Heap Sort

Google Prep Tech 2024

Ordenação | Heap Sort

Heap Sort

Conceito

Heap sort é uma técnica de ordenação baseada em comparações. Ela visualiza os elementos da matriz como uma **heap**.

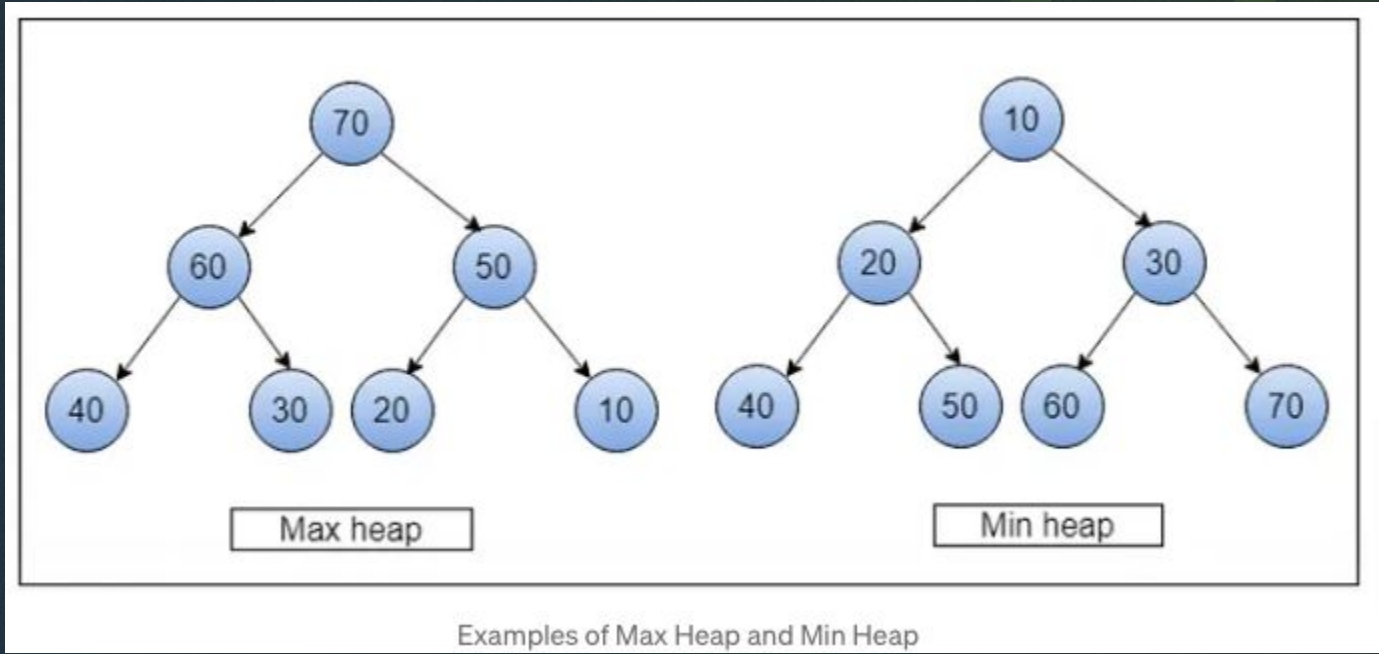
Heap

Uma `heap` é uma estrutura de dados baseada em uma `árvore binária completa` onde todos os nós não folhas são maiores (*Max Heap*) ou menores (*Min Heap*) que seus filhos. Logo, um `heap` é:

- `Heap mínimo` se todos os nós forem **menores** que seus filhos.
- `Heap máximo` se todos os nós forem **maiores** que seus filhos.

Heap Sort

Conceito



Examples of Max Heap and Min Heap

Fonte: https://medium.com/@mopurisreenath/what-is-binary-heap-25cd0f8bed24

Heap Sort

Conceito

Principais **operações** que garantem a ordenação eficiente dos elementos:

1. Construção do Max Heap

Primeiro, o array de entrada é transformado em um *max heap*, que é uma estrutura de dados onde o maior elemento está na raiz, e cada nó pai é maior que seus nós filhos.

2. Heapify

A função `heapify` é usada para manter a **propriedade** do heap. Ela é aplicada repetidamente para garantir que a estrutura do heap seja mantida após cada modificação. A `heapify` é chamada em todos os nós não-folha, começando do último nó não-folha até a raiz.

3. Ordenação

Depois de construir o *max heap*, o maior elemento (na raiz) é **trocado** com o último elemento do heap. O tamanho do heap é então reduzido em um, e a função `heapify` é chamada novamente para restaurar a propriedade do heap para o restante dos elementos. Esse processo é repetido até que todos os elementos estejam ordenados.

Heap Sort

Conceito

Construir o Max Heap

Para cada nó não-folha, aplicar a função `heapify` para garantir que a propriedade do max heap seja mantida.

Ordenar o Array

- Trocar o elemento da raiz (maior elemento) com o último elemento do heap.
- **Reduzir** o tamanho do heap em um.
- Aplicar `heapify` na raiz para restaurar a propriedade do max heap.
- Repetir até que o heap tenha apenas um elemento.

Heap Sort

Algoritmo

```java
public class HeapSort {
    public void sort(int arr[]) {
        int n = arr.length;

        // Construir o max heap
        for (int i = n / 2 - 1; i >= 0; i--) {
            heapify(arr, n, i);
        }

        // Extrair elementos do heap um por um
        for (int i = n - 1; i > 0; i--) {
            // Mover a raiz atual para o final
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            // Chamar heapify na heap reduzida
            heapify(arr, i, 0);
        }
    }
```

Heap Sort

Algoritmo

```c
void heapify(int arr[], int n, int i) {
    int largest = i; // Inicializar o maior como raiz
    int left = 2 * i + 1; // Filho esquerdo
    int right = 2 * i + 2; // Filho direito

    // Se o filho esquerdo é maior que a raiz
    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }

    // Se o filho direito é maior que o maior até agora
    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    // Se o maior não é a raiz
    if (largest != i) {
        int swap = arr[i];
        arr[i] = arr[largest];
        arr[largest] = swap;

        // Recursivamente heapificar a subárvore afetada
        heapify(arr, n, largest);
    }
}
```

Análise da Complexidade:

Complexidade de Tempo: **O(n log n)**

Complexidade de Espaço: **O(1);**

Heap Sort

Algoritmo

Solução usando Java Collection

```java
import java.util.*;

public class HeapSortUsingSTL {

    // Function to perform the heap sort
    public static void heapSort(int[] arr)
    {
        PriorityQueue<Integer> maxHeap
            = new PriorityQueue<>(
                Collections.reverseOrder());
        for (int i = 0; i < arr.length; i++) {
            maxHeap.offer(arr[i]);
        }
        for (int i = arr.length - 1; i >= 0; i--) {
            arr[i] = maxHeap.poll();
        }
    }
```

Análise da Complexidade:

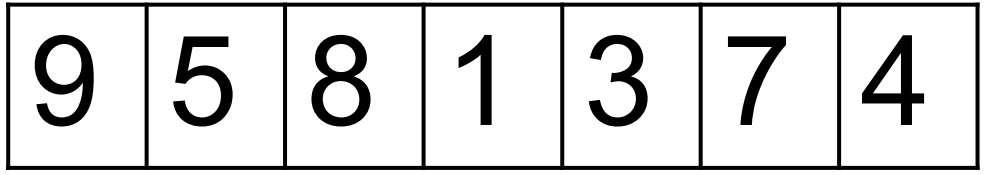
Complexidade de Tempo: **O(n log n)**

Complexidade de Espaço: **O(n);**

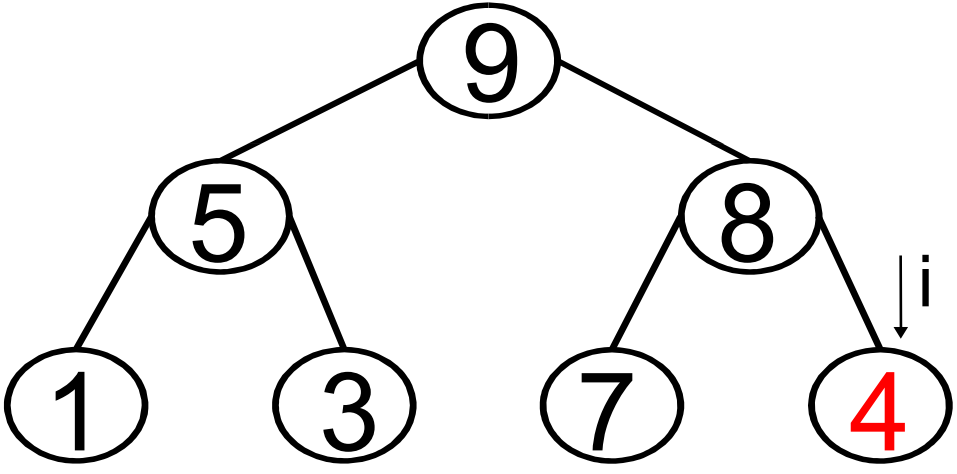
Execução ilustrativa

Heapsort

Código

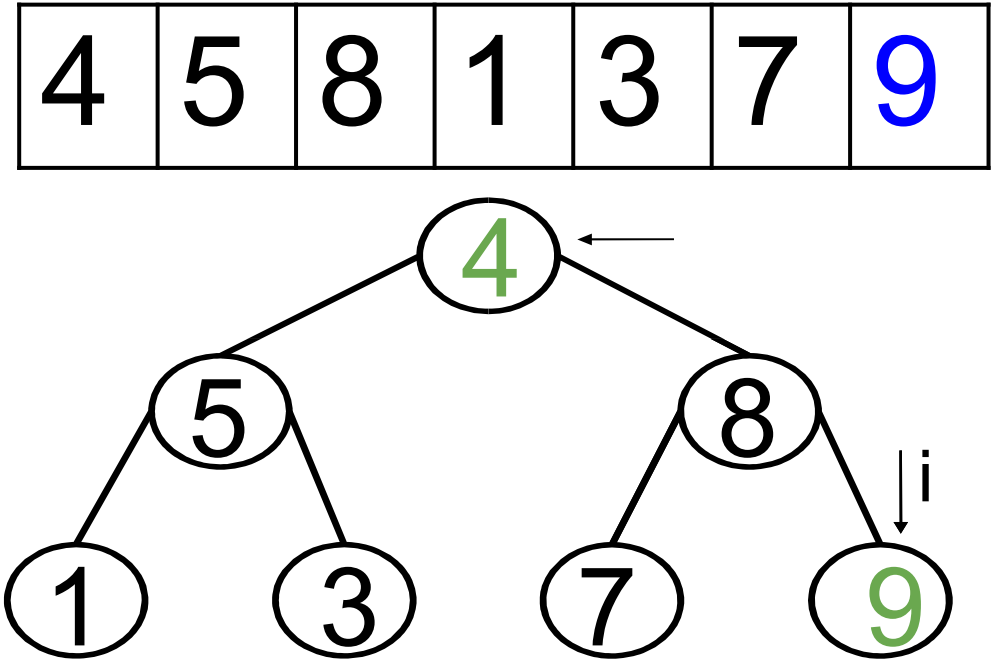


Max Heap



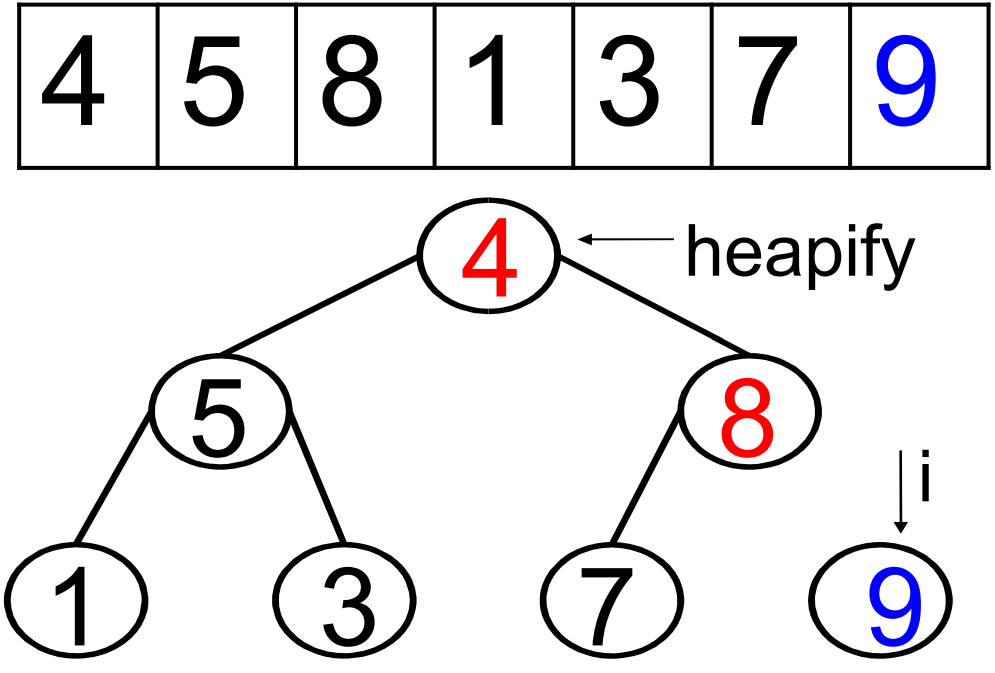
Heapsort

Código



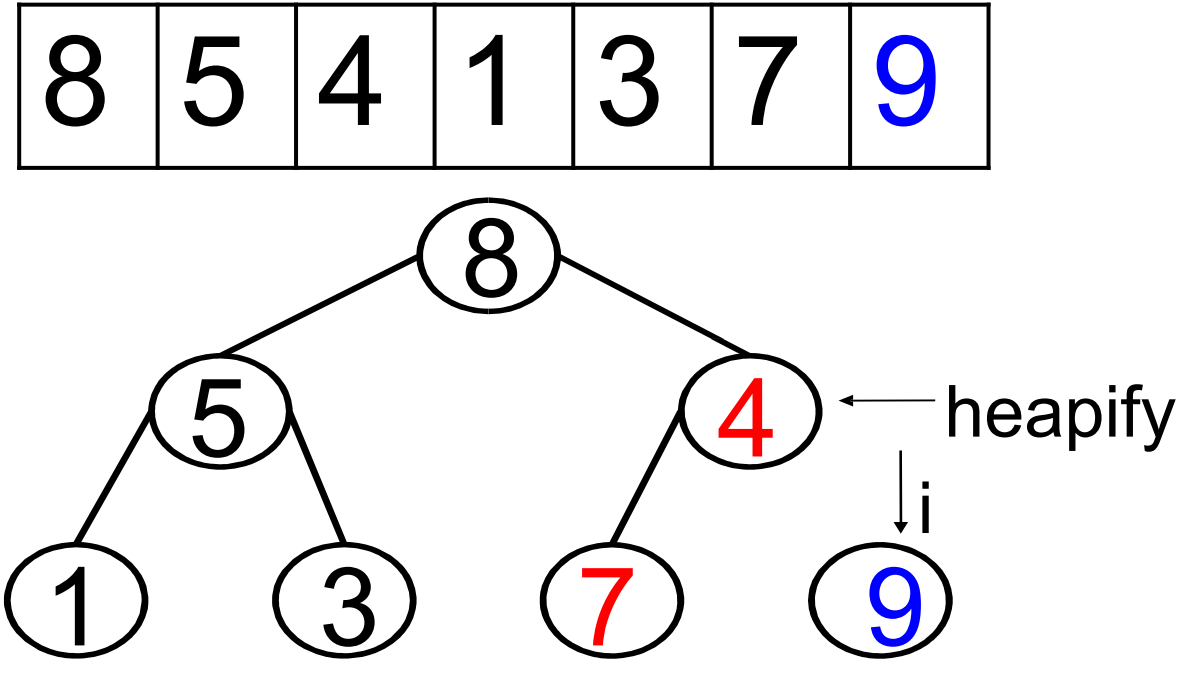
Heapsort

Código



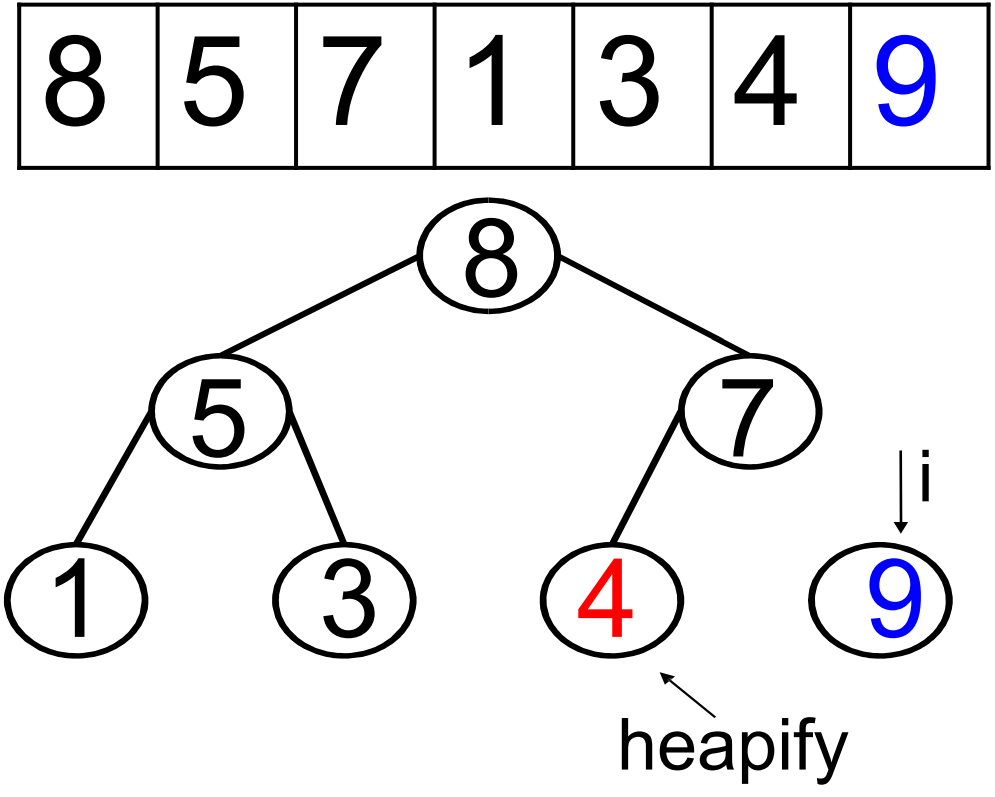
Heapsort

Código



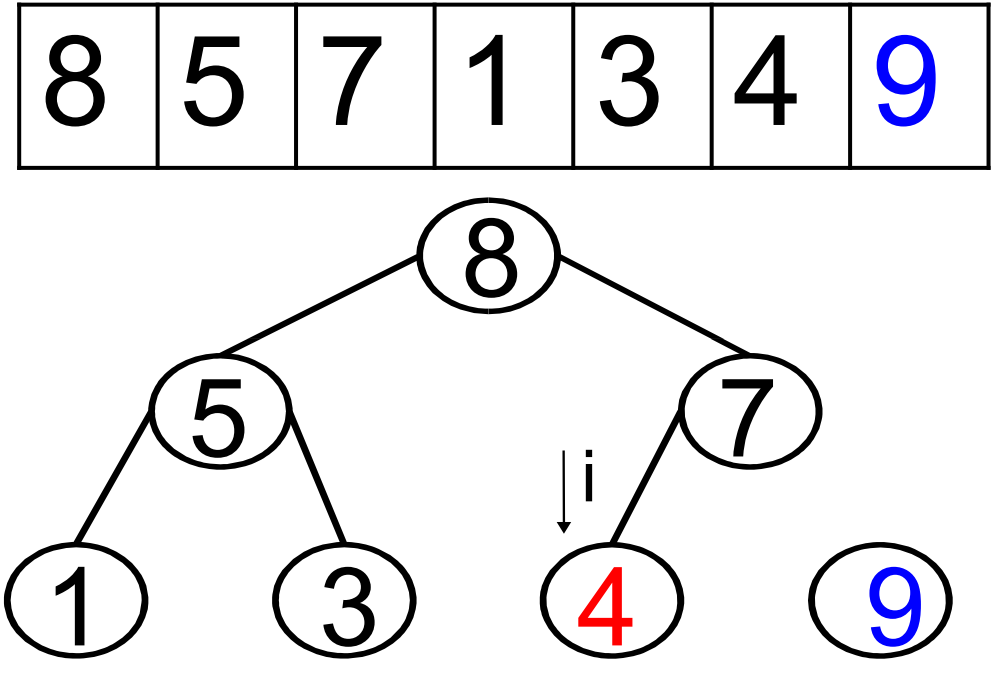
Heapsort

Código



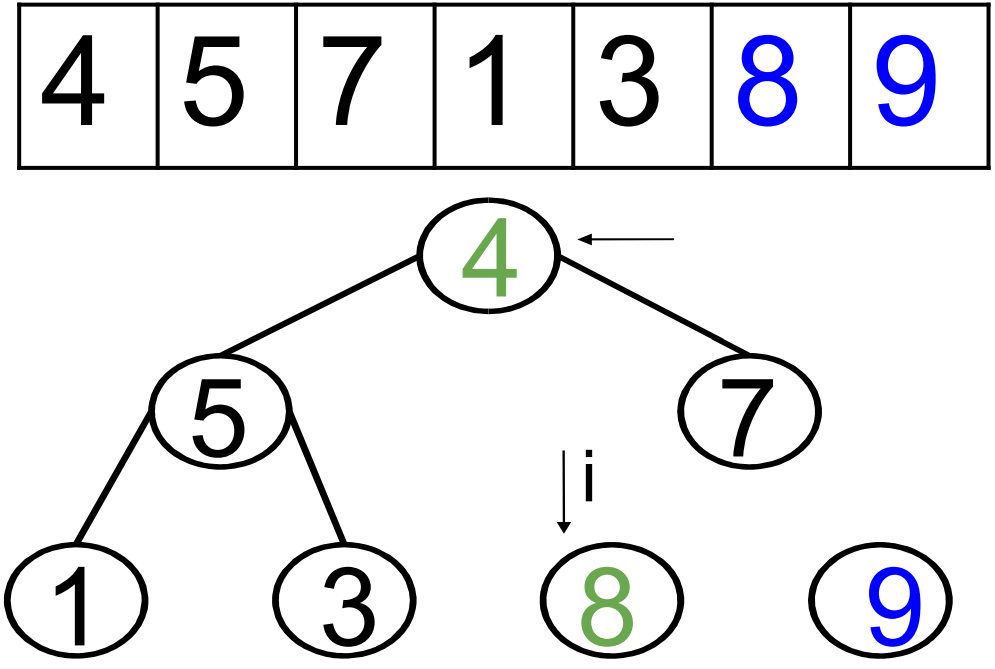
Heapsort

Código



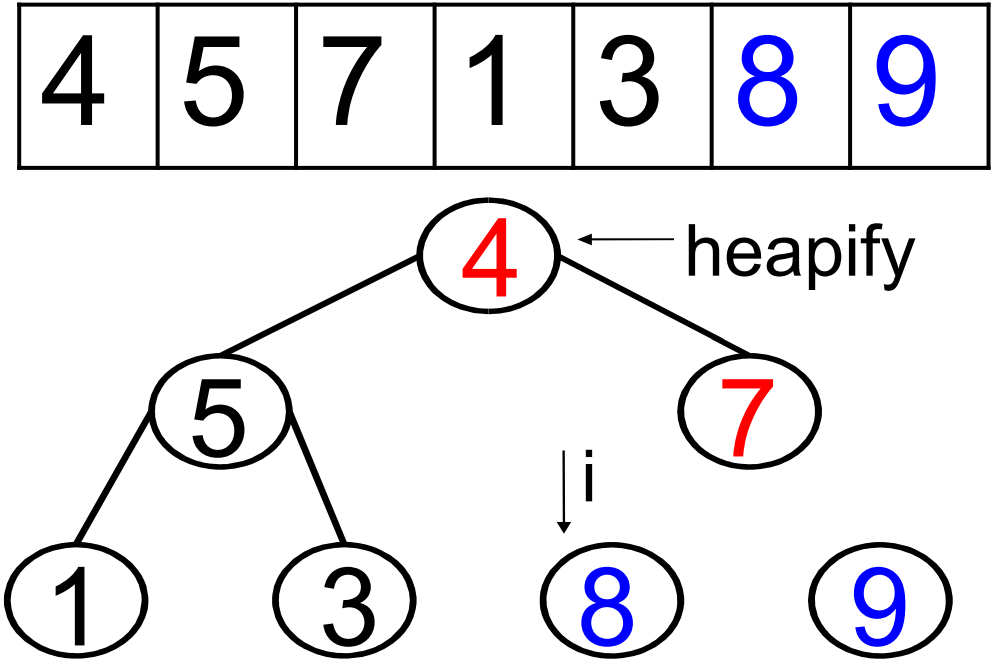
Heapsort

Código



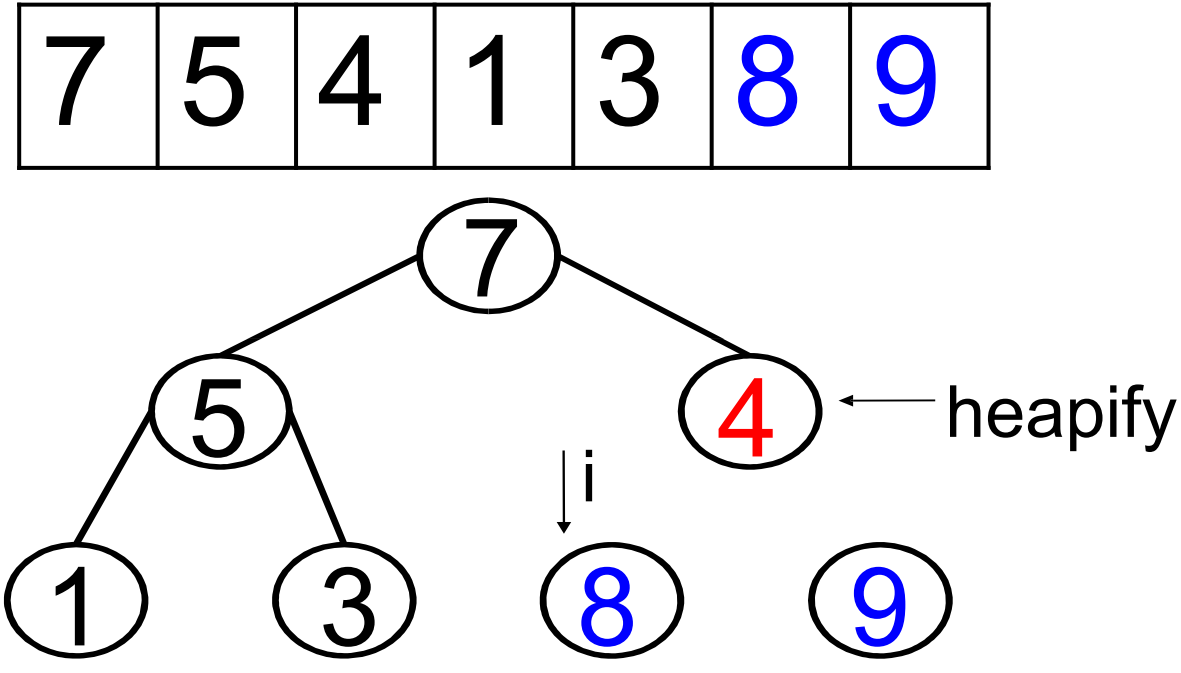
Heapsort

Código



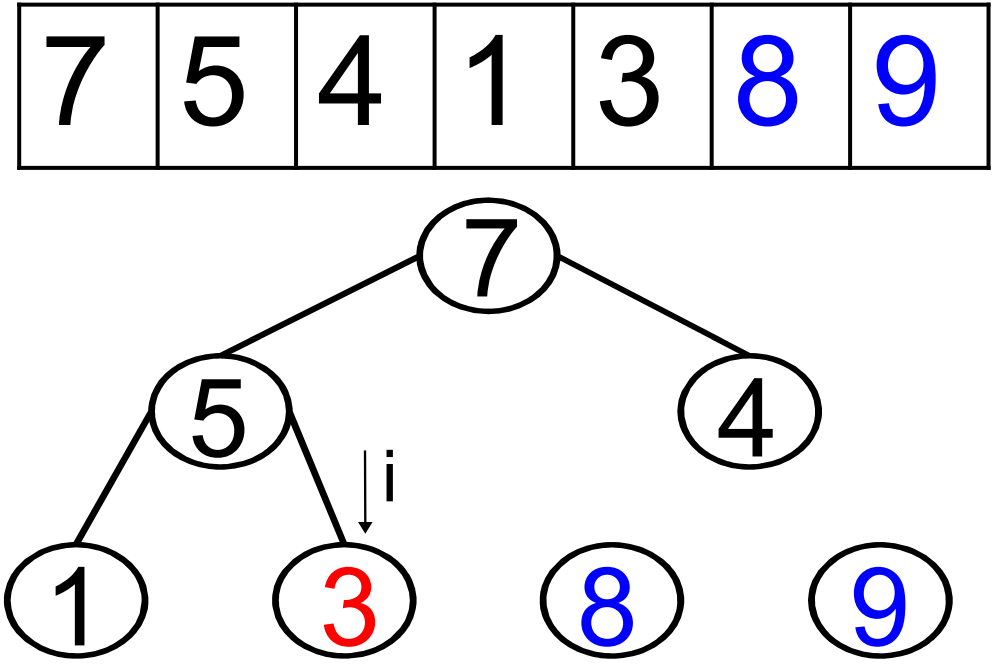
Heapsort

Código



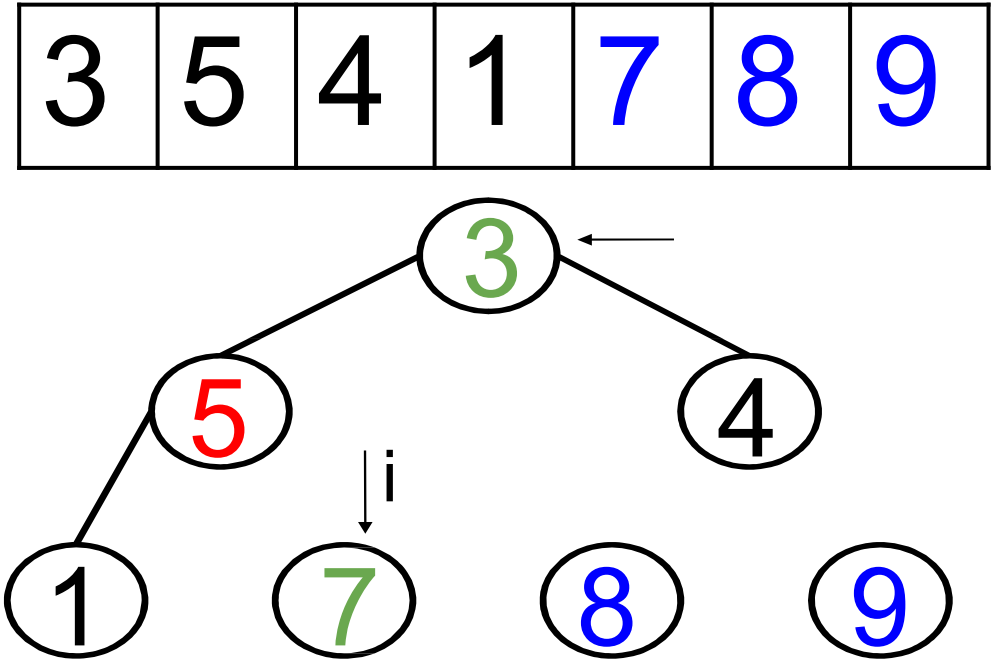
Heapsort

Código



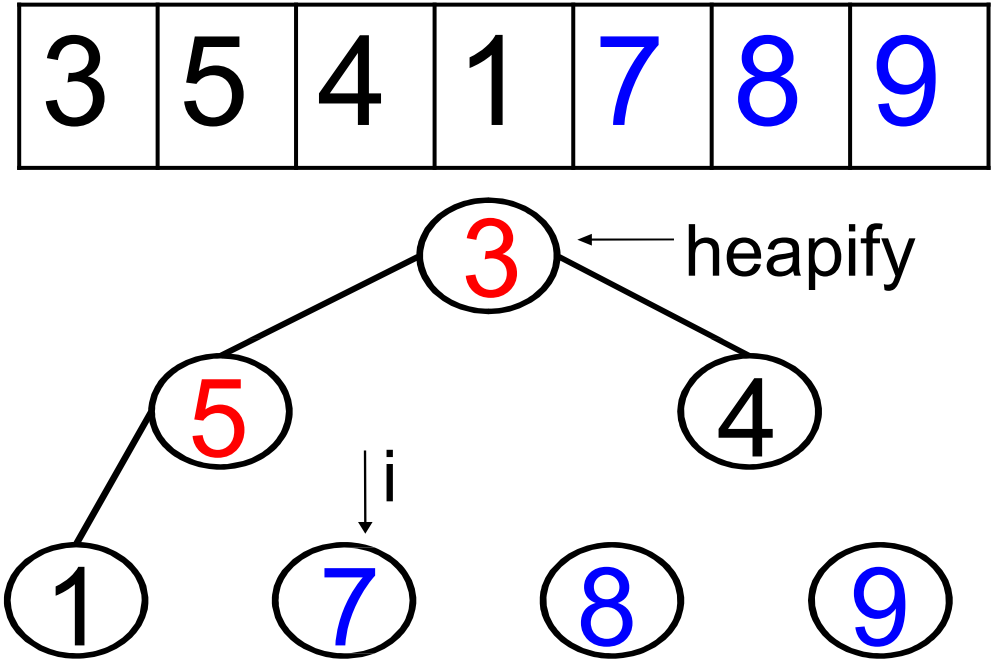
Heapsort

Código



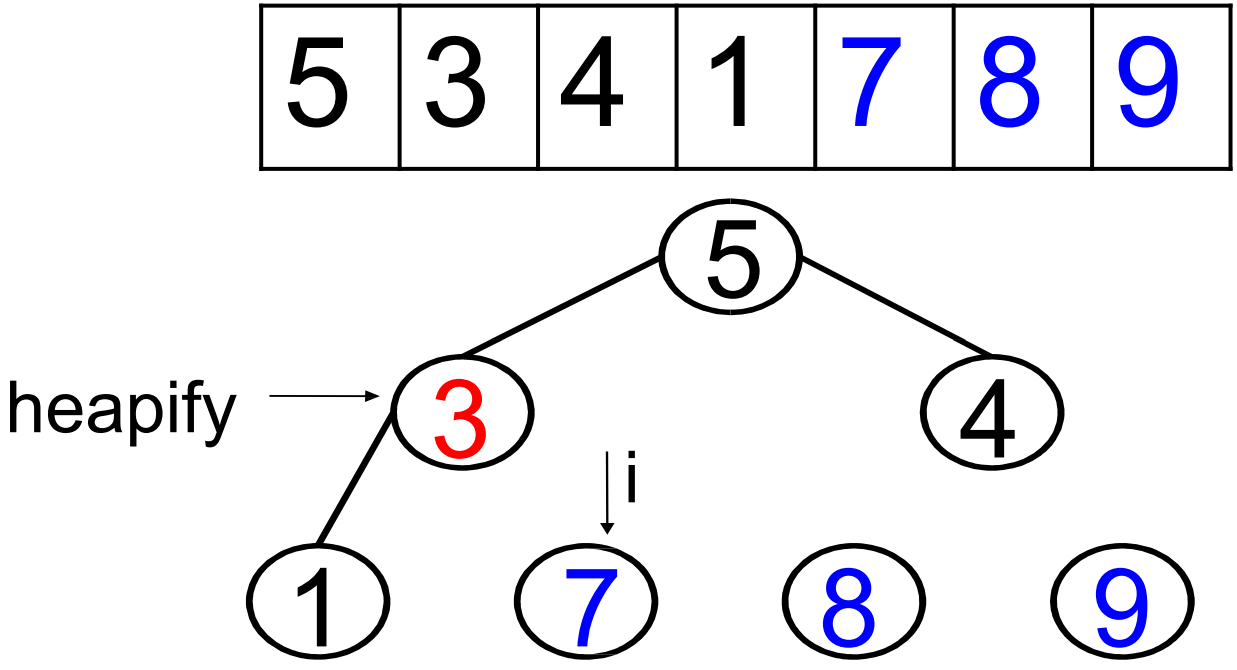
Heapsort

Código



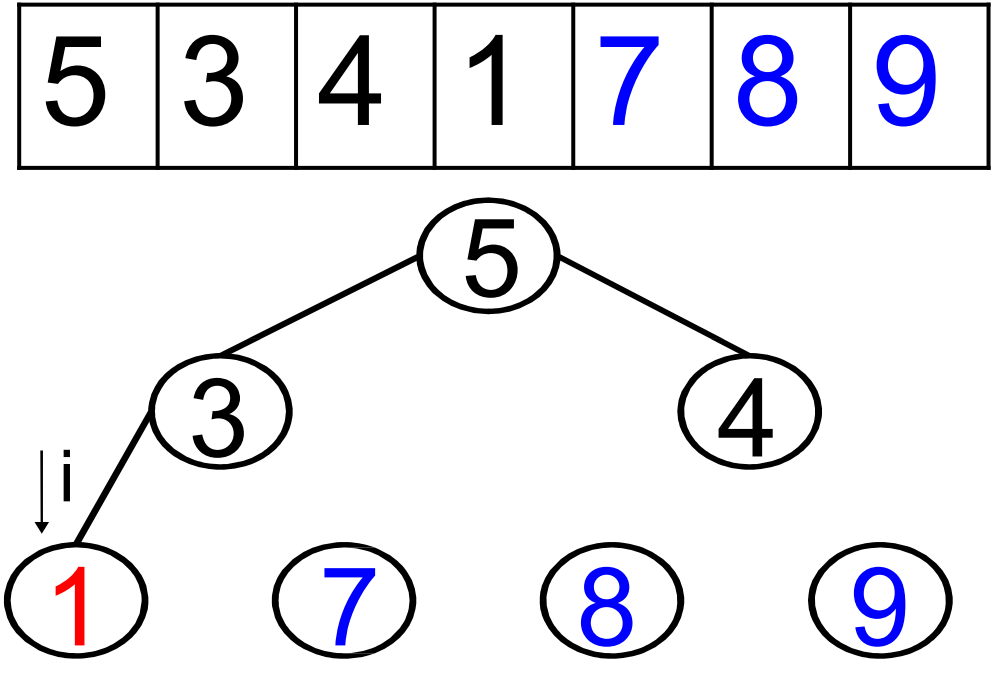
Heapsort

Código



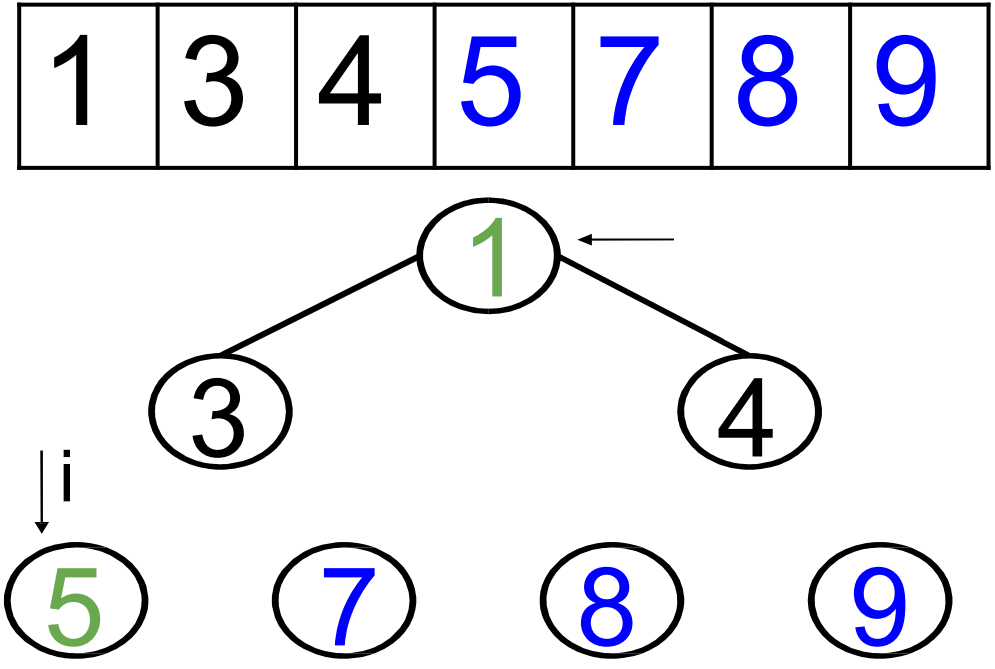
Heapsort

Código



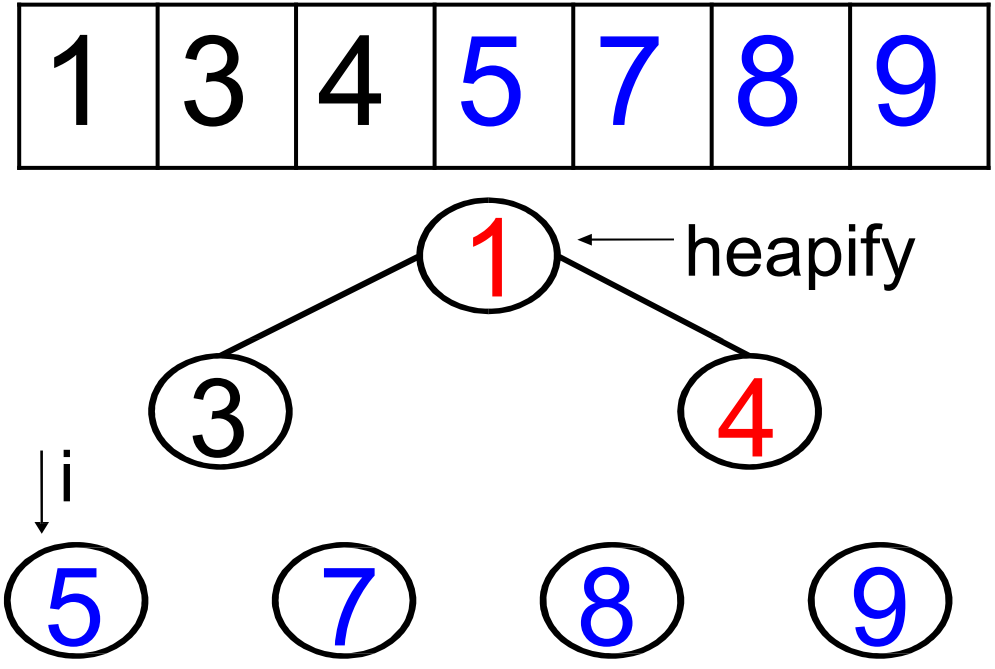
Heapsort

Código



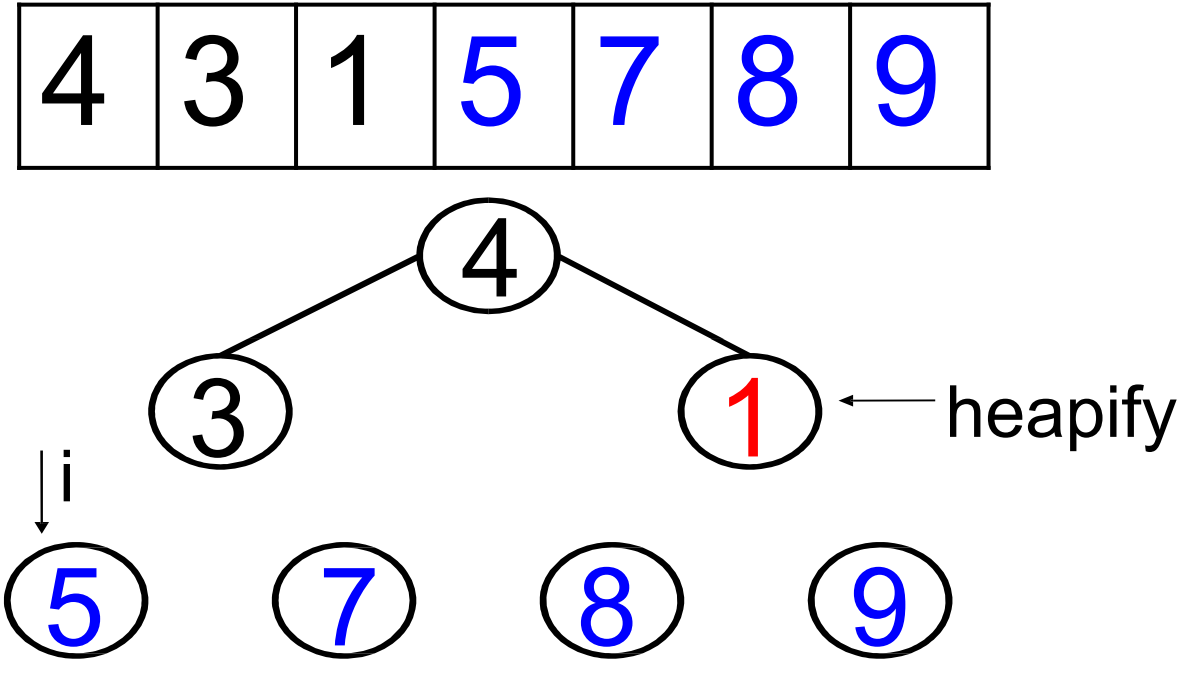
Heapsort

Código



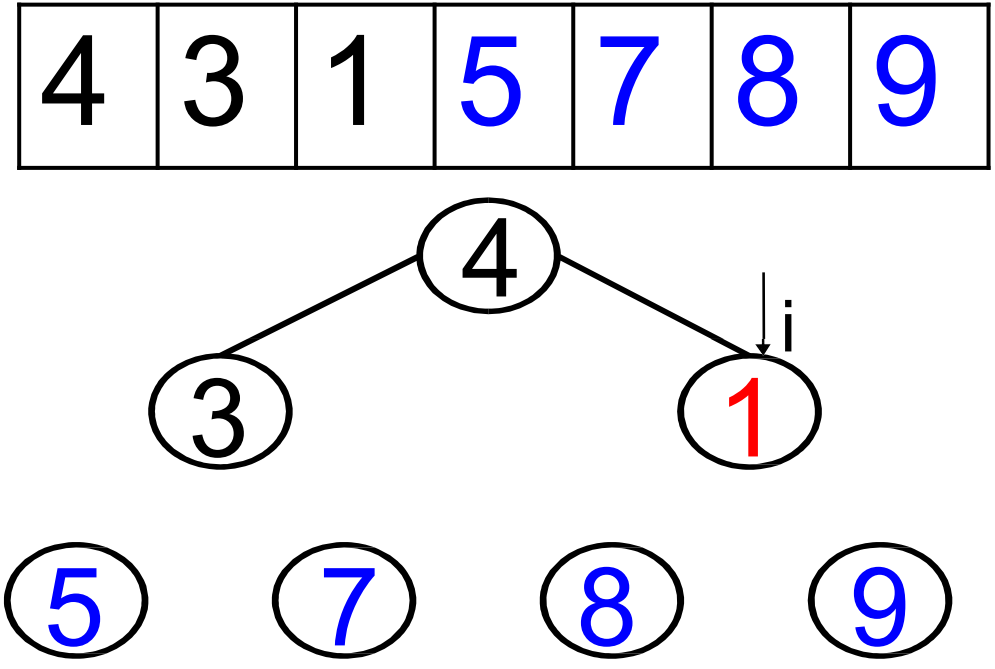
Heapsort

Código



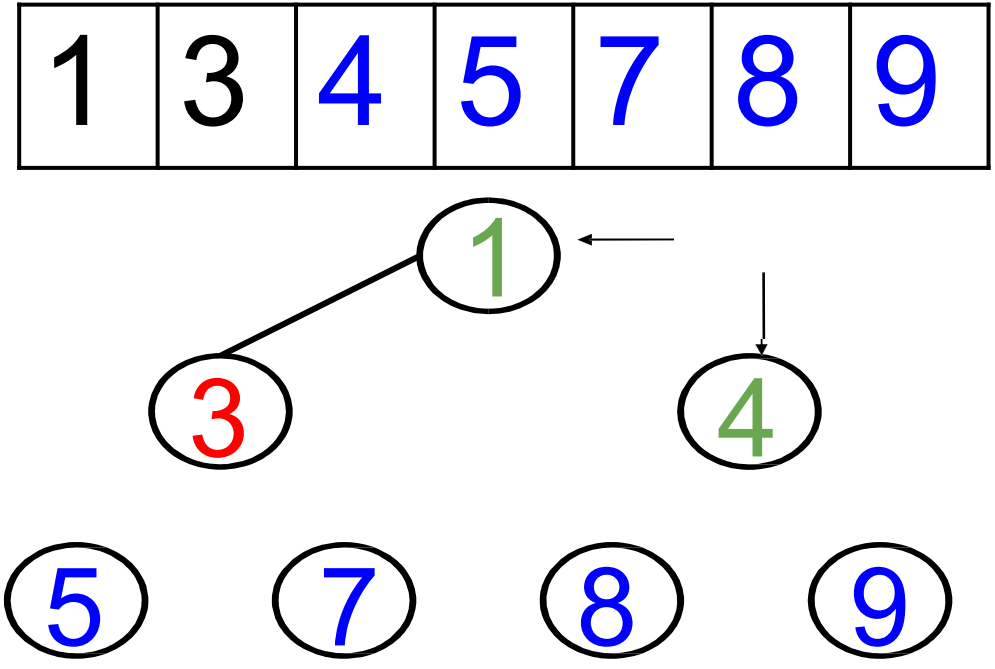
Heapsort

Código



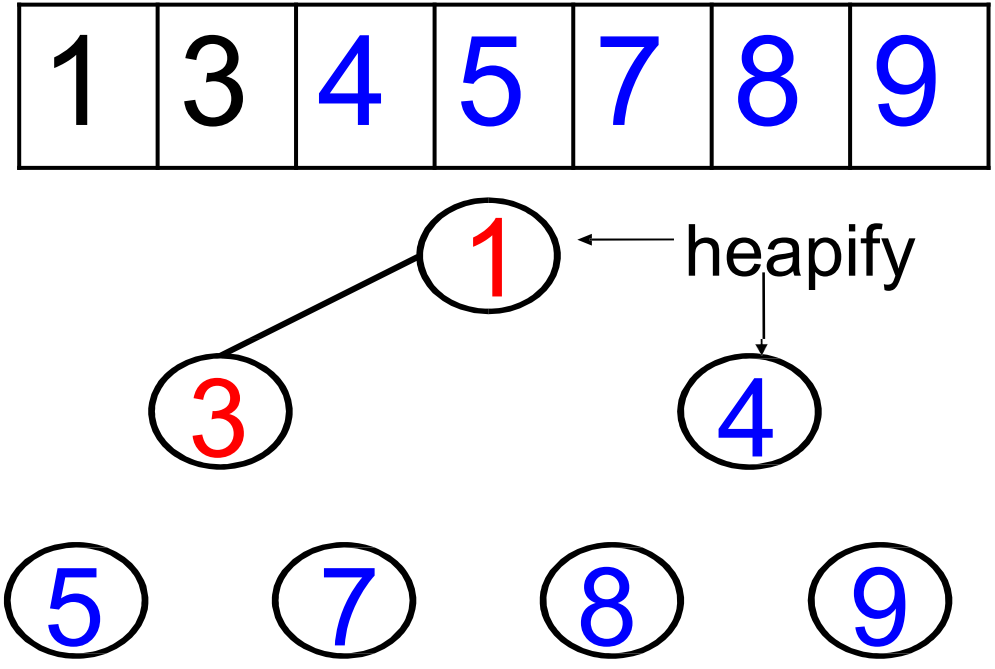
Heapsort

Código



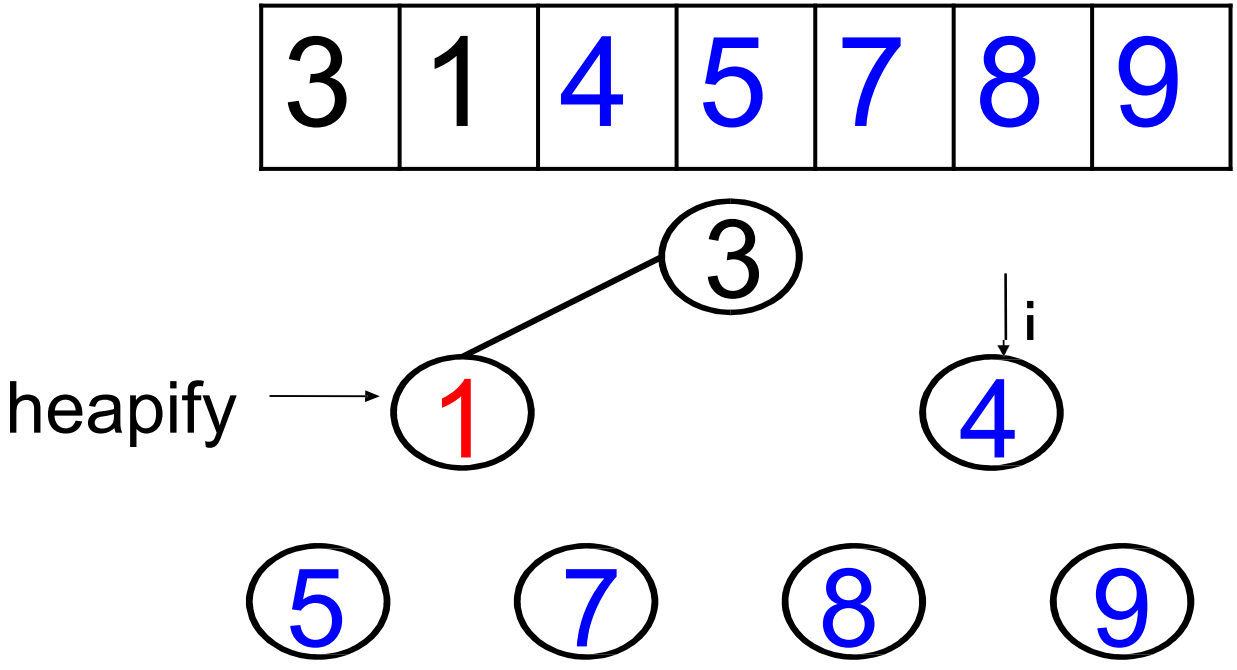
Heapsort

Código



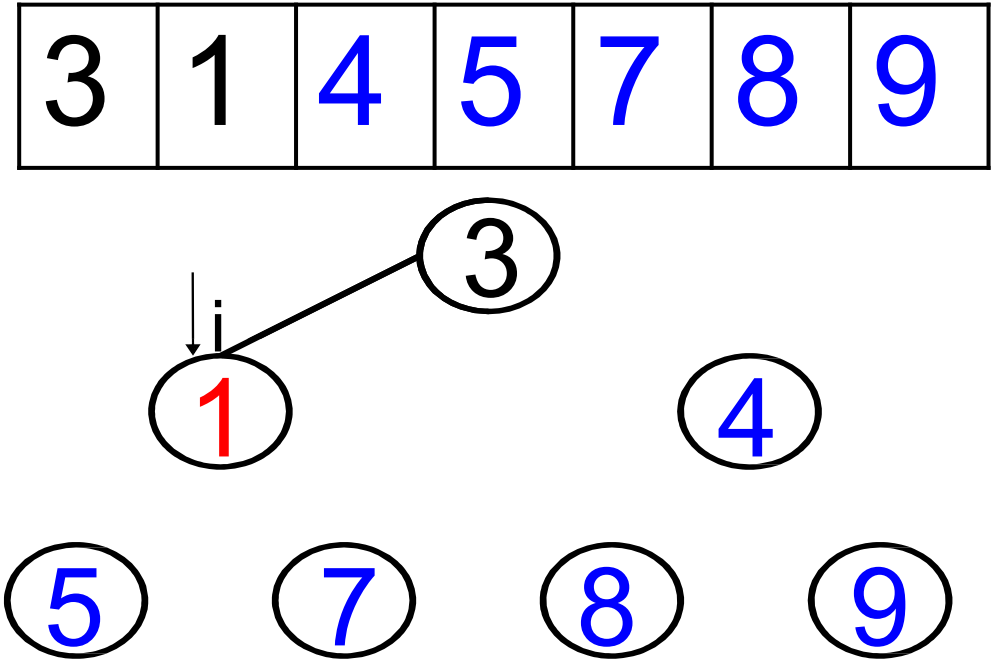
Heapsort

Código



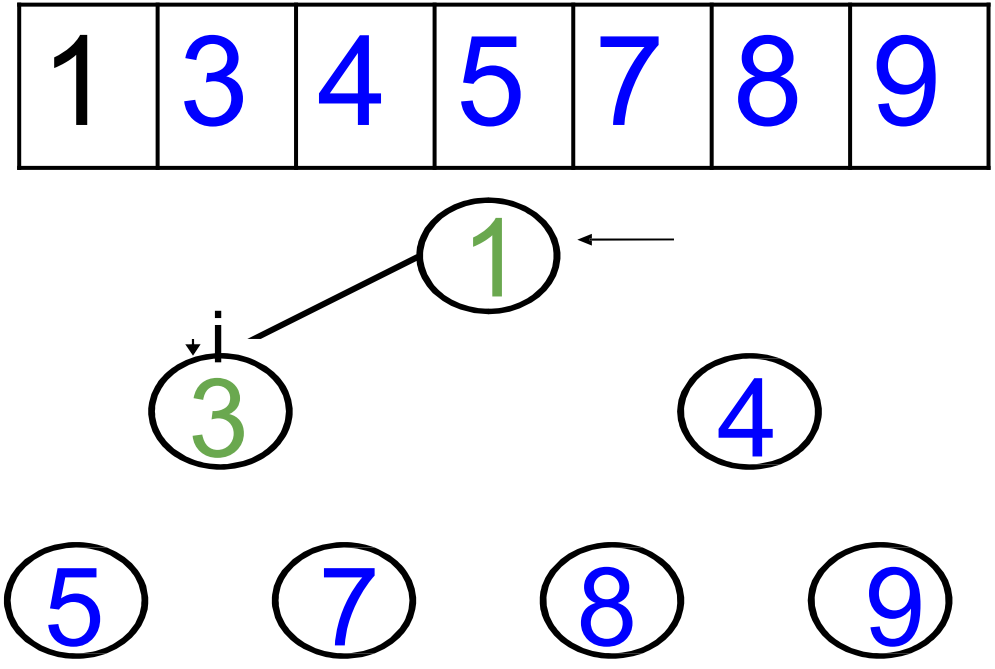
Heapsort

Código



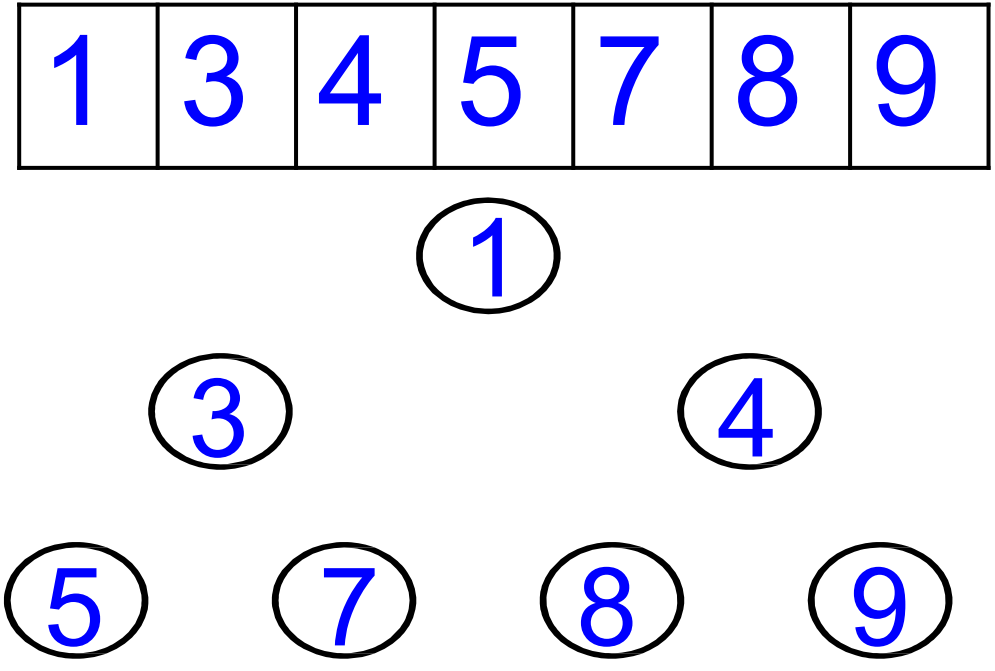
Heapsort

Código



Heapsort

Código



Obrig.ada