Nome: Rafael da Costa Ramos RA: 131093512775

Ciência da Computação 4º Semestre

Análise de Complexidade e Computabilidade de Algoritmos

Relatório de Testes

•**Bubble Sort** – Teste de todos para todos, as primeiras funções são para randomizar e a função de ordenação que mais tarde será chamada na função principal “main”. Ele compara a posição atual com o próximo item e se o atual for maior que o próximo ele realiza a troca, faz esse loop sucessivamente até que a lista esteja ordenada. Este é o PIOR algoritmo em relação a sua eficiência e sua complexidade de tempo é no melhor dos casos O(n) e em médio e pior caso é $O(n^2)$. Na função “main” tem 3 blocos de “for” que estão comentadas, se tirar o comando de comentário e ativar, será realizada a ordenação de acordo com o “for” escolhido, sendo correto apenas 1 estiver ativo enquanto os outros 2 estão comentados para não afetar o algoritmo.

•**Insert Sort** – Esse algoritmo funciona como organizar uma mão de baralho na mão, você seleciona o valor na primeira posição, e compara com o próximo, se o próximo for menor é realizada a troca e olha se o valor anterior a ele também seja menor que o número até que o valor que está sendo olhado seja maior que o valor a esquerda, nesse caso ele se encontrará na posição correta, então continua na posição que parou + 1 e continua comparando o elemento a frente como se estivessem chegando cartas na mão e o jogador estivesse organizando as cartas em sequência para saber o que falta para ganhar o jogo, a sua complexidade é no melhor dos casos O(n) e em médio e pior caso é $O(n^2)$. Na função “main” tem 3 blocos de “for” que estão comentadas, se tirar o comando de comentário e ativar, será realizada a ordenação de acordo com o “for” escolhido, sendo correto apenas 1 estiver ativo enquanto os outros 2 estão comentados para não afetar o algoritmo.

•**Selection Sort** – O objetivo desse algoritmo é Ordena o array “vet” de tamanho “n” usando o algoritmo Selection Sort. Basicamente o selection sort é feita uma sublista não ordenada “j” e se o elemento encontrado não for igual a “i” ele é trocado, sua complexidade de tempo em melhor, médio e pior caso é $O(n^2)$. Na função “main” tem 3 blocos de “for” que estão comentadas, se tirar o comando de comentário e ativar, será realizada a ordenação de acordo com o “for” escolhido, sendo correto apenas 1 estiver ativo enquanto os outros 2 estão comentados para não afetar o algoritmo

•**Merge Sort** – Esse em seu desempenho e sua eficiente se comporta como o MELHOR algoritmo de ordenação desse relatório, basicamente se você tem mais de um elemento divida-o ao meio e chame recursivamente o merge sort para cada metade, compara as metades e as coloca em ordem em uma sublista temporária e depois copia o array temporário para o original e sua complexidade de tempo em melhor, médio e pior caso é O(n log n). Na função “main” tem 3 blocos de “for” que estão comentadas, se tirar o comando de comentário e ativar, será realizada a ordenação de acordo com o “for” escolhido, sendo correto apenas 1 estiver ativo enquanto os outros 2 estão comentados para não afetar o algoritmo.