**QUICKSORT**

Segundo CORMEN (), o método de ordenação quicksort é com frequência a melhor opção prática para ordenação, apesar de no seu pior caso ele possuir um tempo de execução igual a θ(n2).

Ele se torna o mais utilizado devido a eficiência no tempo de execução médio que é de θ(n lg n). O que o torna assim é que esse algoritmo se baseia no paradigma de dividir e conquista, isso ocorre da seguinte formada dado um vetor A[p..r] primeiramente o quicksort irá quebrar esse vetor em duas partes, ou seja, **dividir** com isso temos um subarranjos de A[p..q-1] e A[q+1..r] o ponto onde é escolhido para se realizar a “quebra do vetor ” chamamos de **pivô.**

Esses dois subarranjos são percorridos de p até q-1 (onde q é a posição do nosso **pivô**) com uma variável i que incrementa caso a cada vez que A[i]> A[q], e de q+1 até r com uma variável j em que é decrementada enquanto A[q] < A[j]. E com isso são ordenados por chamadas recursivas a quicksort. Segundo ZIVIANI () ao final o vetor A[p..r] estará particionado de tal forma que os itens em A[p], A[p+1],..A[j] são menores ou iguais a q, e os itens em A[i], A[i+1],..., A[r] são maiores ou iguais a q.

A seguir iremos apresentar sobre o seu funcionamento nas seguintes ordem: Pseudocódigo, Análise do algoritmo, tabelas e gráficos da análise.

Começamos pelo procedimento que implementa o particionamento do Quicksort:

**QUICKSORT** (A,p,r)

1. **if** p < r
2. **then** q := **PARTITION** (A,p,r)
3. **QUICKSORT**(A,p,q-1)
4. **QUICKSORT**(A, q+1,p)

Para ordena um arranjo A inteiro, a chamada inicial é QUICKSORT(A,1,comprimento[a]).

**Particionamento do arranjo:**

Segundo CORMEN o procedimento PARTITION, é que reorganoza o subarranjo A[p..r] localmente:

PARTITION (A,p,r)

1. x ← A[r]
2. i ← p – 1
3. for j:= p to r-1
4. do if A[j] <= x
5. then i ← i + 1
6. trocar A[i] ← A[j]
7. trocar A[i + 1] ↔ A[r]
8. return i +1