Engenharia de Computação

Pesquisa e Classificação de Dados

Aula 7 – HeapSort

Prof. Muriel de Souza Godoi
muriel@utfpr.edu.br

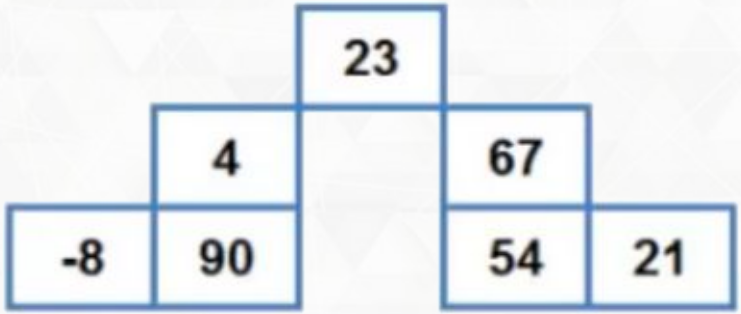
HeapSort

- Ordenação usando Heap (monte)
  - Heap: Vetor que simula uma árvore binária completa (exceção do último nível)
  - Todo elemento pai possui dois elementos como filhos
    - `Pai[i]  => FilhoEsq:[2*i + 1] / FilhoDir [2*i + 2]`

HeapSort - MaxHeap

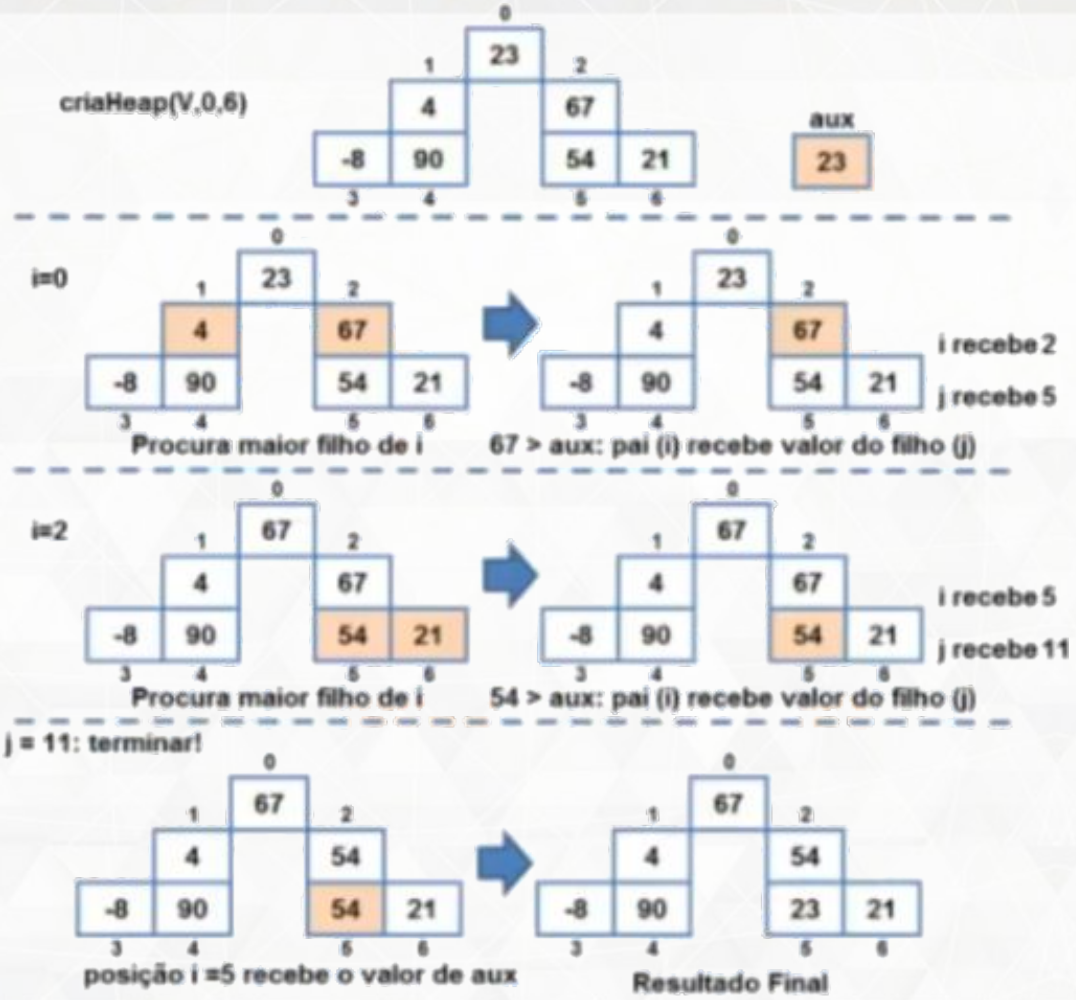
- Um Heap é um MaxHeap se:
  - Todos os nós pais são maiores que seus nós filhos
  - **Consequência**: O maior elemento está na raiz da árvore

Heap



MaxHeap

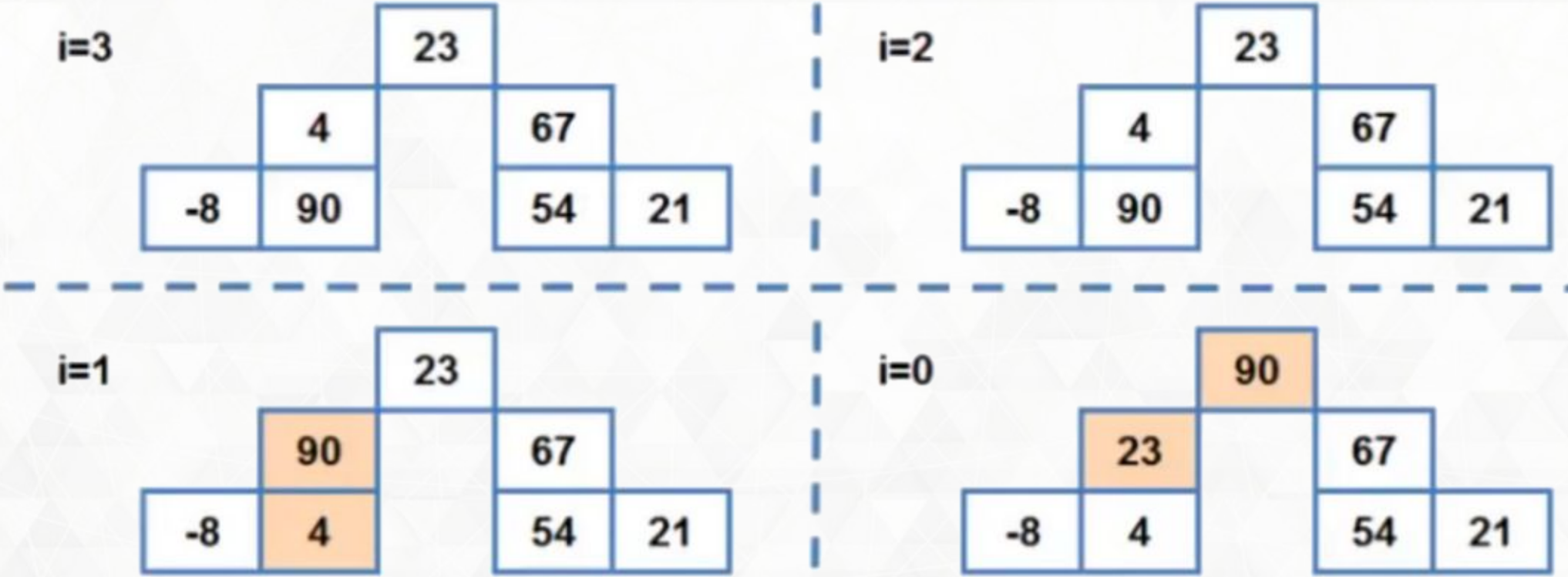
HeapSort - criaHeap



HeapSort - criaHeap

1º comando for: criaHeap(V,i,6)

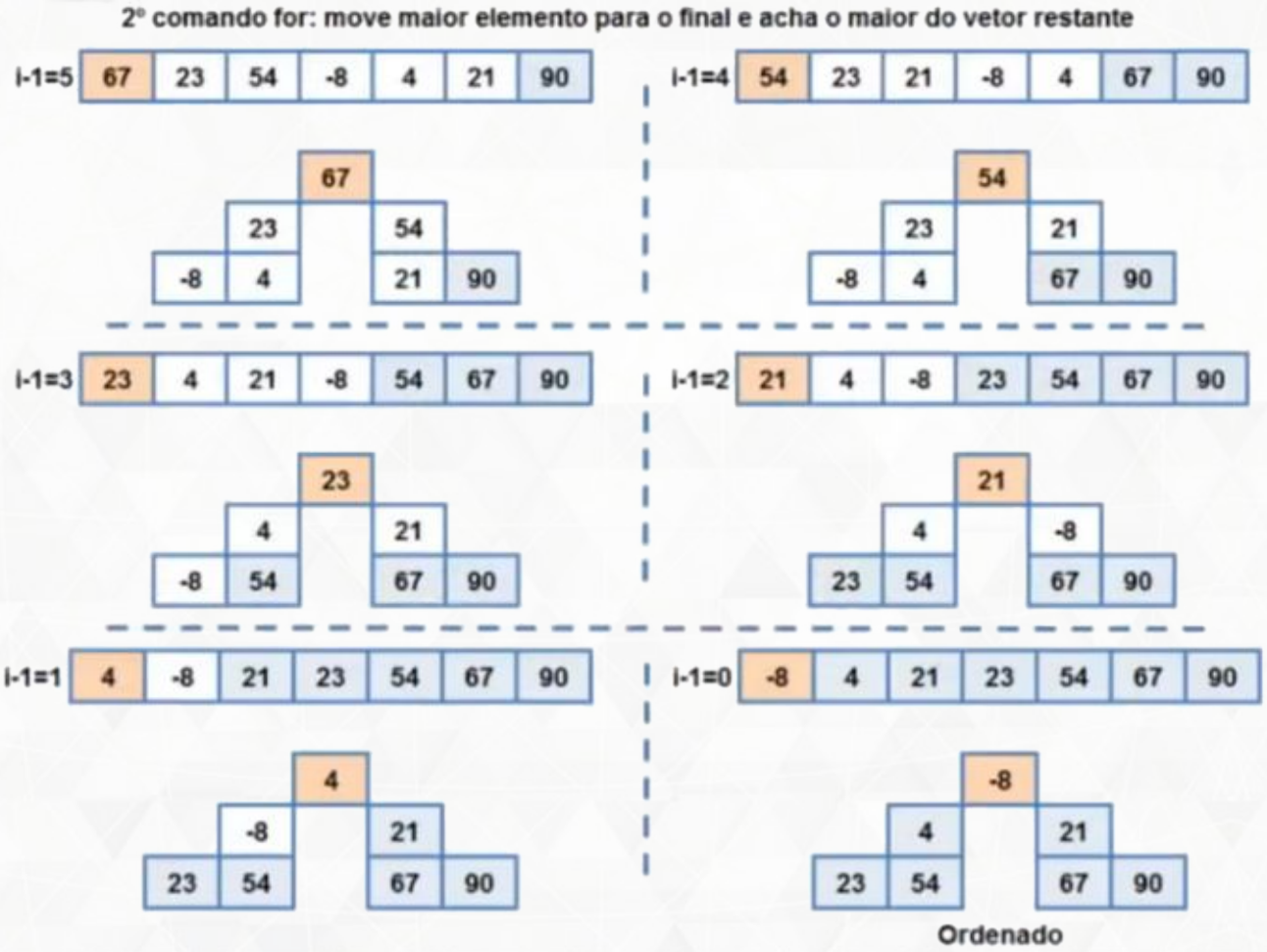
Elemento Pai fica maior que os Filhos



HeapSort

- Como usar um Max Heap para ordenar?
  - **Passo 1:**
    - Transformar o heap em um max-heap
  - **Passo 2:**
    - Move raiz (maior valor) para o final(ultima posicao)
    - Reconstrói o heap do vetor restante

HeapSort


2º comando for: move maior elemento para o final e acha o maior do vetor restante
i-1=5
67 23 54 -8 4 21 90
67
23 54
-8 4 21 90
i-1=4
54 23 21 -8 4 67 90
54
23 21
-8 4 67 90
i-1=3
23 4 21 -8 54 67 90
23
4 21
-8 54 67 90
i-1=2
21 4 -8 23 54 67 90
21
4 -8
23 54 67 90
i-1=1
4 -8 21 23 54 67 90
4
-8 21
23 54 67 90
i-1=0
-8 4 21 23 54 67 90
-8
4 21
23 54 67 90
Ordenado

HeapSort - Pseudocódigo

- O algoritmo usa **2 funções**
  - **heapSort**: divide os dados em vetores cada vez menores
  - **criaHeap**: transforma um **Heap** em um **MaxHeap**

HeapSort - Pseudocódigo

- **heapSort(v, n)**
  - Para todos os índices i da metade até o início do vetor faça:
    - criaHeap(V, i, n-1)
  - Para todos os índices i de trás para frente faça:
    - Troque a posição atual com a primeira
    - criaHeap(V, 0, i-1)

HeapSort - Pseudocódigo

- **criaHeap(v, pai, fim)**
  - aux ← v[pai]
  - filho ← pai * 2 + 1
  - **Enquanto** o filho < fim **faça**:
    - **Se** filho < fim && v[filho] < v[filho +1] **então:**
      - Incrementa filho
    - **Se** aux < v[filho] **então:**
      - V[pai] ← V[filho] //Filho se torna pai
      - pai ← filho //Atualiza e repete o processo
      - filho ← pai * 2 + 1
    - **Senão**:
      - filho ← fim + 1; //Finaliza a repeticao
  - v[pai] ← pai

HeapSort - Complexidade

- Considerando um vetor com **n** elementos, o tempo de execução é sempre:
  - **O(n log n)**;
- Na prática, o HeapSort é mais lento que o QuickSort, exceto no pior caso.

Exercício

- Implemente o **HeapSort** em C considerando as seguintes assinaturas de função
  - **Função 1: heapSort**

```
/**
 * \brief Ordena o vetor usando HeapSort
 *
 * \param v vetor a ser ordenado
 * \param n tamanho do vetor
 *
 * Ordena o vetor usando o método HeapSort
 */
void heapSort(int *v, int n);
```

Exercício

- **Função 2: criaHeap**

```
/**
 * \brief Transforma um heap em maxheap
 *
 * \param v vetor a ser particionado
 * \param pai índice do nó raiz do heap
 * \param fim índice do último elemento do heap
 *
 */
void criaHeap(int *v, int pai, int fim);
```