Insertion sort: esse algoritmo ordena os elementos comparando do final para o começo posicionando os elementos em seus exatos locais para que o conjunto permaneça ordenado, essa comparação é dada criando um vetor e aumentando em 1 o seu tamanho até que não se tenha mais valores, a cada novo valor dentro do vetor criamos uma comparação verificando se esse valor é menor do que o anterior a ele, caso for realiza essa ação repetidamente até que esse valor seja maior que algum outro valor então é adicionado na posição à frente desse valor ou até o valor de j ser igual a -1

A lista[j] no nosso código representa o número anterior ao que está sendo adicionado e a lista[i] pega o novo valor

Complexidade: quadrática

Exemplo: Baralho

Selection sort: esse algoritmo ordena os elementos encontrando o menor valor dentro de um vetor e adiciona ele na sua respectiva posição após isso acontecer ele pula uma posição para não compara mais valores com esse menor valor que foi encontrado realizando isso repetidamente até atingir o tamanho máximo do vetor, ele troca os valores de lugar EX: se temos 33 na posição 0 do vetor e 5 é o menor valor do vetor que está na posição 4, o 5 ocupa a posição 0 e o 33 vai para a posição 5

O i nesse código é responsável por pular a posição e o j por percorrer o vetor

Complexidade: quadrática

Exemplo: Fila de escola

Shell sort: esse algoritmo ordena os elementos utilizando um incremento onde no nosso código o incremento é o tamanho da lista dividido por 2 e esse incremento é responsável por definir o espaçamento dos itens que serão comparados, ou seja, se o incremento é 4 ele irá comparar os itens que estão nos índices 0 e 3 e ordena de acordo com que é o menor dessa comparação após realizar todas as comparações divide o número do incremento por 2 até que o incremento seja menor do que 1

Exemplo: Utilização de incremento

Heap sort: esse algoritmo ordena os elementos utilizando uma árvore binária que após realizar a criação dessa árvore realiza o heap dela, ou seja, cada pai é maior que seus nó filhos e logo após realizar isso ele retira o maior valor da árvore adicionando esse valor no final da lista e diminuído o tamanho da lista para não comparar mais com esse valor e realiza o heap novamente, só andamos até a metade da lista pois assim já conseguimos criar a árvore

Exemplo: Árvore binária

Quick sort: esse algoritmo ordena os elementos dividindo-os em problemas menores selecionamos um pivô e garantimos com que o número a esquerda seja menor que o valor do pivô e o valor a direita seja maior do que o valor do pivô sorteando um novo pivô para as partes que sobraram

Exemplo: Utilização do pivô

Complexidade: logarítmica