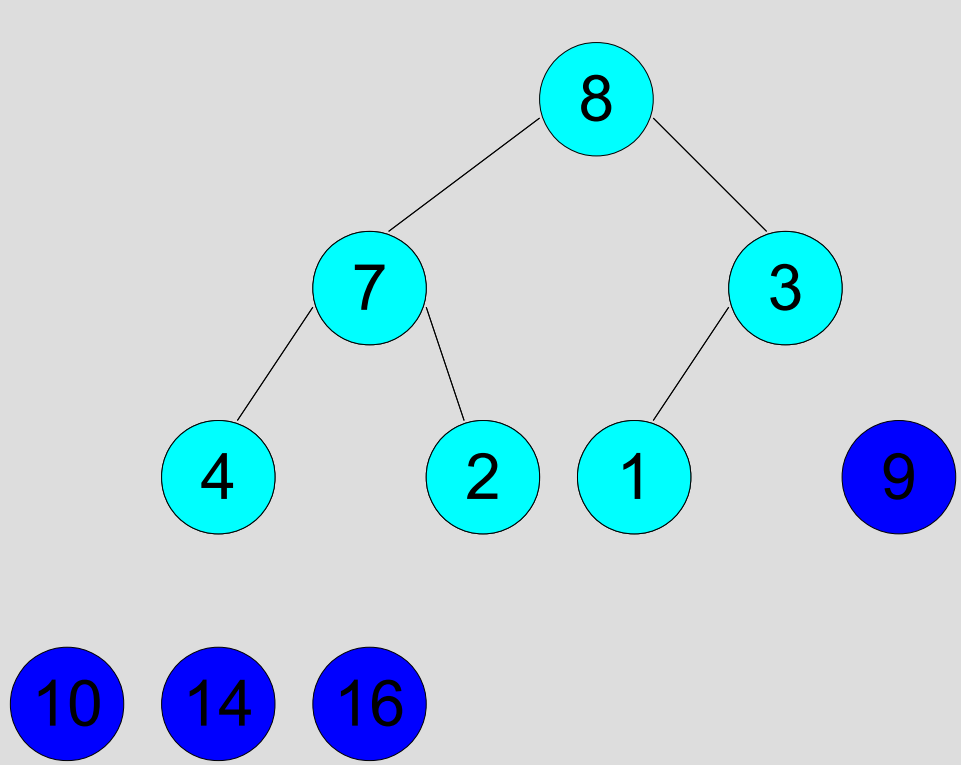


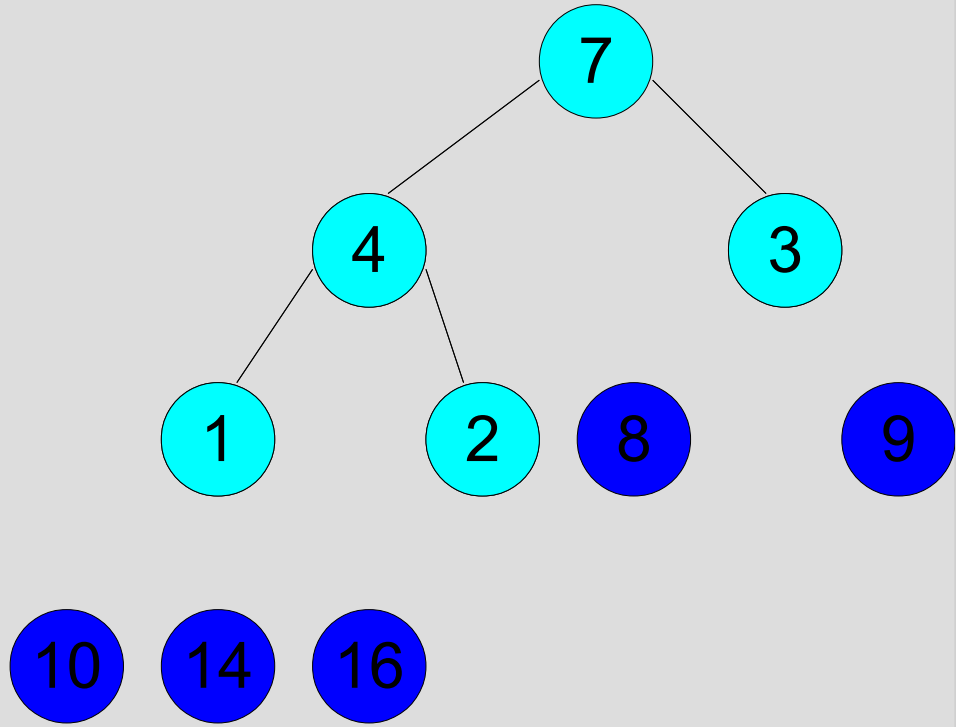
Heap-Sort



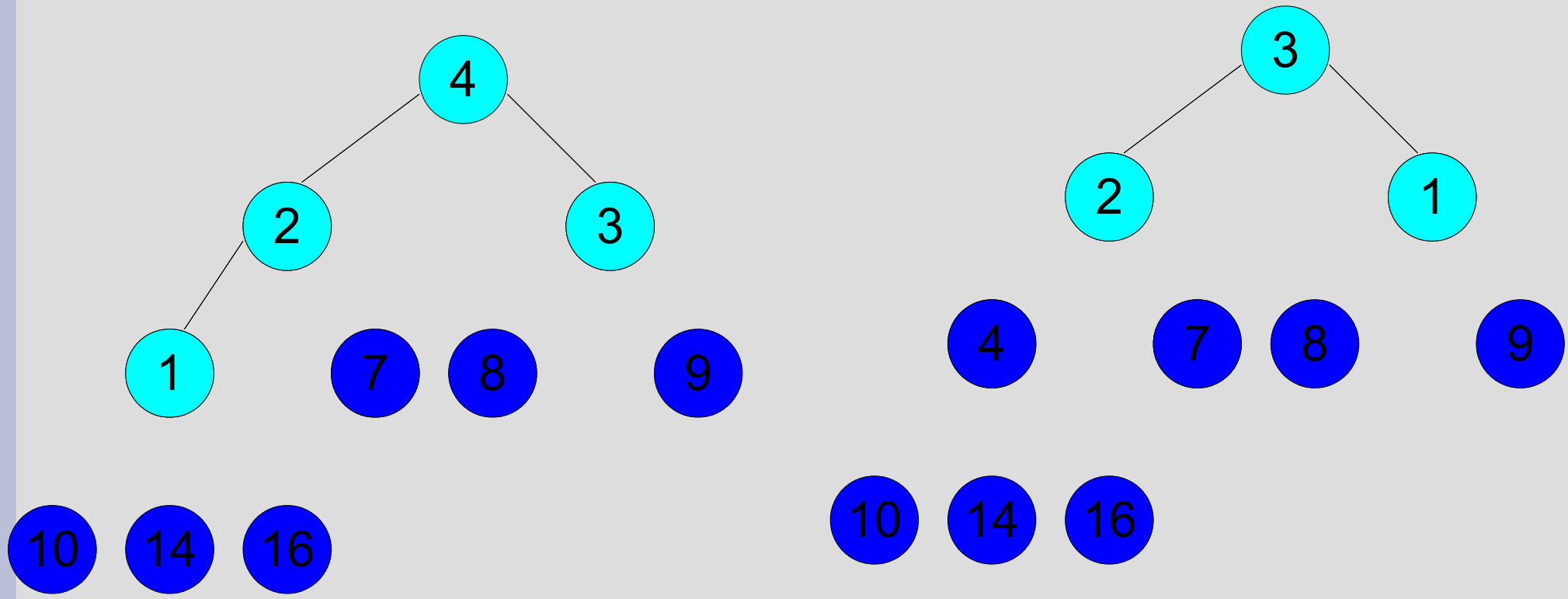


Heap-Sort





Heap-Sort



Heap-Sort