Nome: Mateus Tizotti
Disciplina: Estrutura de Dados em C - 53

Método Bubblesort

O algoritmo de Bubblesort, ou de ordenação por bolhas, é um dos métodos mais simples de se organizar, de maneira tanto crescente quanto decrescente, os itens de um determinado array.

Devido à essa complexidade mais baixa, o algoritmo opera em média com magnitude $O(n^2)$, ou seja, o número de operações necessárias à serem realizadas em um array de n elementos, tende a crescer quadraticamente, o que o torna muito ineficiente para trabalhar com programas que trabalham com uma grande quantidade de dados e necessitam de grande performance.

O funcionamento do algoritmo é bem simples. Basicamente, a cada vez que iteramos pelo array queremos trazer o maior valor para o final da lista, dessa forma, já estará em seu devido local. Como último elemento da lista correto, vamos fazer o mesmo procedimento para o resto do array, por exemplo, em um array de ordem n, realizamos o procedimento de bubblesort para que o maior item da listagem termine na posição n, e em seguida, realizamos o procedimento para um array de ordem n-1, fazendo com que o segundo maior item da lista acabe na posição n-1.

O procedimento citado, é simplesmente iterar pelo array, da posição i=0 até i=n comparando os itens em pares, e caso o item na posição i seja maior que o da posição i+1, trocá-los. Por exemplo:

0	**1**	**2**	**3**	**4**	**5**
5	6	2	1	4	3

Neste array de 6 elementos, realizando o algoritmo de bubblesort iremos comparar primeiramente o elemento de índices 0 e 1, como 5 é menor do que 6, não realizamos nenhuma operação. A seguir comparamos os índices 1 e 2, e como 6 é maior do que 2, trocamos os dois de lugar na listagem.

A troca pode ser realizada diretamente no código ou podemos desenvolver uma função separada para realizar esse procedimento, mas o método continua o mesmo. Armazenamos o valor de array[i] em uma variável temporária, array[i] recebe o valor de array[i+1], e array[i+1] recebe o que armazenamos anteriormente na variável temporária.

Essa troca irá ocorrer até o final do array, uma vez que 6 é o maior número da listagem. Teremos então:

0	**1**	**2**	**3**	**4**	**5**
5	2	1	4	3	6

Agora basta realizar o mesmo procedimento iterando pelo array de i=0 até i = n-1, para ajustarmos a posição do 5, e assim por diante. A visualização dos índices de maior valor subindo para o topo do array faz com que esse método receba seu nome, pela semelhança com bolhas subindo em um tanque de água.

Método Quicksort

O método de ordenação quicksort, por outro lado, possui um nível de complexidade um pouco mais elevado, porém é notavelmente mais eficiente que o método bubblesort. Este foi inventado por Charles Antony Richard Hoare em 1960, e possui complexidade de magnitude O(nlog(n)).

O algoritmo consiste em "Dividir e Conquistar" e aplicação do mesmo método em suas subdivisões através da recursão de uma função de ordenação.

Para aplicarmos este algoritmos devemos seguir alguns passos. Inicialmente, com o nosso array de inteiros fora de ordem, devemos arbitrariamente escolher um "pivô", que irá servir como o ponto de quebra do nosso array, onde iremos colocar todos os números maiores que ele à sua direita e todos os menores à sua esquerda. Desta forma, após essa divisão, nosso pivô estará na sua posição correta.

Recursivamente devemos aplicar esse método para a parte esquerda do pivô (números menores) e a parte direita (números maiores), escolhendo novos pivôs para cada caso e realizando a ordenação.

Vamos exemplificar o método com o mesmo array utilizado no método de blubblesort:

0	**1**	**2**	**3**	**4**	**5**
5	6	2	1	4	3

Por padrão, escolhemos o último elemento da lista como nosso pivô, então iremos iterar pelo nosso array de i=0 até i = n-1, comparando se o valor de cada item é maior ou menor que o nosso pivô. Essa ordenação do pivô é comumente chamada de "Partição", e referenciada dentro da função quicksort.

No começo da iteração, o nosso atual índice de retorno será o zero, uma vez que não sabemos se o nosso pivô será o primeiro item da lista.

Ao comparar o primeiro e o segundo item com o nosso pivô percebemos que eles são maiores, portanto não precisamos fazer nada e nosso índice de retorno permanece no zero. Ao comparar o terceiro item percebemos que 2 é menor que 3, portanto trocamos 2 de lugar com o nosso índice de retorno (posição zero), e subimos nosso índice em uma casa (posição 1).

Após comparar todos os itens, nosso índice de retorno estará na posição correta, e todos os números menores que nosso pivô a sua esquerda, analogamente teremos todos os maiores à sua direita. Apenas realizamos a troca do pivô para seu índice que acabamos de encontrar e retornamos o índice.

Recursivamente, apenas precisamos chamar a função quicksort novamente começando em 0 até nosso índice retornado, e mais uma vez do índice retornado até o final do array. Desta forma, iremos subdividir o array quantas vezes for necessário, até que todos os pivôs estejam na posição correta.

Referências:

https://pt.wikipedia.org/wiki/Bubble_sort (acessado em 24/03/2020)
https://pt.wikipedia.org/wiki/Quicksort (acessado em 24/03/2020)
https://www.youtube.com/watch?v=eqo2LxRADhU (acessado em 22/03/2020)
https://www.youtube.com/watch?v=67k3I2GxTH8&t=497s (acessado em 23/03/2020)