MÉTODOS DE ORDENAÇÃO DE VETORES Renan Roos - 21/04/2016

Esse manual apresentará rapidamente como funciona cada método de ordenação de dados na teoria com pseudocode e/ou na linguagem C#. Também teremos os prints de como é realizado no nosso software de execução. Todo o software foi feito na linguagem C# e dentro de cada método há o código comentado.

São aplicados nesse trabalho os métodos de seleção, inserção, heap, quick e oddeven (par-ímpar). Todos os exemplos mostrados e printados aqui, foram feitos com vetores de tamanho de 100 caracteres.

MÉTODO DE SELEÇÃO (SELECTION SORT)

Um dos algoritmos mais simples de ordenação. Funciona da seguinte maneira. O método selecione o menor número do vetor, trocando de posição com o primeiro da lista

Assim, o segundo da lista será comparado com o restante, caso encontre um menor que ele, eles trocam de lugar. E assim é realizado todo o processo até todo o vetor estar ordenado.

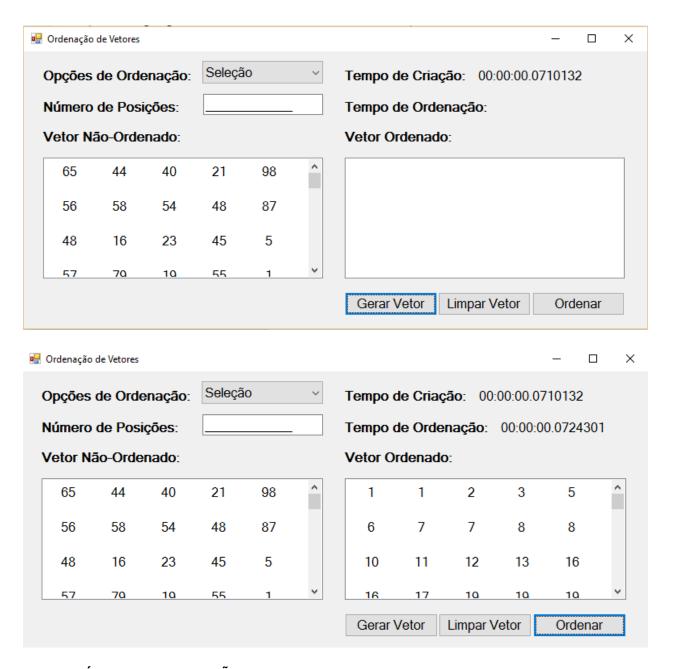
Vantagens: Custo linear no tamanho da entrada para o número de movimentos de registros – a ser utilizado quando há registros muito grandes.

Desvantagens: Não adaptável. Não importa se o arquivo está parcialmente ordenado. Algoritmo não é estável.

Código em C usado em nosso material de referência, vide seção Referência Bibliográfica.

No programa de ordenação de dados:

Método: Seleção



MÉTODO DE INSERÇÃO (INSERTION SORT)

O método de Inserção divide o vetor em duas partes, uma lista classificada e uma não classificada. Em cada movimento, o algoritmo move a variável do vetor não classificado para o classificado, até estar toda classificada.

E assim faz todas as comparações, como acontece no método de Seleção.

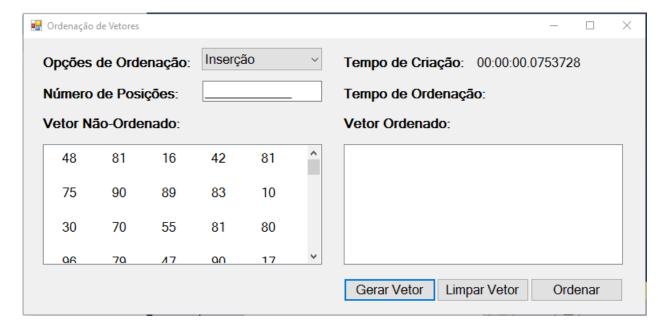
Vantagens: Laço interno é eficiente, inserção é adequado para ordenar vetores pequenos, pois o custo é linear. É estável.

Desvantagens: Número de comparações tem crescimento quadrático. Alto custo de movimentação de elementos no vetor.

Código em C usado em nosso material de referência, vide seção Referência Bibliográfica.

No programa de ordenação de dados:

Método: Inserção





• MÉTODO HEAP SORT

O Heap Sort trata-se de grafos em uma árvore binária, como estudamos em Estrutura de Dados. Teremos no topo da árvore o "max heap", então temos os pais, e cada pai deve ser maior que seus filhos que estão no nível abaixo. Exatamente conforme uma árvore binária funciona. E são preenchidos da esquerda para a direita.

Primeiro trocamos o menor número pelo "max heap", pois como ele é nosso maior número, ele vai para o final do vetor, e não precisamos mais nos preocupar com ele, pois ele já está no lugar correto.

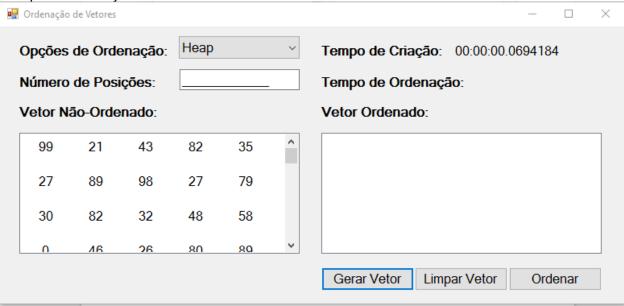
Como o número que foi trocado não é o maior do vetor, começam as comparações entre o número, conforme é feito nas árvores binárias, entre pais e filhos para fazermos do "max heap" o maior número da árvore. Feito isso, ele troca de lugar com o penúltimo da lista, e fica lá, pois está no lugar correto. E assim, recomeça todo o algoritmo novamente, até termos uma lista completamente ordenada.

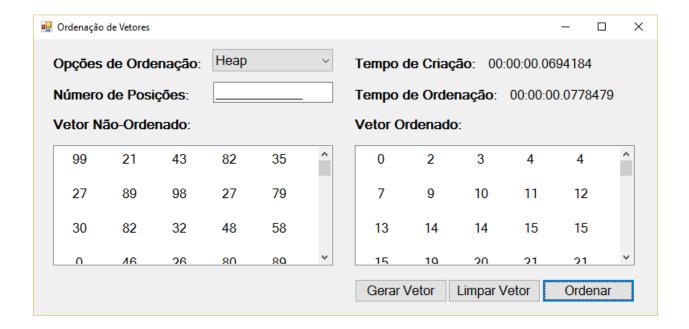
Código em C usado em nosso material de referencie, vide seção Referência Bibliográfica.

```
void Refaz(Indice Esq, Indice Dir, Item *A)
{    Indice i = Esq;
    int j;
    Item x;
    j = i * 2;
    x = A[i];
    while (j <= Dir)
    {       if (j < Dir)
            {             if (A[j].Chave < A[j+1].Chave) j++;
            }
            if (x.Chave >= A[j].Chave) goto L999;
            A[i] = A[j];
            i = j; j = i *2;
        }
        L999: A[i] = x;
        Algoritmos e Estrutura de Dados II
```

No programa de ordenação de dados:

Método: Heap Sort





MÉTODO QUICK SORT

O método Quick Sort provavelmente é o mais utilizado entre os desenvolvedores. O seu principal foco é dividir um grande vetor, em dois vetores menores para facilitar e agilizar o ordenamento.

Temos alguns elementos para a ordenação, o primeiro é o pivot, que é o conhecido como a variável auxiliar. Para dividir esse vetor em dois menores, usaremos o Wall (muro) para separar entre lista ordenada e lista não ordenada. O pivot é o último do vetor, e ele compara-se a todos os outros números dentro do vetor. Quando ele acha um número menor que ele, automaticamente ele joga esse número para o lado esquerdo do Wall, que é nossa lista ordenada. Reordenaremos até todos os números menores que o pivot esteja do lado esquerdo da Wall. Caso o menor número esteja no meio da lista, ele trocará de lugar com o primeiro número da lista e então a Wall vai para depois desse número.

Se o primeiro número do lado direito da Wall for maior que nosso pivot, ele troca de lugar com o pivot, agora, tornando-se o pivot. E assim por diante, refazendo todo o processo até todo o vetor estar ordenado.

Vantagens:

Necessita de apenas uma pequena pilha como memória auxiliar que ele usa. Esse memória que é utilizada a mais que os outros métodos, é maior, mas não tão significativa para o computador. Tendo essa memória a mais, faz com que o método seja um dos mais rápidos, por isso o nome quick (rápido).

Desvantagens:

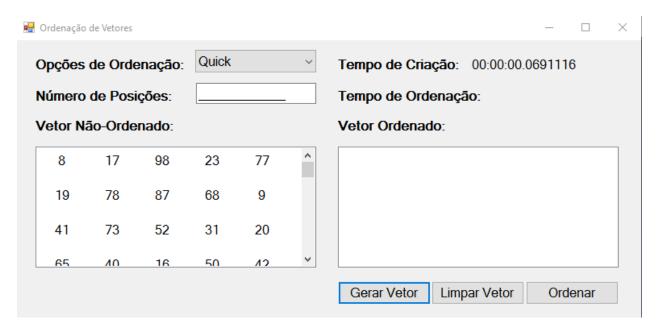
Sua implementação é muito delicada e difícil, pois um pequeno engano pode fazer com que gastemos mais memória do que realmente precisaríamos. O método não é estável.

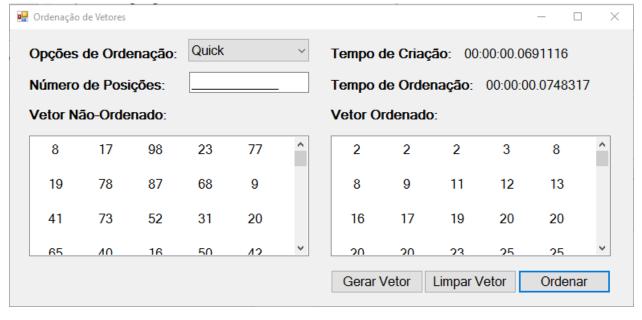
Código em C usado em nosso material de referência, vide seção Referência Bibliográfica.

■ Função Partição:

No programa de ordenação de dados:

Método: Quick Sort



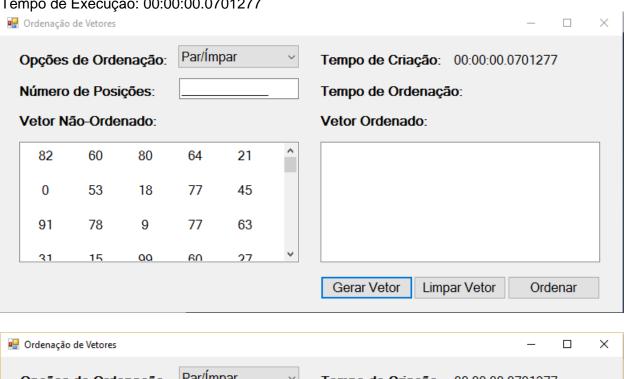


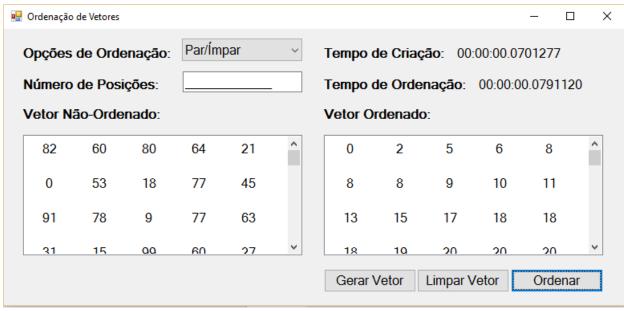
MÉTODO PAR/ÍMPAR (ODD-EVEN)

É uma variação do Bubble Sort, onde Even significada par e Odd é ímpar. Ele funciona através da comparação de todos os números ímpares e pares do vetor. É feita a comparação entre os números, formando pares ordenados. Então, o primeiro é par e o segundo ímpar, caso fuja desse padrão, eles trocam de lugar.

No programa de ordenação de dados:

Método: Par-Ímpar





• COMPARAÇÃO DE DESEMPENHO ENTRE OS MÉTODOS

Inserção:

Melhor usar para menos de 20 elementos. Método é estável.

Melhor desempenho que Bubble Sort.

Seleção:

Melhor usar para até mil números.

Melhor desempenho que Bubble Sort e Inserção.

Quicksort:

É recursivo, o que demanda uma pequena quantidade de memória adicional.

Seu auxiliar chama-se pivot. Chama um método de ordenação simples nos arquivos pequenos.

Melhor desempenho que Bubble Sort, Inserção e Seleção.

Heapsort:

Não necessita de nenhuma memória adicional.

Feito com grafos e árvores binárias, dentro da Estrutura de Dados. Dá rapidez aos movimentos.

Ímpar-Par:

Variação do Bubble Sort, mas com características específicas.

É mais complicado do que implementar o Bubble.

Utilizados em casos onde precisamos somente ordenar por par ou impar.

REFERÊNCIAS BIBLIOGRÁFICAS

Ordenação: Introdução e métodos elementares http://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/sorting-intro.pdf
Acessado em: 21/04/2016

Livro "Projeto de Algoritmos" – Nívio Ziviani Capítulo 4 - http://www2.dcc.ufmg.br/livros/algoritmos/ Acessado em: 21/04/2016

Algoritmos de Ordenação - http://www.rafaeldiasribeiro.com.br/downloads/ED_4.pdf
Acessado em: 21/04/2016