1 - Introdução

Para fazer operações de busca em um array é interessante que os seus elementos estejam ordenados em ordem crescente ou descrecente. O desafio consiste em encontrar o algoritmo que seja capaz de ordenar os elementos de um array consumindo pouco recurso computacional (memória e processador), pois um bom algoritmo de ordenação:

- Não deve criar um array auxiliar, pois o array pode ter bilhões de elementos e isso causaria sobrecarga na memória;
- Não deve criar loops excessivos, pois isso causaria sobrecarga do processador.

O problema de ordenação não se se aplica somente a arrays, mas também a listas encadeadas.

A ordenação consiste em permutar (rearranjar) os elementos de um array de tal modo que eles fiquem em ordem crescente ou decrescente. A seguir serão apresentados três algoritmos simples de ordenação.

2 - Algoritmo de Ordenação por Inserção

O algoritmo de ordenação por inserção (insertion sort) é aplicado a uma estrutura do tipo array ou lista (https://pt.wikipedia.org/wiki/Insertion_sort).

A Figura 1 mostra uma implementação do algoritmo. O subarray a esquerda de i pode ser percorrido sucessivas vezes trocando os valores de lugar, isso faz com que o algoritmo tenha as seguintes características:

- Não necessita de um array auxiliar para ser implementado;
- Útil para arrays com poucos elementos;
- Muito lento para arrays com muitos elementos;
- Ele é estável – não muda a ordem relativa de elementos com valores iguais;
- Ordem de complexidade:
  - Melhor caso: O(n) quando o array está ordenado;
  - Médio caso: $O(n^2/4)$ quando o array tem valores aleatórios sem ordem de classificação (crescente ou decrescente);
  - Pior caso: $O(n^2)$ quando o array está em ordem inversa daquela que deseja ordenar.

```
public void insercao(int[] v){
   int aux, j;
   /* o loop se repete i-1 vezes */
   for (int i = 1; i < v.length; i++) {
      aux = v[i];
      /* pode chegar a percorrer todo o subarray v[0..i-1] */
      for (j = i-1; j >= 0 && v[j] > aux; j--){
         v[j+1] = v[j];
      }
      v[j+1] = aux;
   }
}
```

Figura 1 – Método que implementa o algoritmo de ordenação por inserção.

3 - Algoritmo de Ordenação por Seleção

O algoritmo de ordenação por seleção (selection sort) é um algoritmo de ordenação baseado em se passar sempre o menor valor do subarray em análise para o início (https://pt.wikipedia.org/wiki/Selection_sort), por exemplo:

- 1ª iteração: a posição v[0] irá receber o menor valor do subarray v[0..n];
- 2ª iteração: a posição v[1] irá receber o menor valor do subarray v[1..n];
- 3ª iteração: a posição v[2] irá receber o menor valor do subarray v[2..n];
- nª iteração: a posição v[n-1] irá receber o menor valor do subarray v[n-1..n].

A Figura 2 mostra uma implementação do algoritmo.

Características do algoritmo:

- Não necessita de um array auxiliar para ser implementado;
- Ele não é estável – muda a ordem relativa de elementos com valores iguais;
- Útil para arrays com poucos elementos;
- Muito lento para arrays com muitos elementos;
- Ordem de complexidade: será sempre O(n²) em todos os casos.

```
public void selecao(int[] v){
   int menor, aux, j;
   for(int i = 0; i < v.length-1; i++){
      menor = i;
      /* identifica a posição do menor valor no subarray [i..v.length] */
      for(j= i+1; j < v.length; j++){
         if (v[j] < v[menor]){
            menor = j; /* índice do menor valor */
         }
      }
      /* troca o menor valor com a posição i */
      aux = v[i];
      v[i] = v[menor];
      v[menor] = aux;
   }
}
```

Figura 2 – Método que implementa o algoritmo de ordenação por seleção.

4 - Algoritmo de Ordenação Buble Sort

O algoritmo de ordenação por bolha (bubble sort) é um algoritmo cuja a ideia é percorrer o array diversas vezes, a cada iteração o maior valor flutua para a última posição do subarray (https://pt.wikipedia.org/wiki/Bubble_sort). O movimento de flutuar lembra a forma como as bolhas de ar sobem até a superfície da água, daí vem o nome do algoritmo, mas as trocas são par a par, ou seja, o maior valor irá trocar de posição sucessivamente até atingir o final do subarray. Por exemplo:

- 1ª iteração: a posição v[n] irá receber o maior valor do subarray v[0..n];
- 2ª iteração: a posição v[n-1] irá receber o menor valor do subarray v[0..n-1];
- 3ª iteração: a posição v[n-2] irá receber o maior valor do subarray v[0..n-2];
- nª iteração: a posição v[1] irá receber o maior valor do subarray v[0..1].

A Figura 3 mostra uma implementação do algoritmo, neste exemplo usou-se a variável trocado para identificar que não existe a necessidade de novas iterações, pois não foram trocados elementos na iteração anterior.

Características do algoritmo:

- Não necessita de um array auxiliar para ser implementado;
- Útil para arrays com poucos elementos;
- Muito lento para arrays com muitos elementos;
- Ordem de complexidade:
  - Melhor caso: O(n) quando o array está ordenado;
  - Médio caso: $O(n^2)$ quando o array tem valores aleatórios sem ordem de classificação (crescente ou decrescente);
  - Pior caso: $O(n^2)$ quando o array está em ordem inversa, daquela que deseja ordenar.

```
public void bolha(int[] v){
   boolean trocado = true;
   /* percorre todo o array n-vezes ou até não haver trocas */
   for(int aux, i, fim = v.length-1; fim > 0 && trocado; fim--){
      trocado = false;
      /* a cada iteração o maior valor vai parar na última posição do subarray [0..fim] */
      for(i = 0; i < fim; i++){
         if( v[i] > v[i+1] ){
            aux = v[i]; /* troca os pares de valores */
            v[i] = v[i+1];
            v[i+1] = aux;
            trocado = true;
         }
      }
   }
}
```

Figura 3 – Método que implementa o algoritmo de ordenação bubble sort.

5 - Exercícios

1 – Programar um método que ordena os elementos do array em ordem decrescente usando o algoritmo de inserção.

2 – Programar um método que ordena os elementos do array em ordem decrescente usando o algoritmo de seleção.

3 – Programar um método que ordena os elementos do array em ordem decrescente usando o algoritmo buble sort.

4 – Programar um método que ordena os elementos de um array de String, pode ser usado o algoritmo de inserção, de seleção ou bubble sort.