Projeto Da Disciplina

Anápolis, 05 de março de 2025

Nome da disciplina: Arvores e Grafos

Aluno: Matheus Marques Portela

Lista de exercícios

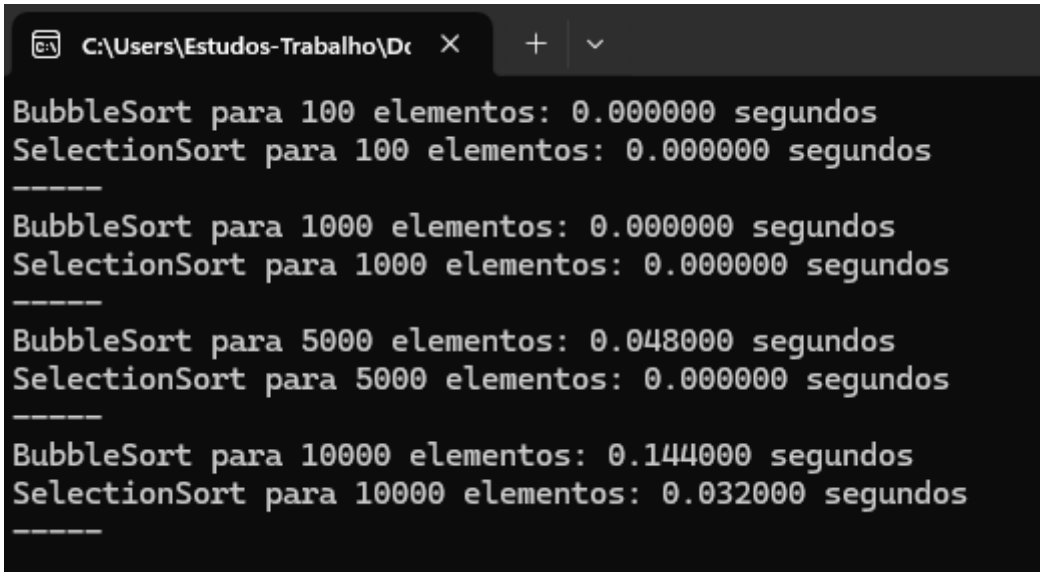
1) O método bubble sort ele percorre um array repetidamente fazendo a comparação dos elementos adjacentes e trocando-os se estiverem na ordem errada.
   Já o método Selection Sort ele divide um array em duas partes aonde seleciona o menor elemento de uma sequência e o coloca na primeira posição. Esse processo é repetido até que o array inteiro esteja ordenado.
   Comparando os dois modelos de ordenação o Selection Sort realiza menos trocas comparado com o bubble sort.

4) No método Bubble Sort temos a seguinte complexidade:
   - Melhor Caso: **O(n)** – Quando o array já está ordenado. Nesta complexidade o tempo de execução cresce linearmente com o tamanho da entrada;
   - Pior caso: **O(n ^ 2)** – Quando o array está em ordem reversa. Está complexidade é típica para algoritmos com loops aninhados;
   - Caso médio: **O(n ^ 2)** – Para a maioria das entradas aleatória.

   Já o método Selection Sort temos a seguinte complexidade:

   - **O(n ^ 2):** Essa complexidade está presente em todos os casos do modelo selection, ele será mesmo para um array já ordenado, ou independente da ordem inicial ou para todas as entradas.

5) Com o Early Stopping podemos diminuir o número de comparações realizadas pelo método trazendo uma versão otimizada. Abaixo segue um exemplo de como funciona essa função:
   Suponha que temos o array [3, 2, 1, 4, 5], nele faremos 4 comparações para validar se o array está ordenado, na primeira passagem temos duas trocas ficando assim: [2,1,3,4,5], já na segunda passagem temos somente uma troca ficando da seguinte forma: [1,2,3,4,5]. No Bubble Sort para esse array fazemos 4 passagens por padrão que seria (n – 1), com o Early Stopping não precisamos realizar todas essas passagens, pois quando não existe mais troca colocamos um BREAK no código fazendo pausar e evitando fazer passagens desnecessárias, neste exemplo realizaremos 3 passagens com o early stopping.
6) O teste sucedeu com o seguinte resultado:



Com ele podemos ver como o Selecition Sort se comporta com maior número de dados se comparado com o Bubble Sort.

9) O Bubble Sort é estável, mantendo a ordem relativa de elementos iguais, enquanto o Selection Sort não é estável, pois ele pode trocar os elementos de forma não adjacente, alterando a ordem relativa. Segue o exemplo prático:

- Bubble Sort: [(2, A), (1, B), (3, C), (2, D)], ao usarmos o bubble sort a lista ficará assim - [(1, B), (2, A), (2, D), (3, C)]. Podemos observar que os dois valores com o número 2 mantiveram a posição original de ordem.
- Selection Sort: Já no Selection usando a mesma lista o resultado seria o seguinte: [(1, B), (2, D), (2, A), (3, C)], ele fez a troca na ordem dos vetores com valor 2, por isso o Selection não é estável.

10) O método Insertion Sort insere cada elemento em uma sub lista ordenada. Ele começa do segundo elemento da lista desordenada e vai comparando com o anterior e movendo-o para a posição correta, assim repetindo o processo até ordenar a lista. Um exemplo prático seria uma ordenação de notas do aluno no sistema aonde assim que for inserido uma nota a lista de notas pode ser reordenada.

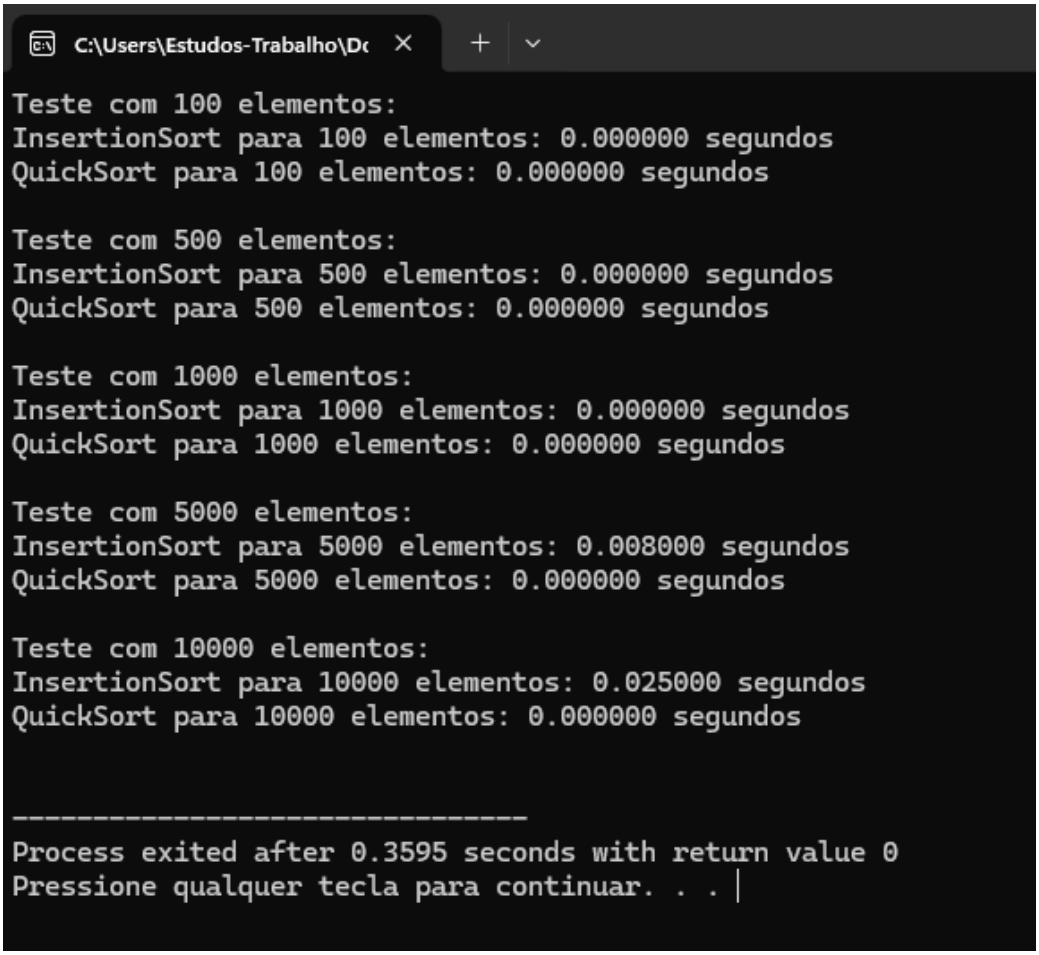
12) O Insertion Sort é mais eficiente quando o array está quase ordenado ou tem poucos elementos. Ele faz menos iterações deixando o algoritmo mais eficiente. No melhor caso do Insertion será o O(n) e os demais O(n^2).

13) No Insertion Sort temos três tipos de casos iguais aos outros métodos, abaixo segue a comparação entre esses casos:

- **Melhor Caso:** No melhor caso aonde o array já está ordenado ele faz as comparações, mas sem realizar nenhuma troca. Logo sua complexidade será O(n);
- **Pior Caso:** No pior caso o array está ordenado de forma inversa, assim o método terá que percorrer todo o array para fazer a comparação e ser movido as primeiras posições. Nesse caso sua complexidade será O(n^2);
- **Caso médio:** Já o caso médio podemos exemplificar com elementos ordenados de forma aleatória, ele se compara muito com o pior caso, mas tem uma ligeira melhora em questão

de trocas e comparações, mas a ordem ainda continua sendo quadrática. Sua complexidade é definida por O(n^2).

16) Ao realizar uma comparação do método Quick com o Insertion Sort temos o seguinte resultado:



Nele podemos observar que para volume de dados menores o Insertion pode ser quão rápido se comparado com o quick, agora já com volume de dados maiores o Quick se mostra superior, pois para entras médias e grandes temos a complexidade O(n log n), enquanto o Insertion cresce com O(n^2). Para pequenas entradas, InsertionSort pode ser eficiente, especialmente se a lista estiver quase ordenada.

17) O Quick tem o funcionamento aonde o array é subdivido em subarrays menores de forma recursiva graças ao elemento pivô. Na média complexidade temos O(n log n) e o pior caso O(n^2). Para que o método se enquadre na complexidade média a função tem uma recursividade aonde o vetor se divide e temos (log n) e a cada nível temos O(n) operações, fazendo a junção temos a complexidade O(n log n). Já no pior caso o pivô se divide de uma forma desbalanceada, como ao escolher sempre o primeiro ou último elemento em um vetor já ordenado. Isso cria n níveis recursivos em vez de log n.

18) O quick Sort não é considerado um método estável, pois ele não preserva a ordem relativa dos elementos do array após a ordenação. Isso ocorre porque durante a divisão os elementos podem ser movidos de forma não previsível, e um elemento igual pode acabar na posição de outro.