1) **Bubble Sort -** O Bubble Sort é o algoritmo de ordenação mais simples, ele funciona trocando repetidamente os elementos adjacentes se eles estiverem na ordem errada. Ela sempre irá rodar no pior caso O(n^2) mesmo se o vetor já estiver ordenado, porém pode ser otimizado para parar o algoritmo caso não há nenhuma troca.
2) **Insertion Sort -** O algoritmo primeiramente marca a seção onde o vetor ordenado está localizado, primeiramente, essa seção é após o primeiro elemento, ou seja o elemento na posição 0 está ordenado, e adiante ainda não ordenou. Logo após, ele irá avançar a seção da parte ordenada do vetor, e irá fazer comparação, e ir trocando o elemento até sua posição. Ele repete o processo até a seção ordenada ser o vetor completo. Seu pior e médio caso na notação Big O é O(n^2).
3) **Quick Sort -** O Quick Sort é um algoritmo que utiliza o conceito de dividir e conquistar. Ele escolhe um elemento como pivô, e divide o vetor em torno do pivô escolhido, existem diversas versões do Quick Sort que escolhem pivô de maneira diferente, a implementada no código trata o último elemento como pivô. Ele utiliza duas variáveis, uma para guardar a posição do menor elemento (i), e a outra para controlar o loop (j). Caso a comparação seja verdade, ele incrementa o “i”, e troca o elemento na posição “i” com a posição “j”. Se a comparação for falsa, ele não faz nada, e continua com o looping. Como a função é recursiva, ele repete até o vetor ficar ordenado. Seu pior caso na notação Big O é O(n^2), e em média O(n log(n)).
4) **Shell Sort -** O Shell Sort é uma variação do Insertion Sort. Em Insertion Sort movemos elementos apenas uma posição à frente, a ideia do Shell Sort é permitir a troca de valores distantes. Ele começa na posição n/2, onde n é o tamanho do vetor, e irá colocar os elementos a direita da lacuna na sua posição correta. Sua complexidade na notação Big O é O(n(log(n))^2) para seu pior e médio caso.
5) **Merge Sort -** Assim como o Quick Sort, o Merge Sort utiliza o conceito de dividir e conquistar. Ele irá se chamando e repartindo pela metade recursivamente toda o vetor da esquerda até o momento em que não houver mais como repartir, ele irá comparar os dois elementos, trocá-los caso necessário, e depois juntar ambos, depois irá repetir o processo na parte direita. No final ele irá comparar o resultado a parte esquerda, com a parte da direita, e ordenar caso tenha necessidade. Na notação Big O, tanto o tempo médio quanto o pior caso é O(n log(n)).
6) **Bucket Sort -** O Bucket Sort é um algoritmo de ordenação que funciona divindo o vetor em um número finito de recipientes, logo depois cada recipiente é ordenado individualmente, depois irá organizando os recipientes e gerando o vetor ordenado completo. O Bucket Sort, dentre os estudados é o que apresenta melhor desempenho, seu pior caso é O(n^2), porém seu tempo médio de solução é O(n+k).