INSTITUTO FEDERAL
RIO GRANDE DO SUL

Estrutura de Dados II

Algoritmos de Ordenação:

- Ordenação QuickSort
- MergeSort

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

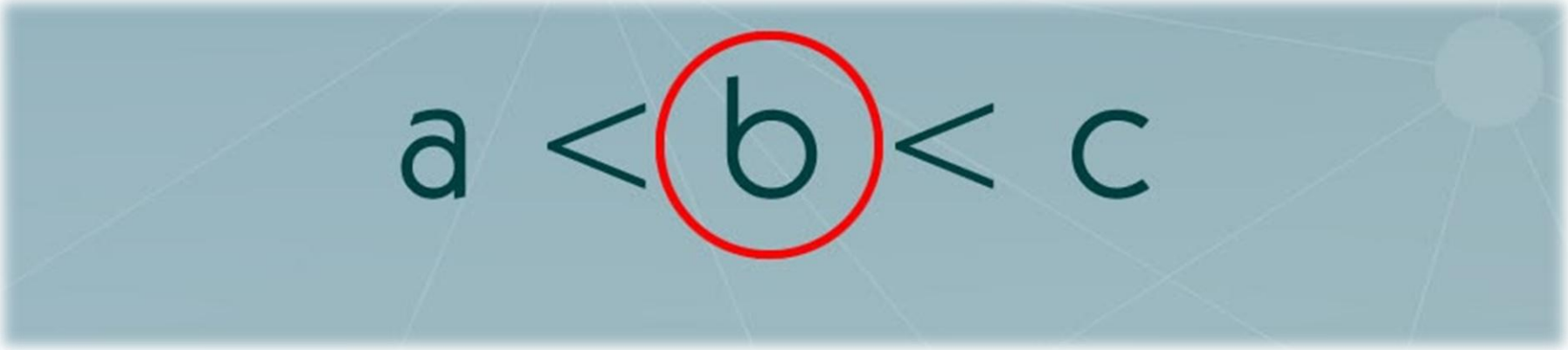
INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmos de Ordenação

Alguns Algoritmos de Ordenação:

- Bubblesort
- Ordenação por Contagem
- Ordenação por Inserção
- Ordenação por Seleção
- **MergeSort**
- **QuickSort**

ORDENAÇÃO QUICKSORT



INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação Quicksort

O Algoritmo Quicksort foi criado em 1960. É considerado um dos métodos de ordenação mais rápido para uma ampla variedade de situações, devido a complexidade $O(n^2)$ no pior caso e $O(n \log n)$ no melhor e médio caso.

O Algoritmo é implementado em várias linguagens, largamente utilizado/customizado em projetos de linguagens de programação o que o torna um dos mais utilizados.

A lógica do funcionamento do algoritmo baseia-se em uma rotina fundamental cujo nome é ***particionamento***.

Particionar significa escolher um número qualquer presente no conjunto de dados, chamado de ***pivot***, e colocá-lo em uma posição tal que todos os elementos à esquerda são menores ou iguais e todos os elementos à direita são maiores.

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação Quicksort

O Quicksort também adota a estratégia de divisão e conquista.

Os passos são:

a) Escolha um elemento da lista, denominado ***pivot (pivô)***;

b) Rearranje a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele. Ao fim do processo o pivô estará em sua posição final e haverá duas sublistas não ordenadas. **Essa operação é denominada *partição***;

c) Recursivamente ordene a sublista dos elementos menores e a sublista dos elementos maiores;

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação Quicksort

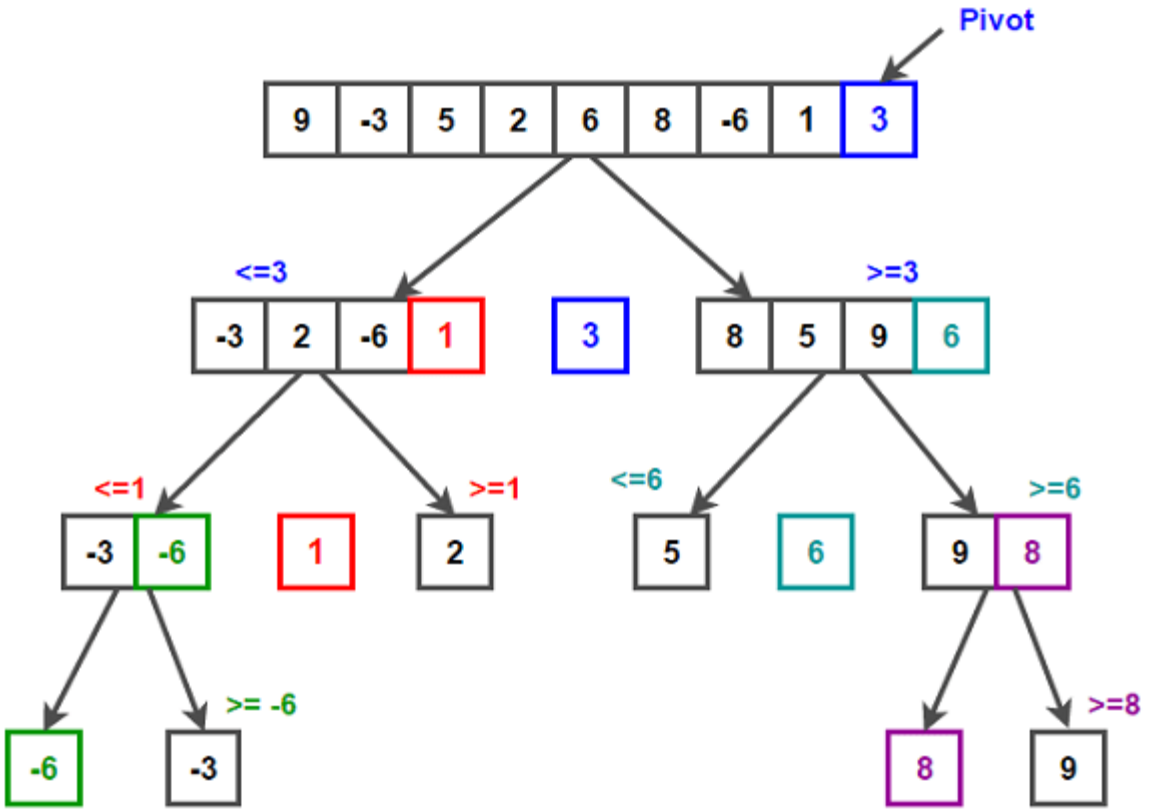
A base da recursão são as listas de tamanho zero ou um, que estão sempre ordenadas.

O processo é finito, pois a cada iteração pelo menos um elemento é posto em sua posição final e não será mais manipulado na iteração seguinte.

Apesar de estar na mesma classe de complexidade do **Merge Sort**, há experimentos que demonstram que **o Quick Sort em seu melhor caso e caso médio é por volta de 3x mais eficiente que o Merge Sort**, porque ele contém constantes menores.

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Ordenação Quicksort



INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação Quicksort

```
...

main ()
{

//Inicialização do Vetor
   int v[5]= {1,7,4,3,5}, n=5;
   int i, j = 0,  aux;

//Algoritmo de Ordenação
 quickSort(v,0,4);

 //Laço de impressão do Vetor
   for (int q=0; q<5; q++)
   {
        printf("%d \n",v[q]);
    }
...
```

```
void swap(int* a, int* b) {
  int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;}

int partition(int vec[], int left, int right) {
  int i, j;
  i = left;
  for (j = left + 1; j <= right; ++j) {
    if (vec[j] < vec[left]) {
      ++i;
      swap(&vec[i], &vec[j]);
    }
  }
  swap(&vec[left], &vec[i]);
  return i;}

void quickSort(int vec[], int left, int right) {
  int r;
   if (right > left) {
    r = partition(vec, left, right);
    quickSort(vec, left, r - 1);
    quickSort(vec, r + 1, right);  }
}
```

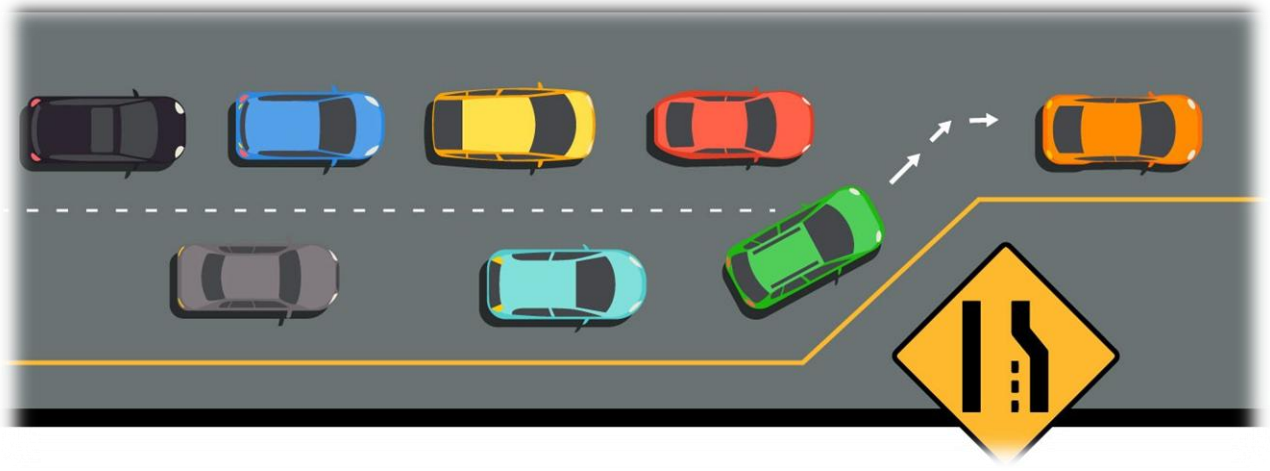
INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação QuickSort

- Quantas **chamadas recursivas foram** necessárias para ordenar um conjunto de 5 elementos: {1,3,5,9,10}?

- Quantas **chamadas recursivas foram** são necessárias para ordenar um conjunto de 5 elementos: {10,9,5,3,1}?

ORDENAÇÃO MERGESORT



INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de ordenação Mergesort

Esse é um algoritmo que segue uma **técnica de dividir**, ordenar e juntar (**dividir para conquistar**).

A eficiência é a mesma para melhor, pior e caso médio. Independentemente de como os dados do array estão organizados a ordenação será eficaz. ($O(n\log_2 n)$)

O conceito para que ele funcione é o seguinte:

- Ordenar uma estrutura significa ordenar várias subestruturas internas já ordenadas.
- Caso essas estruturas não estejam ordenadas, basta ordená-las pelo mesmo método (ordenar suas subestruturas internas... até o infinito).
- Para Ordenar Subestruturas, basta dividir a estrutura atual e fazer uma chamada recursiva.

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de ordenação Mergesort

Os três passos do algoritmos que se aplicam ao ***mergesort*** são:

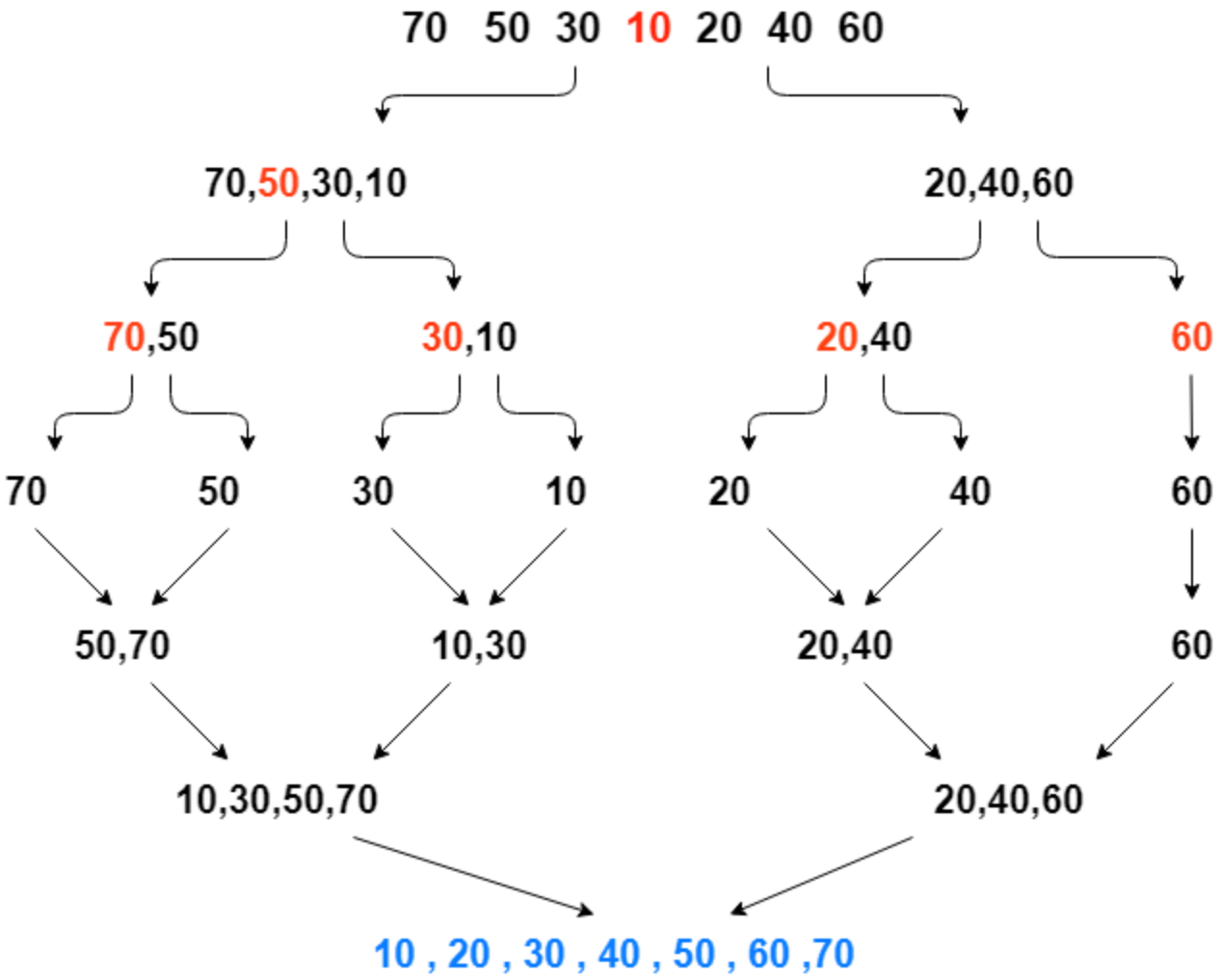
- **Dividir**: Calcula o ponto médio do sub-arranjo.
- **Conquistar**: Recursivamente resolve dois subproblemas, cada um de tamanho n/2, o que contribui com para o tempo de execução.
- **Combinar**: Unir os sub-arranjos em um único conjunto ordenado.

Exemplo:

- Ordenar 70, 50, 30, 10, 20, 40, 60... o algoritmo irá dividir em pares: {70,50}, {30,10}, {20,40}, {60}...
- Ordenados {50,70}, {10,30}, {20,40}, {60}.
- Depois ir fazendo o merge desses dados, ou seja, juntando-os em dois pares ordenados: {10,30,50,70}, {20,40,60},
- Depois juntar novamente: 10, 20, 30, 40,50, 60, 70.
- No final do algoritmo a seqüência inicial está ordenada a partir de divisões.

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Ordenação Mergesort



Mergesort

```
void mergeSort(int vec[], int vecSize)
{
  int mid;
  if (vecSize > 1)
  {
    mid = vecSize / 2;
    mergeSort(vec, mid);
    mergeSort(vec + mid, vecSize - mid);
    merge(vec, vecSize);
  }
}


main ()
{
//Inicializaçã do Vetor
    int v[5]= {1,7,4,3,5}, n=5;
         int i, j = 0,  aux;

//Algoritmo de Ordenação
 mergeSort(v,n);

//Laço de impressão do Vetor
         for (int q=0; q<5; q++)
         {
             printf("%d \n",v[q]);
         }
}
```

```
//Kernell do Algoritmo
void merge(int vec[], int vecSize) {
  int mid;
  int i, j, k;
  int* tmp;
  tmp = (int*) malloc(vecSize * sizeof(int));
  if (tmp == NULL) {
    exit(1);  }
  mid = vecSize / 2;
  i = 0;
  j = mid;
  k = 0;
  while (i < mid && j < vecSize) {
    if (vec[i] < vec[j]) {
      tmp[k] = vec[i];
      ++i;    }
    else {
      tmp[k] = vec[j];
      ++j;    }
    ++k;  }
  if (i == mid) {
    while (j < vecSize) {
      tmp[k] = vec[j];
      ++j;
      ++k;    }  }
  else {
    while (i < mid) {
      tmp[k] = vec[i];
      ++i;
      ++k;    }  }
    for (i = 0; i < vecSize; ++i) {
    vec[i] = tmp[i];  }
  free(tmp);}
```

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Algoritmo de Ordenação Merge Sort

- Quantas **chamadas recursivas foram** são necessárias para ordenar um conjunto de 5 elementos: {1,3,5,9,10}?

- Quantas **chamadas recursivas foram** são necessárias para ordenar um conjunto de 5 elementos: {10,9,5,3,1}?

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Laboratório

- Execute o Programa “Gera_RAND” para criar um arquivo com 100.000 número randômicos.
- Altere o Programa do Algoritmo de QuickSort para que leia o arquivo e gere um arquivo ordenado.
- Altere o Programa do Algoritmo MergeSort para que leia o arquivo e gere um arquivo ordenado.
- **Qual dos dois foi mais rápido?**