Estruturas de Dados
Ordenação de dados

Prof. Tales Nereu Bogoni
tales@unemat.br

UNEMAT
Universidade do Estado de Mato Grosso

Ordenação de dados

UNEMAT
Universidade do Estado de Mato Grosso

- Ordenar é o processo de rearranjar um conjunto de objetos em uma ordem específica
- O objetivo da ordenação dos dados é facilitar a recuperação posterior de elementos do conjunto ordenado
- Os dados só devem ser ordenados se o custo computacional da ordenação é menor do que o custo para recuperação dos dados de forma não ordenada
- A ordenação dos dados é sempre feita com base em dos campos da estrutura de dados, ou um conjunto deles agrupados, chamado de campo chave

Classificação dos métodos de ordenação

UNEMAT
Universidade do Estado de Mato Grosso

- Interna
  - Conjunto de dados relativamente pequeno
  - Como medidas de complexidade deve ser analisada a quantidade de comparações e a quantidade permuta dos dados
  - Permite ir diretamente para uma determinada informação, desde que saiba sua posição
- Externa
  - Grandes volumes de dados (não cabem na memória do computador)
  - Registro armazenados em dispositivos de armazenamento auxiliar
  - Os dados são acessados sequencialmente

Métodos de ordenação interna

UNEMAT
Universidade do Estado de Mato Grosso

- Métodos simples ou diretos
  - Algoritmos pequenos e de fácil compreensão
  - Pequena quantidade de dados
  - Algoritmos executados poucas vezes
  - Mais lentos que os métodos eficientes
- Métodos eficientes
  - Executam menos comparações e substituições que os métodos simples
  - Algoritmos mais complexos
  - Quantidade maior de dados
  - Algoritmos executados várias vezes

Estabilidade de algoritmos

UNEMAT
Universidade do Estado de Mato Grosso

- Um algoritmo de ordenação diz-se **estável** se preserva a ordem de registros de chaves iguais. Isto é, se tais registros aparecem na sequência ordenada na mesma ordem em que estão na sequência inicial.
- **Algoritmos estáveis**
  - Bubble sort
  - Insertion sort
  - Merge sort
- **Algoritmos não estáveis**
  - Selection Sort (Depende da implementação)
  - Quicksort
  - Shellsort

Exemplos de métodos de ordenação

UNEMAT
Universidade do Estado de Mato Grosso

- Métodos simples
  - Ordenação por permutação – *Bubble sort*
  - Ordenação por inserção – *Insertion sort*
  - Ordenação por seleção – *Selection sort*
- Métodos eficientes
  - Ordenação por inserção com incrementos decrescentes – *Shell sort*
  - Ordenação por particionamento e junção de dados – *Merge sort*
  - Ordenação por particionamento de dados – *Quick sort*

Vídeos demonstrativos: https://www.geeksforgeeks.org/bubble-sort/

Bubble sort

vetor

tamanho

```
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)
      for (j = 0; j < n-i-1; j++)
         if (arr[j] > arr[j+1]) {
            int aux = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = aux;
         }
}
```

Variáveis de controle
i -> elementos não ordenados do vetor
j -> comparação com o restante do vetor

Verifica se o elemento não está ordenado

Permuta os dados
Coloca em ordem

Demonstração: https://www.geeksforgeeks.org/bubble-sort/

Insertion sort

UNEMAT
Universidade do Estado de Mato Grosso

vetor

tamanho

```
void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
      key = arr[i];
      j = i-1;
      while (j >= 0 && arr[j] > key)
      {
         arr[j+1] = arr[j];
         j = j-1;
      }
      arr[j+1] = key;
   }
}
```

Variáveis de controle
i -> posição no sub-vetor não ordenado
Key -> valor a ser ordenado
j -> posição no sub-vetor ordenado

Deslocamento no sub-vetor não ordenado

Guarda a chave

Encontra a posição da chave no sub-vetor ordenado

Permuta os dados
Coloca em ordem

Demonstração: https://www.geeksforgeeks.org/insertion-sort/

Selection sort

vetor

tamanho

```
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    for (i = 0; i < n-1; i++)
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        swap(&arr[min_idx], &arr[i]);
    }
}
```

Variáveis de controle

Procura o menor elemento do vetor

Troca o menor elemento com o primeiro desordenado

Demonstração: https://www.geeksforgeeks.org/selection-sort/

Shell sort

vetor

tamanho

```
int shellSort(int arr[], int n)
{
   for (int gap = n/2; gap > 0; gap /= 2)
   {
      for (int i = gap; i < n; i += 1)
      {
         int temp = arr[i];
         int j;
         for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
            arr[j] = arr[j - gap];

         arr[j] = temp;
      }
   }
   return 0;
}
```

Separa o valor de um elemento

Percorre o vetor com passo do tamanho do gap

Coloca no ultimo elemento o valor separado

Demonstração: https://www.geeksforgeeks.org/shellsort/

Merge sort (1)

UNEMAT
Universidade do Estado de Mato Grosso

vetor

Posição inicial e final

```
void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
            int m = l+(r-l)/2;

             mergeSort(arr, l, m);
             mergeSort(arr, m+1, r);

             merge(arr, l, m, r);
    }
}
```

Divide o vetor no meio

Executa para as duas partes do vetor

Une os vetores

Demonstração: https://www.geeksforgeeks.org/merge-sort/

Merge sort (2)

UNEMAT
Universidade do Estado de Mato Grosso

vetor

Posição inicial e final

```
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    i = 0;
    j = 0;
    k = l;
```

Vetores Temporários

Variáveis de Controle do merge

```
    while (i < n1 && j < n2)
    {   if (L[i] <= R[j])     {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1)  {
        arr[k] = L[i];        i++;        k++;   }

    while (j < n2) {  arr[k] = R[j];   j++;   k++;}
}
```

Ordena os dados

Copia a parte que sobra dos vetores

Quick sort (1)

UNEMAT
Universidade do Estado de Mato Grosso

vetor

Posição inicial e final

```
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

Critério de Parada

Divide o vetor (outro algoritmo)

Ordena as duas partes do vetor

Demonstração: https://www.geeksforgeeks.org/quick-sort/

Quick sort (2)

UNEMAT
Universidade do Estado de Mato Grosso

vetor

Posição inicial e final

```
int partition (int arr[], int low, int high)
{
        int pivot = arr[high];
        int i = (low - 1);
    for (int j = low; j <= high- 1; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
         }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
```

Define o pivô

Primeiro elemento a ser ordenado

Percorre o vetor

Ordena em relação ao pivô

Troca os elementos