Estrutura de Dados

Métodos de Ordenação
Heapsort

Prof. Nilton Luiz Queiroz Jr.

Métodos de ordenação

- Dentro da categoria de métodos de ordenação em memória primária existem alguns métodos de ordenação que são mais eficientes que os métodos simples:
  - Mergesort;
  - Quicksort;
  - **Heapsort**;
  - Shellsort;

Heapsort

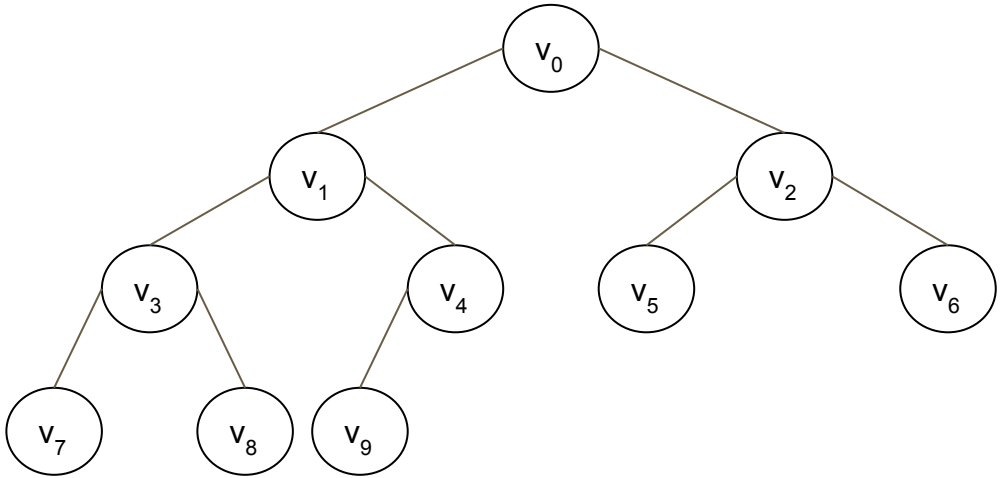
- Uma outra alternativa para ordenação de maneira eficiente é utilizar como apoio uma árvore;
  - Mais específico, uso de um heap;
- O método foi proposto por Floyd e Williams em 1964;
- Consiste basicamente em explorar propriedades de um heap para ordenar a sequência de elementos;

Heap

- Um heap é uma árvore binária completa até o penúltimo nível;
  - A diferença máxima de altura entre os nós folha é de um nível;
- Pode-se usar um vetor para representar um heap;
  - Supondo um vetor A[0..(n-1)] representando um heap e cada elemento do vetor representando um nó temos:
    - O pai de um nó que está na posição i do vetor é dado pelo elemento $\lfloor (i-1)/2 \rfloor$;
    - O nó na posição 1 não tem pai;
    - Um nó na posição i tem dois filhos, sendo o nó da posição 2i+1 seu filho esquerdo e o nó da posição 2i+2 seu filho direito;

Heap

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9



Heap

- Existem dois tipos de heap:
  - Heap Máximo:
    - Para todo nó i: A[pai(i)] $\geq$ A[i];
    - Ou seja, o **maior** elemento está na raiz da árvore;
      - Isso é válido para todas subárvores de uma árvore;
  - Heap Mínimo:
    - Para todo nó i: A[pai(i)] $\leq$ A[i];
    - Ou seja, o **menor** elemento está na raiz da árvore;
      - Isso é válido para todas subárvores de uma árvore

Heapsort

- A ideia do algoritmo é usar um heap máximo para encontrar o maior elemento restante e colocá-lo na posição adequada;
  - Pode ser visto como uma variação do método de ordenação por seleção;
- O algoritmo consiste em:
  - Construir um heap máximo;
  - Trocar o primeiro elemento com o último;
  - Reconstruir o heap ignorando o último elemento;
    - Como apenas um elemento foi trocado, basta “descer” tal elemento pelo caminho do seu maior filho;
  - Isso deve ser feito até que o vetor restante tenha tamanho 1;
    - Ou seja, na primeira iteração ignora-se o último elemento;
    - Na segunda o último e o penúltimo
    - e assim por diante

Heapsort

- Para implementar o algoritmo é necessário o uso de dois procedimentos essenciais:
  - Max-heapfy:
    - Recebe um nó i e o coloca em sua posição correta no heap;
  - Build-max-heap:
    - Constroi o heap máximo a partir de um vetor qualquer, aplicando o procedimento max-heapfy em todos nós que não são folhas;

Heapsort

- O procedimento max-heapfy faz os seguintes passos:
  - Verifica qual o maior elemento entre os dois filhos e o pai;
  - Se o maior elemento for um dos dois filhos então troca o filho com o pai;
  - Aplica o procedimento max-heapfy de maneira recursiva para o filho trocado;
- O procedimento build-max-heap faz os seguintes passos:
  - Chama max-heapify para todos os nós não folha de maneira decrescente, iniciando em $\lfloor (n-1)/2 \rfloor$ até 1.
    - Os nós que estão nas posições $0..\lfloor (n-1)/2 \rfloor$, onde n é o número de elementos no vetor, são nós não folha;

Exemplo

- Construir o heap máximo no seguinte vetor

5	2	1	7	6	3	8	9	4

Exemplo

5	2	1	7	6	3	8	9	4
			i				2i+1	2i+2

Troca com o maior filho e aplica max-heapfy nele

5	2	1	9	6	3	8	7	4
			i				2i+1	2i+2

Exemplo

5	2	1	9	6	3	8	7	4
		i			2i+1	2i+2		

Troca com o maior filho e aplica max-heapfy nele

5	2	8	9	6	3	1	7	4
		i			2i+1	2i+2		

Exemplo

5	2	1	9	6	3	8	7	4
	i		2i+1	2i+2				

Troca com o maior filho e aplica max-heapfy nele

5	9	8	2	6	3	1	7	4
	i		2i+1(i')	2i+1	2			

Troca com o maior filho e aplica max-heapfy nele

5	9	8	7	6	3	1	2	4
			i'				2i'+1	2i'+2

Exemplo

5	9	8	7	6	3	1	2	4
i	2i+1	2i+2						

Troca com o maior filho e aplica max-heapfy nele

9	5	8	7	6	3	1	2	4
i	2i+1(i')	2i+2						

Troca com o maior filho e aplica max-heapfy nele

9	7	8	5	6	3	1	2	4
	i'		2i'+1 (i'')	2i'+2			2i''+1	2i''+2

Exemplo

- Heap máximo construído;

9	7	8	5	6	3	1	2	4

Exemplo

- Com o heap máximo construído agora trocamos o primeiro elemento com o último (v[n-1]) e aplica-se o procedimento max-heapfy no primeiro
  - O procedimento max-heapfy passa a ignorar o ultimo elemento;

9	7	8	5	6	3	1	2	4

4	7	8	5	6	3	1	2	9

8	7	4	5	6	3	1	2	9

Exemplo

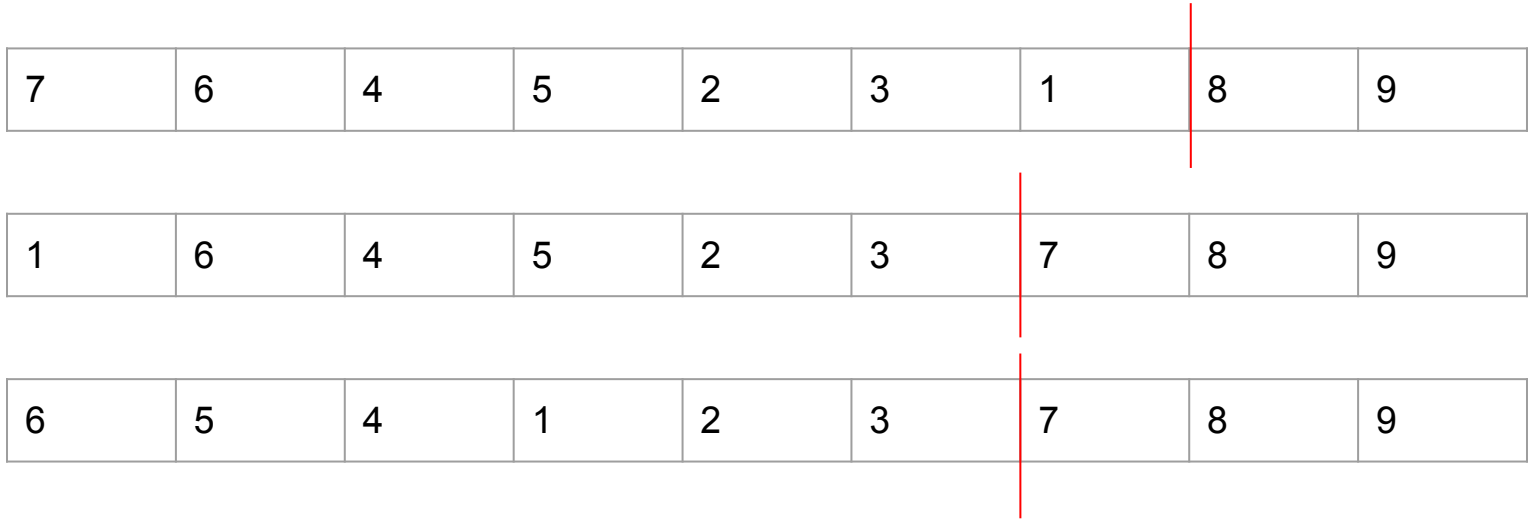
- Assim, o novo heap é composto somente das posições 0 até n-2, então o elemento da primeira posição é trocado com o da última (v[n-2]) e o max-heapfy é aplicado na primeira posição, ignorando os dois ultimos;

8	7	4	5	6	3	1	2	9

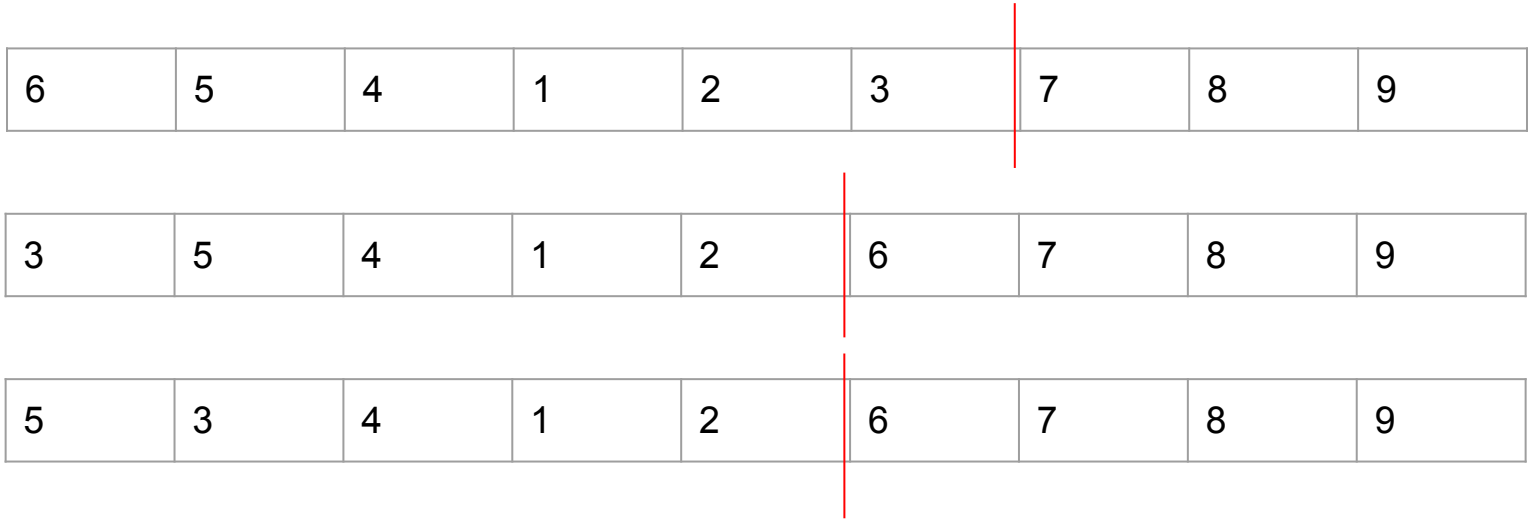
2	7	4	5	6	3	1	8	9

7	6	4	5	2	3	1	8	9

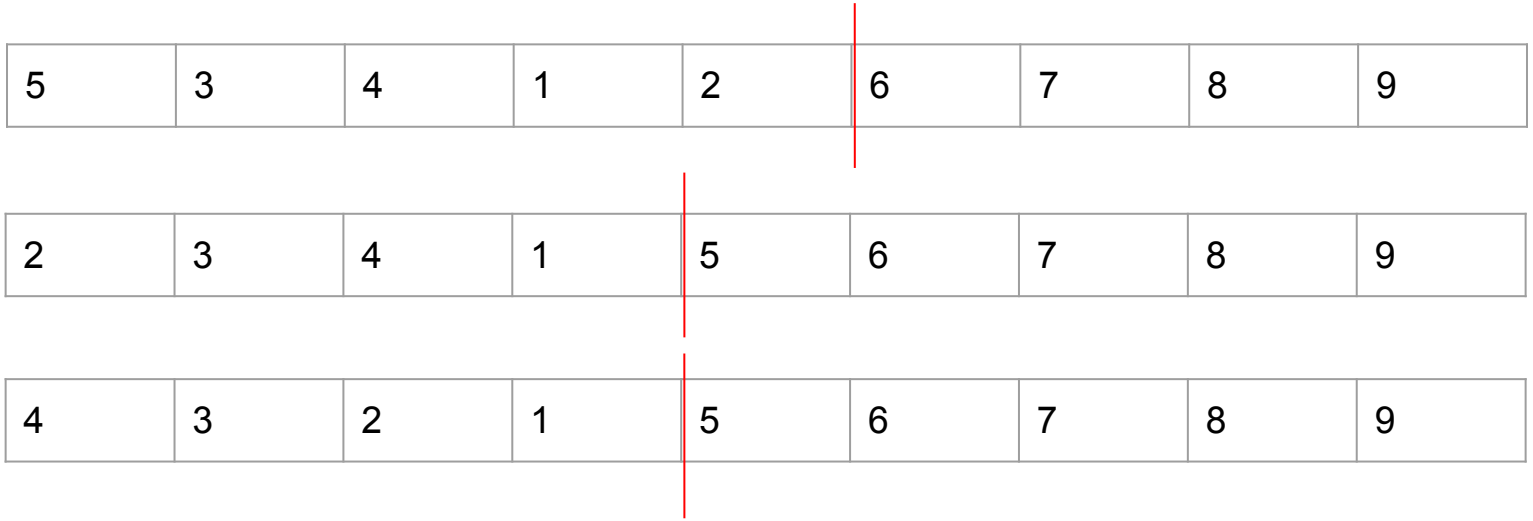
Exemplo



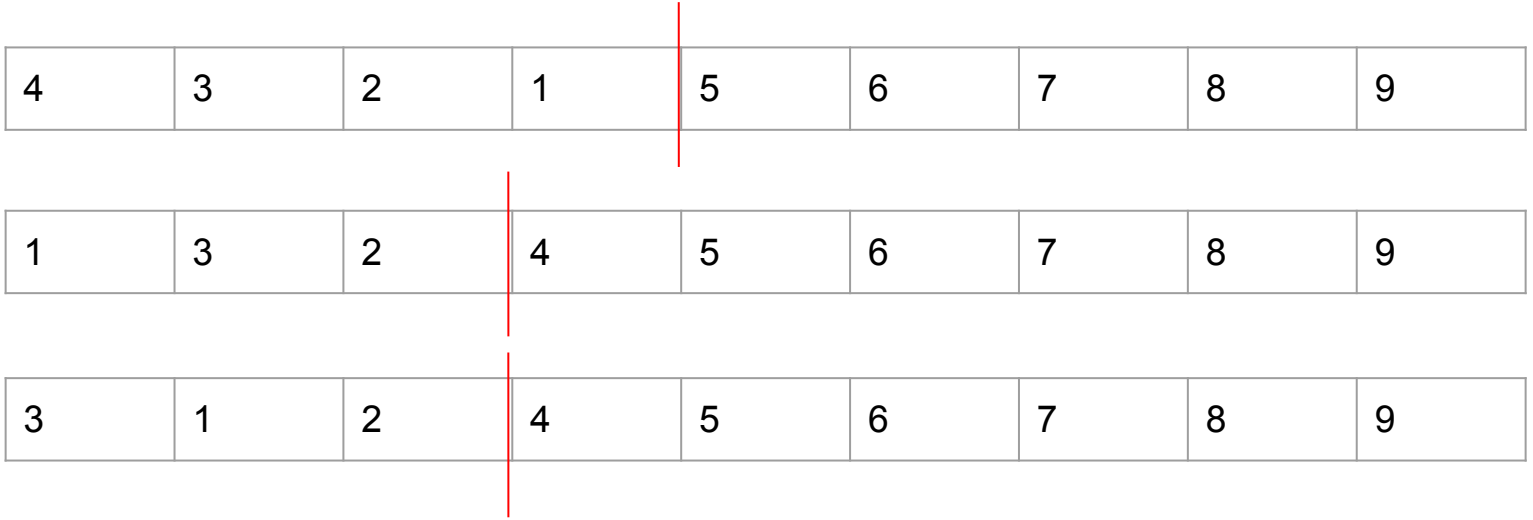
Exemplo



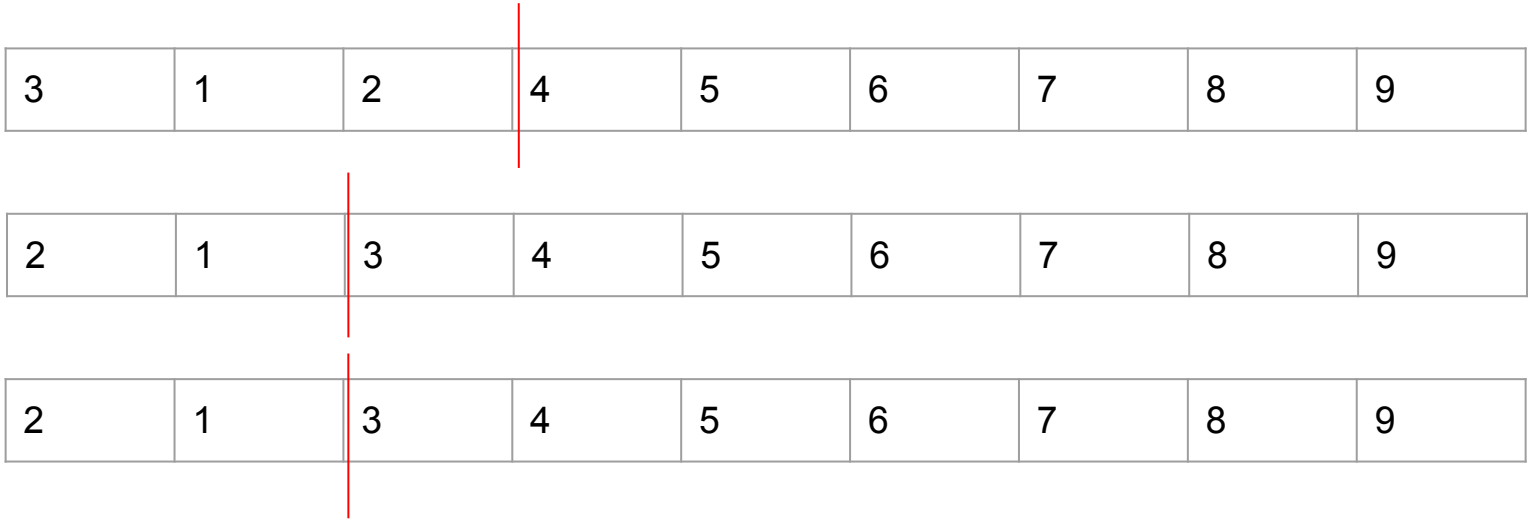
Exemplo



Exemplo

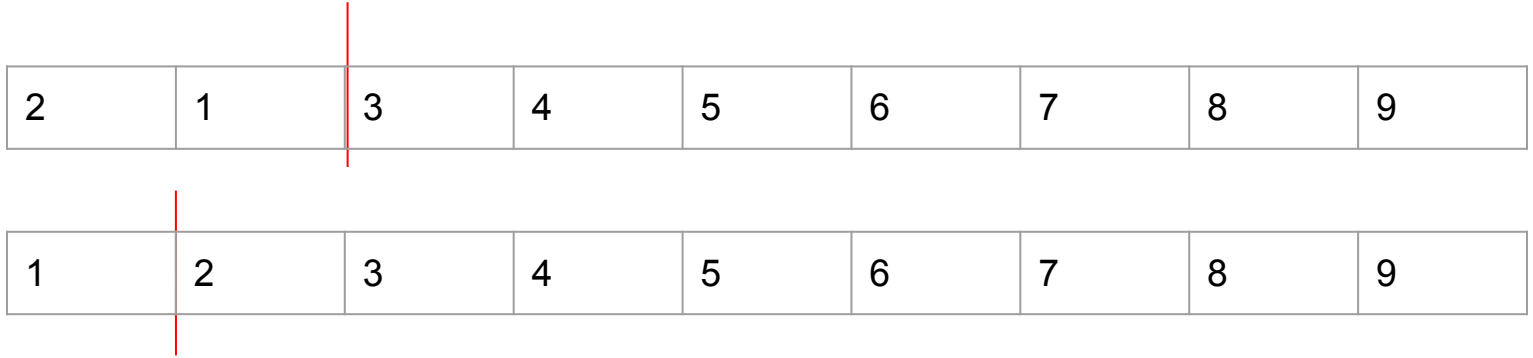


Exemplo



Exemplo

- Quando o vetor tem tamanho 1, ele estará ordenada;



Implementação

- Para implementar o heapsort é iremos implementar os procedimentos:
  - Max-heapfy;
    - Pode ser implementado de maneira recursiva ou iterativa;
  - Build-max-heap;
- Além de um procedimento auxiliar para trocar dois valores:

```
void trocar(int *a,int *b){
    int aux;
    aux=*a;
    *a=*b;
    *b=aux;
}
```

Implementação

- Implementar o procedimento maxheapfy, que recebe um vetor, seu tamanho e o índice do elemento a ser inserido no heap;
  - Esse procedimento assume que todos os valores que estão na subárvore de raiz com índice i formam um heap;

```
void maxheapfy(int v[],int i, int n){
    /*algoritmo*/
}
```

Implementação

- Maxheapfy recursivo

```
void maxheapfy(int v[],int i, int n){
    int fe,fd,maior;
    fe=2*i+1;
    fd=2*i+2;
    maior=i;
    if(fd < n && v[fd]>v[maior]){
        maior=fd;
    }
    if(fe < n && v[fe]>v[maior]){
        maior=fe;
    }
    if(maior!=i){
        trocar(&v[i],&v[maior]);
        maxheapfy(v,maior,n);
    }
}
```

Implementação

- Construir um heap máximo utilizando o procedimento maxheapfy:

```
void buildmaxheap(int v[],int n){
    /*algoritmo*/
}
```

Implementação

- Construir um heap máximo utilizando o procedimento maxheapfy:

```
void buildmaxheap(int v[],int n){
    int i;
    for(i=(n-1)/2;i>=0;i--){
        maxheapfy(v,i,n);
    }
}
```

Implementação

- Implementar o heapsort utilizando os procedimentos buildmaxheap e maxheapfy:

```
void heapsort(int v[], int n){
    int i;
    buildmaxheap(v,n);
    for(i=n-1;i>0;i--){
        trocar(&v[i],&v[0]);
        maxheapfy(v,0,i);
    }
}
```

Heapsort

- Vantagem:
  - Seu pior caso, melhor caso e caso médio são $O(n \log_2 n)$
  - A ordenação é feita *in situ*;
- Desvantagens:
  - Comparado a outros algoritmos como o quicksort, ele é em média duas vezes mais lento;
    - A construção do heap máximo consome muito tempo;
  - O algoritmo não é estável;

Exercícios

1. Mostre o passo a passo da ordenação dos seguintes valores com o algoritmo quicksort:
   a. 1, 6, 5, 25, 15, 3, 9, 7
   b. 6, 3, 2, 4, 1, 9, 0
2. Implemente uma versão iterativa do procedimento maxheapfy.

Referências

ZIVIANI, N.; Projeto de Algoritmos com Implementações em Pascal e C, 3ª Edição, Livraria Pioneira Editora, 1996.