**Introdução**

Para este trabalho, foi pedido que implementasse 4 metodos de ordenação, sendo estes Merge Sort, Selection Sort, Insertion Sort e Bubble Sort. Cada um da forma crescente e decrescente para um arquivo externo de palavras.

**Solução Proposta**

Para economia de codigo, o primeiro menu verifica se o usuario quer ordenar crescente ou decrescentemente, com essa informação, pude economisar linhas de codigo, ja que a mesma função chamada pra ordenar crescentemente, tambem ordena decrescentemente, graças a uma estrutura condicional IF. O segundo menu é uma função fora da principal, e esta atraves de um SWITCH leva o programa para as funçoes de ordenação dependendo da escolha do usuario. A Função main roda em um while simples que é responsavel pela pergunta sobre a ordem que se quer ordenar, dentro dela é chamada o segundo menu, que recebe como parametro a primeira resposta, que vai ser importante para decidir em qual ordem será ordenado a lista, o segundo menu recebe a escolha do metodo e atravez de um switch leva para a função desejada, levando a resposta do primeiro menu isto é, a ordem que se deseja ordenar a lista. Cada função de ordenação tem dois modos de entrar, se a primeira resposta for 1 ou 2, e sao metodos classicos de ordenação sem alguma alteração

**Analise de Complexidade**

Merge Sort – O(N log N) para melhor e pior caso, tem vantagem de ser um algorimo regular, isto é, sabemos o que esperar dele, mesmo que ele nao chegue nunca à complexidade ideal (proximo de N), podemos saber à priori quanto ele demorará para executar.

Insertion Sort – Melhor caso O(N);

Pior caso O(N^2);

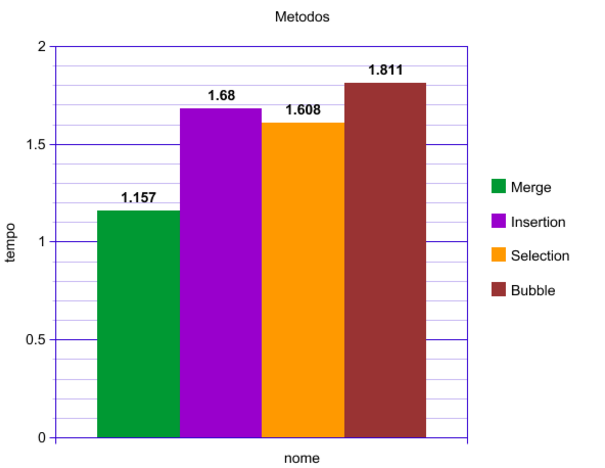
Bubble Sort - Melhor caso O(N);

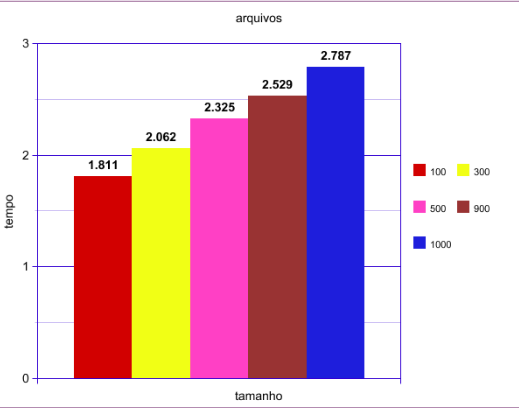
Pior caso O(N^2);

Selection Sort - O(N^2) para todos os casos

**Testes**

Baseado no tempo de execução do proprio compilador fiz a seguinte analise, ilustrada em graficos, ja que nao consegui de nenhuma forma colocar timer no meu codigo





**Maquina Utilizada**

HP 1000 com windows 10, compilador code blocks.

**Conclusão**

Com o trabalho pude estudar mais a fundo estes metodos de ordenação e determinar destes qual é o melhor, que obtive como resultado o merge, mesmo que para um grupo pequeno de dados, notou-se a difeença, o que se notaria mais claramente em um grupo maior de dados, isso se deve a sua complexidade de ordem O(N log N), enquanto os rivais tem o pior caso erm O(N log N).