Algoritimos (simples) de Ordenação II

Técnicas de Programação - 2025/02

Última atualização: 01/09/2025 10:07

Em um primeiro momento, estudaremos dois algoritmos simples de ordenação: *Bubble Sort* e *Selection Sort*. Um dos nossos objetivos aqui é contar o número de **comparações** feitas por esses algoritmos, sempre considerando o **pior caso**.

Selection Sort

Vamos agora desenvolver um algoritmo a partir de uma ideia de alto nível:

1. Comece encontrando o menor elemento do array. Troque ele de posição com o elemento na posição `0`.
2. Faça o mesmo para a posição `1`, mas procurando o mínimo agora a partir do elemento `1`. Troque esse elemento com o da posição `1`
3. Repita isso para todas as posições do array, sempre tomando cuidado para pegar o mínimo da porção à direita da posição atual.

Vamos agora simular essa ideia para entendê-la melhor.

Exercício: Simule a ideia acima para o array `A = [3, 5, 4, 2]`. Você deve colocar uma linha para cada iteração do algoritmo, ou seja, cada vez que é feita uma troca entre o elemento atual e o menor da porção que resta do array. Em cada linha escreva os índices que são feitas a troca e o estado de `A` após a troca ser feita. A primeira linha já está preenchida.

Iteração	Índice Menor	A
0	3	$A = [2, 5, 4, 3]$

Exercício: Faça agora o mesmo para o array $A = [5, 6, 8, 11, 13, 15]$.

Iteração	Índice Menor	A

Exercício: Faça uma última simulação para $A = [3, 7, 12, 0, -1, 4, 8]$. Agora você já deve estar pegando o jeito!

Iteração	Índice Menor	A

O vídeo abaixo mostra uma simulação (dançada) para o array $[3, 0, 1, 8, 7, 2, 5, 4, 9, 6]$

Figure 1: Simulação do algoritmo *Selection Sort*

Revendo nossa simulação, podemos separa o algoritmo em **duas partes**:

1. Calcular o mínimo para cada iteração i. Só consideramos os elementos a partir do índice i
2. Trocar i com o mínimo e fazer até $N - 1$ iterações do algoritmo.

Comece pelo passo 1, escrevendo no espaço abaixo.

Algorithm 1: MenorDoSubArray

Input : array A, int i
Output: int
Result: Encontra o índice com menor valor no intervalo $[i, N)$

Agora faça a repetição para todas as posições do algoritmo acima para todas as posições. Não se esqueça de realizar a troca do elemento na posição atual com o menor daquele subarray.

Algorithm 2: SelectionSort

Input : array A, int i
Output: int
Result: Ordena o array A em ordem crescente.

Temos aqui um algoritmo que chama outro. Vamos construir nosso argumento pouco a pouco.

Exercício: Dado um array de tamanho N, quantas vezes o loop do algoritmo *SelectionSort* roda? Logo, quantas vezes *MenorDoSubArray* é chamada?

Exercício: Agora vamos olhar para o algoritmo *MenorDoSubArray*. Para um array de tamanho N e passando $i = 4$, quantas iterações são feitas dentro do loop? E para $i = 6$? Escreve o número de iterações em termos de N e i genéricos.

Desafio: Agora calcule quantas operações são feitas no total para um array de tamanho N. Junte ambos os exercícios acima em uma expressão só e calcule seu valor em termos de N.