
Estrutura de Dados

— Métodos de Ordenação —
Mergesort

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;

Mergesort

- O algoritmo mergesort usa a estratégia de dividir para conquistar
- Divide o vetor em dois e em seguida ordena cada subvetor com o mergesort de maneira recursiva;
- Sua ideia básica é intercalar subvetores já ordenados;
 - Dessa maneira, se divide os vetores até chegar em vetores de tamanho 1;
 - Após chegar em tamanho 1, vetores são intercalados;

Mergesort

- O método consiste em:
 - Encontrar o meio do vetor;
 - Aplicar o mergesort para ordenar a primeira metade;
 - Aplicar o mergesort para ordenar a segunda metade;
 - intercalar os elementos de ambas metades;
- Ao invés de realmente dividir o vetor usa-se índices para indicar o início, e o fim do subvetor;
- O procedimento para intercalar ambas metades irá criar cópias temporárias dos subvetores para sua execução;

Obs: o mergesort só é aplicado quando o vetor tem mais de um elemento.

Exemplo

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

Exemplo

Aplicando mergesort ao lado esquerdo do vetor de 8 elementos

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

5	2	1	7
---	---	---	---

Exemplo

Aplicando mergesort no lado esquerdo do vetor de 4 elementos

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

5	2	1	7
---	---	---	---

5	2
---	---

Exemplo

Aplicar mergesort no lado esquerdo do vetor de 2 elementos

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

5	2	1	7
---	---	---	---

5	2
---	---

5

Contém só um elemento, então está ordenado

Exemplo

Aplicar mergesort no lado direito do vetor de 2 elementos

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

5	2	1	7
---	---	---	---

5	2
---	---

2

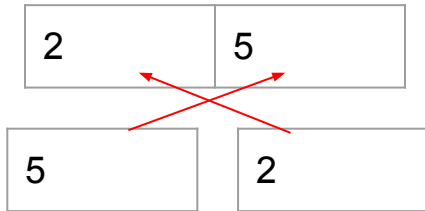
Contém só um elemento, então está ordenado

Exemplo

Fazer merge entre {5} e {2}

2	5	1	7	6	4	8	3
---	---	---	---	---	---	---	---

2	5	1	7
---	---	---	---



Exemplo

Aplicar mergesort do lado direito do vetor de 4 elementos

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

2	5	1	7
---	---	---	---

1	7
---	---

Exemplo

Aplicar mergesort do lado esquerdo do vetor de 2 elementos ($\{1,7\}$)

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

2	5	1	7
---	---	---	---

1	7
---	---

1

Contém só um elemento

Exemplo

Aplicar mergesort do lado direito do vetor de 2 elementos ($\{1,7\}$)

5	2	1	7	6	4	8	3
---	---	---	---	---	---	---	---

2	5	1	7
---	---	---	---

1	7
---	---

7

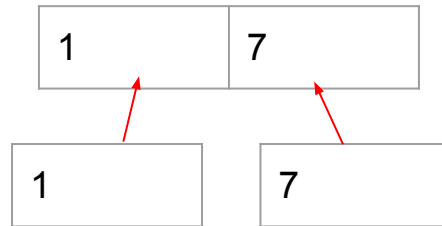
Contém só um elemento

Exemplo

Fazer merge entre {1} e {7}

2	5	1	7	6	4	8	3
---	---	---	---	---	---	---	---

2	5	1	7
---	---	---	---



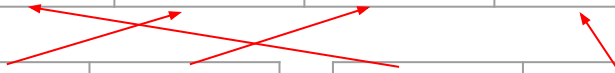
Exemplo

Fazer merge entre $\{2,5\}$ e $\{1,7\}$

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

1	2	5	7
---	---	---	---

2	5	1	7
---	---	---	---



Exemplo

Aplicar mergesort no lado direito do vetor de 8 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

6	4	8	3
---	---	---	---

Exemplo

Aplicar mergesort no lado esquerdo do vetor de 4 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

6	4	8	3
---	---	---	---

6	4
---	---

Exemplo

Aplicar mergesort no lado esquerdo do vetor de 2 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

6	4	8	3
---	---	---	---

6	4
---	---

6

Contém somente um elemento

Exemplo

Aplicar mergesort no lado direito do vetor de 2 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

6	4	8	3
---	---	---	---

6	4
---	---

4

Contém somente um elemento

Exemplo

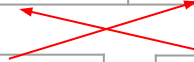
Fazer merge entre {6} e {4}

1	2	5	7	4	6	8	3
---	---	---	---	---	---	---	---

4	6	8	3
---	---	---	---

4	6
---	---

6	4
---	---



Exemplo

Aplicar mergesort no lado direito do vetor de 4 elementos

1	2	5	7	4	6	8	3
---	---	---	---	---	---	---	---

4	6	8	3
---	---	---	---

8	3
---	---

Exemplo

Aplicar mergesort no lado esquerdo do vetor de 2 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

4	6	8	3
---	---	---	---

8	3
---	---

8

Contém somente um elemento

Exemplo

Aplicar mergesort no lado esquerdo do vetor de 2 elementos

1	2	5	7	6	4	8	3
---	---	---	---	---	---	---	---

4	6	8	3
---	---	---	---

8	3
---	---

3

Contém somente um elemento

Exemplo

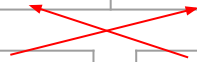
Fazer merge entre {8} e {3}

1	2	5	7	6	4	3	8
---	---	---	---	---	---	---	---

4	6	3	8
---	---	---	---

3	8
---	---

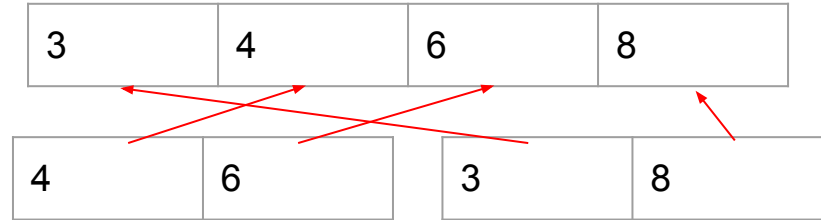
8	3
---	---



Exemplo

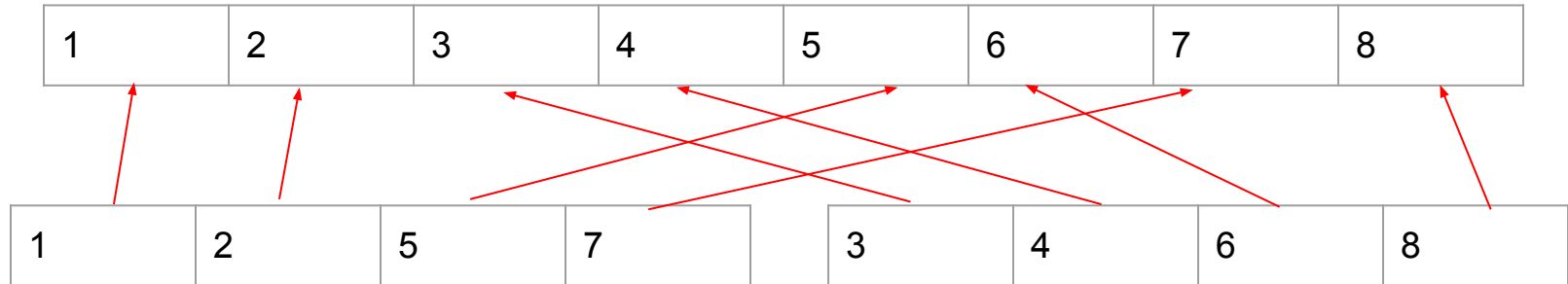
Fazer merge entre $\{4,6\}$ e $\{3,8\}$

1	2	5	7	3	4	6	8
---	---	---	---	---	---	---	---



Exemplo

Fazer merge entre $\{1,2,5,7\}$ e $\{3,4,6,8\}$



Implementação


- Para facilitar a implementação do mergesort, pode-se dividi-lá em duas partes:
 - Fazer a intercalação (merge) nos vetores;
 - Construir o procedimento recursivo do mergesort utilizando a intercalação;

Implementação

- A ideia do procedimento de intercalação é a seguinte:
 - Receber o vetor, um índice inicial, um índice final, e um índice que represente o meio entre o início e o fim;
 - Criar dois vetores auxiliares;
 - Um para armazenar os elementos das posições do índice inicial até o índice do meio;
 - Outro para armazenar os elementos das posições do índice do meio +1 até a posição do índice final;
 - Fazer o vetor “principal” receber a intercalação de ambos;

Implementação

	ini			m				fim
elemento	1	5	7	9	2	3	4	10
índice	0	1	2	3	4	5	6	7


Vetor 1 Vetor 2

Implementação

- Procedimento de intercalar

```
void merge(int v[], int ini, int meio, int fim){  
    /*algoritmo*/  
}
```

Dica: Para criar um vetor de inteiros com n posições pode-se utilizar a função malloc da seguinte maneira:

```
int *vet = (int*) malloc (n*sizeof(int));  
//os elementos do vetor vet poderão ser acessados como vetor[i];
```

É importante liberar a memória alocada para vet após usá-la

```
free(vet);
```

Implementação

```
1. void merge(int v[], int ini,int meio, int fim){
2.     int tv1,tv2,i,j,k, *v1,*v2;
3.     tv1=meio-ini+1;
4.     tv2=fim-meio;
5.     v1=(int*)malloc(tv1*sizeof(int));
6.     v2=(int*)malloc(tv2*sizeof(int));
7.     j=ini;
8.     for(i=0;i<tv1;i++){
9.         v1[i]=v[j]; j++;
10.    }
11.    for(i=0;i<tv2;i++){
12.        v2[i]=v[j]; j++;
13.    }
14.    i=j=0;
15.    k=ini;
```

```
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32. }
```

```
while(i<tv1 && j<tv2){
    if(v1[i]<v2[j]){
        v[k]=v1[i]; i++;
    }else{
        v[k]=v2[j]; j++;
    }
    k++;
}
while(j<tv2){
    v[k]=v2[j]; k++; j++;
}
while(i<tv1){
    v[k]=v1[i]; k++; i++;
}
free(v1);
free(v2);
```

Implementação

- Considerando a função merge, implemente o algoritmo mergesort;
 - Lembre-se que ini é o **índice** do primeiro elemento, e que fim é o **índice** do último elemento;

```
void mergesort(int v[],int ini, int fim){  
    /*algoritmo*/  
}
```


Implementação

- Considerando a função merge, implemente o algoritmo mergesort;
 - Lembre-se que ini é o **índice** do primeiro elemento, e que fim é o **índice** do último elemento;

```
void mergesort(int v[],int ini, int fim){
    int meio=(ini+fim)/2;
    if(ini<fim){
        mergesort(v,ini,meio);
        mergesort(v,meio+1,fim);
        merge(v,ini,meio,fim);
    }
}
```

Mergesort

- Vantagem:
 - O pior, melhor e caso médio tem o mesmo tempo assintótico;
 - Tempo de execução na ordem de $O(n \log_2 n)$;
- Desvantagem:
 - Consumo de memória é alto;
 - Usa vetores adicionais para intercalar os elementos;

Exercícios

1. Mostre o passo a passo da ordenação dos seguintes valores com o algoritmo mergesort:
 - a. 1, 6, 5, 25, 15, 3, 9, 7
 - b. 6, 3, 2, 4, 1, 9, 0