Quick Sort



Primeiro vamos falar de como o Quick Sort funciona, ele utiliza o paradigma de programação Dividir para Conquistar. Esse paradigma é uma abordagem recursiva em que a entrada do algoritmo é ramificada múltiplas vezes a fim de quebrar o problema maior em problema menores da mesma natureza.

Exemplo:

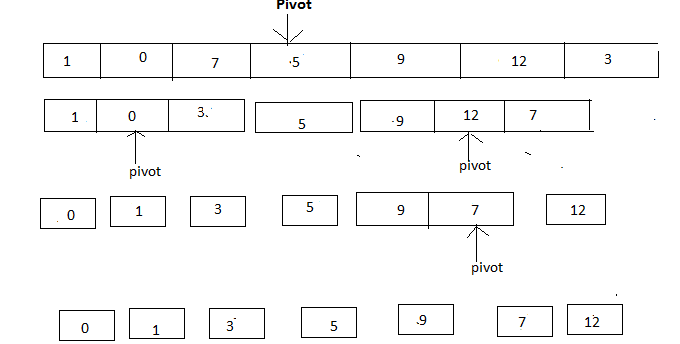
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 5 | 4 | 9 | 0 | 6 | 7 | 3 |

Primeiro passo temos que escolher um pivô que no nosso caso será o 3 e com isso temos que ir fazendo as trocas dos números até que todos os elementos a esquerda desse pivô sejam menores que ele e todos os elementos a direita desse pivô sejam maiores que ele. Neste caso a nossa tabela ficaria da seguinte forma

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 3 | 9 | 5 | 6 | 7 | 4 |

No caso nossa tabela ficaria da seguinte forma e agora a gente tem todos os elementos da esquerda do nosso pivô menor que ele e todos os elementos da direita de nosso pivô maiores que ele, com isso basta chamarmos o quick sort novamente como se fossem duas sub listas a primeira sub lista está na direita e a segunda sub lista está na esquerda e novamente precisamos definir um pivô para cada sub lista.

Abaixo vou colocar uma imagem para melhor entender essa configuração.



Aqui mostra as sub listas até chegarmos na lista totalmente ordenada.

Agora vamos para a explicação do Algoritmo quick sort realizado na linguagem (C).

Primeiramente eu determino menor >= maior dentro do if após isso eu seleciono meu pivô na posição x e determino as posições esquerda e direita

E coloco como direita = menor +1

E esquerda como maior e abro um laço de repetição while (direita <=esquerda), ou seja, eu vou repetir esse while até a direita for <= esquerda!

Após isso nos próximos while eu começo a colocar todos os valores que eu encontrar menor que o meu x na direita e todos meus valores maiores que o meu x na esquerda. Utilizando direita++ e esquerda--.

E faço um if dizendo se direita for menor que esquerda eu começo a realizar as trocas dos valores com a variável tempo [índice direita] / [índice esquerda] e insiro os valores nos locais corretos com direita++ e esquerda—

E após o if coloco vet [menor] = vet[esquerda] e vet [esquerda] = x assim fazendo a troca do menor que está na esquerda com o meu pivô.

Por fim coloco a função quick\_sort (vet,menor, esquerda -1) e quick\_sort (vet, esquerda +1), maior).

Complexidade

Melhor caso: O melhor caso do Quicksort ocorre quando as partições são balanceadas, isto é, cada partição com n/2n/2 elementos.

Pior Caso: O pior caso é simplesmente o contrário do melhor caso quando ocorre partições que não são balanceadas ou seja n/2 e n/1 n/2 elementos.