Trabalho Final de Estrutura de Dados I

1)

Selection Sort:

Definição

O selection sort é baseado em se passar o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem desejada), depois o de segundo menor valor para a segunda posição, e assim é feito sucessivamente com os n − 1 elementos restantes, até os últimos dois que restarem.

Algoritmo em C

void selection\_sort(int num[], int tam) {

int i, j, min, aux;

for (i = 0; i < (tam-1); i++)

{

min = i;

for (j = (i+1); j < tam; j++) {

if(num[j] < num[min])

min = j;

}

if (num[i] != num[min]) {

aux = num[i];

num[i] = num[min];

num[min] = aux;

}

}

}

Vantagens

* Um algoritmo simples de ser implementado em comparação aos demais.
* Não necessita de um vetor auxiliar (in-place).
* Por não usar um vetor auxiliar para realizar a ordenação, ele ocupa menos memória.
* Uns dos mais velozes na ordenação de vetores de tamanhos pequenos.

Desvantagens

* Um dos mais lentos para vetores de tamanhos grandes.
* Não é estável.
* Faz sempre n² comparações, independente do vetor está ordenado ou não.

https://pt.wikipedia.org/wiki/Selection\_sort

Insertion Sort:

Definição

É o algoritmo de ordenação que, dado uma estrutura (array, lista) constrói uma matriz final com um elemento de cada vez, uma inserção por vez. Assim como algoritmos de ordenação quadrática, é bastante eficiente para problemas com pequenas entradas, sendo o mais eficiente entre os algoritmos desta ordem de classificação.

Algoritmo em C

void insertionSort(int original[], int length) {

int i, j, atual;

for (i = 1; i < length; i++) {

atual = original[i];

for (j = i - 1; (j >= 0) && (atual < original[j]); j--) {

original[j + 1] = original[j];

}

original[j+1] = atual;

}

}

Vantagens:

* É o método a ser utilizado quando o arquivo está "quase" ordenado
* É um bom método quando se desejar adicionar poucos elementos em um arquivo já ordenado, pois seu custo é linear.
* É estável.

Desvantagens:

* Alto custo de movimentação de elementos no vetor.

https://pt.wikipedia.org/wiki/Insertion\_sort

Bubble Sort:

Definição:

É um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, e a cada passagem fazer flutuar para o topo o maior elemento da sequência. Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo.

Algoritmo

**procedure** bubbleSort( A **:** lista de itens ordenaveis ) **defined as:**

**do**

trocado := false

**for each** i **in** 0 **to** length( A ) - 2 **do:**

// verificar se os elementos estão na ordem certa

**if** A[ i ] > A[ i + 1 ] **then**

// trocar elementos de lugar

trocar( A[ i ], A[ i + 1 ] )

trocado := true

**end if**

**end for**

// enquanto houver elementos sendo reordenados

**while** trocado

**end procedure**

Vantagens:

No melhor caso, o algoritmo executa {\displaystyle n} operações relevantes, onde {\displaystyle n} representa o número de elementos do vetor.

Desvantegens:

No pior caso, são feitas {\displaystyle n^{2}} operações. A complexidade desse algoritmo é de ordem quadrática. Por isso, ele não é recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados.

https://pt.wikipedia.org/wiki/Bubble\_sort

Radix Sort:

Definição:

é um algoritmo de ordenação rápido e estável que pode ser usado para ordenar itens que estão identificados por chaves únicas. Cada chave é uma cadeia de caracteres ou número, e o *radix sort* ordena estas chaves em qualquer ordem relacionada com a lexicografia.

Algoritmo em C

void radixsort(int vetor[], int tamanho) {

int i;

int \*b;

int maior = vetor[0];

int exp = 1;

b = (int \*)calloc(tamanho, sizeof(int));

for (i = 0; i < tamanho; i++) {

if (vetor[i] > maior)

maior = vetor[i];

}

while (maior/exp > 0) {

int bucket[10] = { 0 };

for (i = 0; i < tamanho; i++)

bucket[(vetor[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

bucket[i] += bucket[i - 1];

for (i = tamanho - 1; i >= 0; i--)

b[--bucket[(vetor[i] / exp) % 10]] = vetor[i];

for (i = 0; i < tamanho; i++)

vetor[i] = b[i];

exp \*= 10;

}

free(b);

}

Vantagens:

Desvantegens:

Quick Sort:

Definição

Algoritmo

Vantagens:

Desvantegens:

Merge Sort:

Definição

Algoritmo

Vantagens:

Desvantegens:

Busca sequencial:

Definição

Algoritmo

Vantagens:

Desvantegens:

Busca Binária:

Definição

Algoritmo

Vantagens:

Desvantegens: