Métodos de Ordenação

Métodos de ordenação são utilizados para organizar itens em uma determinada seqüência. Estes algoritmos podem ser utilizados para ordenar qualquer tipo de dados, desde que sejam passíveis de ordenação crescente ou decrescente.

Exemplos de aplicação: ordem numérica e ordem alfabética.

Bubble sort (bolha)

Utiliza a idéia de percorrer um vetor diversas vezes. A cada vez que ocorre um percurso, o maior elemento da seqüência vai para o topo. Devido a este deslocamento do maior elemento ser semelhante a uma bolha em um tanque com água (a bolha sempre procura mais próximo a superfície), este algoritmo recebe este nome.

Características:

- é um dos mais simples algoritmos de ordenação;
- de fácil implementação.

Algoritmo em C++:

```
void bubblesort (int a[10]){
     for(int j= 10-1; j>0; j--){
          for(int i=0;i<j;i++){
               if(a[i+1] < a[i]){
                    int aux = a[i];
                    a[i] = a[i+1];
                    a[i+1] = aux;

               }
          }
     }
}
```

Selection sort

Semelhante ao Bubble sort, utiliza a idéia de percorrer um vetor diversas vezes. A principal diferença está em que o algoritmo procura os menores valores e vai colocando-os nas posições iniciais da seqüência.

Características:

- é um dos mais simples algoritmos de ordenação;
- de fácil implementação.

Algoritmo em C++:

```
// tam é o tamanho do vetor
void selectionsort(int vetor[],int tam) {
   int i, j;
   int min, aux;
   for(i=0; i<tam-1; i++) {
      min = i;
      aux = vetor[i];
      for(j=i+1; j<tam; j++) {
        if (vetor[j] < aux)
        {
           min=j;
           aux=vetor[j];
        }
      }
      aux = vetor[i];
      vetor[i] = vetor[min];
      vetor[min] = aux;
   }
}
```

Insertion sort

Funcionamento: percorre uma seqüência de elementos da esquerda para a direita, e a medida que avança, vai deixando os vetores da esquerda ordenados.

Características:

- é um dos mais simples algoritmos de ordenação;
- de fácil implementação;
- eficiente quando aplicado a um pequeno número de elementos.

Algoritmo em C++:

```
// tam é o tamanho do vetor
void insertionsort(int vec[], int tam) {
  int i, j;
  int key;

  for (j = 1; j < tam; ++j) {
    key = vec[j];
    i = j - 1;
    while (vec[i] > key && i >= 0) {
      vec[i+1] = vec[i];
      --i;
    }

    vec[i+1] = key;
  }
}
```

Quick sort

O algoritmo de Quick sort adota a técnica de divisão por conquista, e utiliza os seguintes passos:

1. seleciona um elemento da lista, denominado pivô;
2. rearranja a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele. Ao fim do processo o pivô estará em sua posição final e haverá duas sublistas não ordenadas. Essa operação é denominada partição;
3. recursivamente ordene a sublista dos elementos menores e a sublista dos elementos maiores.

Características:

- método de ordenação muito rápido e eficiente.

No programa abaixo há dois métodos que representam a implementação do método Quick sort para na linguagem C++.

Exemplo de programa utilizando o método de Quick sort (em C++):

```
#include <cstdlib>
#include <iostream>
using namespace std;

int partition(int vec[], int left, int right) {
  int i, j;

  i = left;
  for (j = left + 1; j <= right; ++j) {
    if (vec[j] < vec[left]) {
      ++i;
      int aux = vec[i];
      vec[i] = vec[j];
      vec[j] = aux;
    }
  }
  int aux = vec[left];
  vec[left] = vec[i];
  vec[i] = aux;


  return i;
}

void quicksort(int vec[], int left, int right) {
  int r;

  if (right > left) {
    r = partition(vec, left, right);
    quicksort(vec, left, r - 1);
    quicksort(vec, r + 1, right);
  }
}


int main(int argc, char *argv[])
{
    int vetor[] = {9, 7, 0, 1, 2};
    quicksort(vetor, 0, 4);
    for (int i=0; i<5; i++){
        cout<<vetor[i]<<",";
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}
```