A ideia é escolher um pivo, achar o local para este pivo e depois dividir o vetor em outros dois, aqueles que são menores que o pivo e aqueles que são maiores que o pivo e chamar o mesmo algortimo para ambos esses vetores até que tudo esteja ordenado.

A ideia para achar o local do pivo é colocar o pivo no ultimo lugar do array e depois:

i = inicio - 1

j = inicio

Com isso vc começa a analisar se v[j] é menor que o pivo, pois se for vc vai aumentar o i e trocar o i e o j de lugar:

if (v[j] < pivo){

i++;

swap(i, j);

}

Vc faz isso até chegar no fim do vetor, ou seja, até j estar no pivo, quando isso acontecer vc vai trocar j (o fim) com i + 1, isso pois, todos os elementos de i para tras são elementos menores que o pivo e todos elementos de i + 1 para frente são elementos maiores que o pivo, logo a implementação ficaria:

void swap(int \*v, int i, int j){

int tmp = v[i];

v[i] = v[j];

v[j] = tmp;

}

int Partition(int \*v, int inicio, int fim){

int meio = (inicio + fim) / 2;

int pivo = v[meio];

swap(v, meio, fim);

int i = inicio - 1;

int j;

for (j = inicio; j < fim; j++){

if (v[j] < pivo){

i++;

swap(v, i, j);

}

j++;

}

swap(v, i + 1, fim);

return i + 1;

}

void QuickSort(int \*v, int inicio, int fim){

if (inicio >= fim) return;

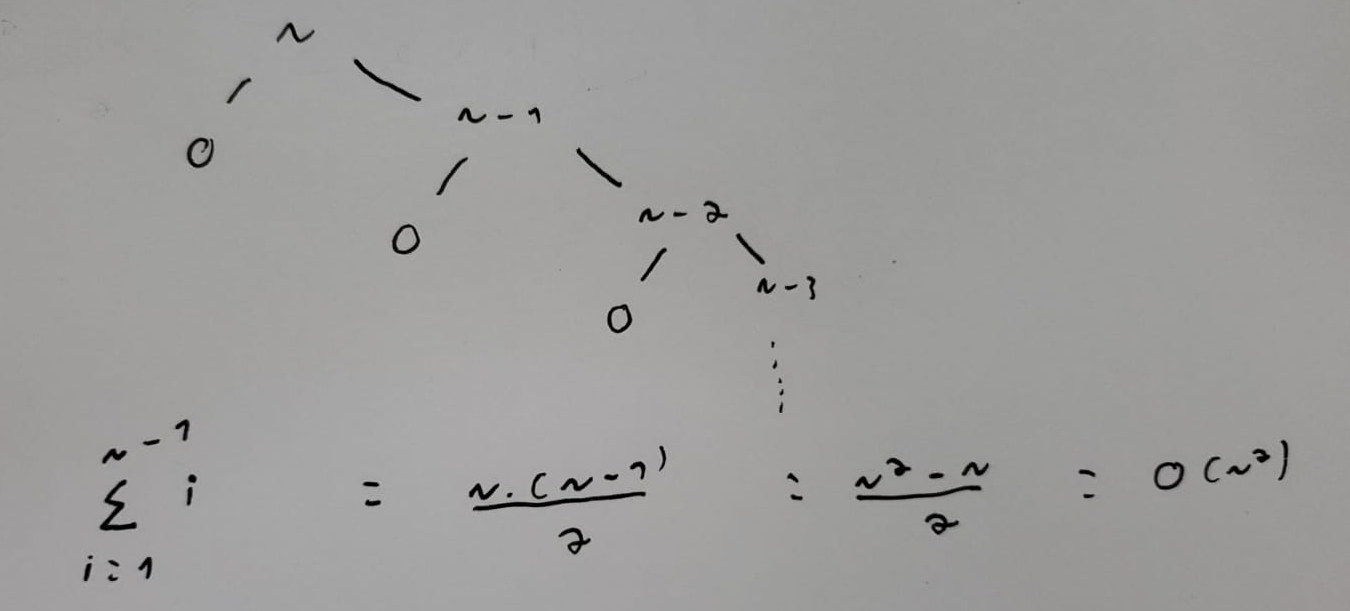
int pivo = Partition(v, inicio, fim);

QuickSort(v, inicio, pivo);

QuickSort(v, pivo + 1, fim);

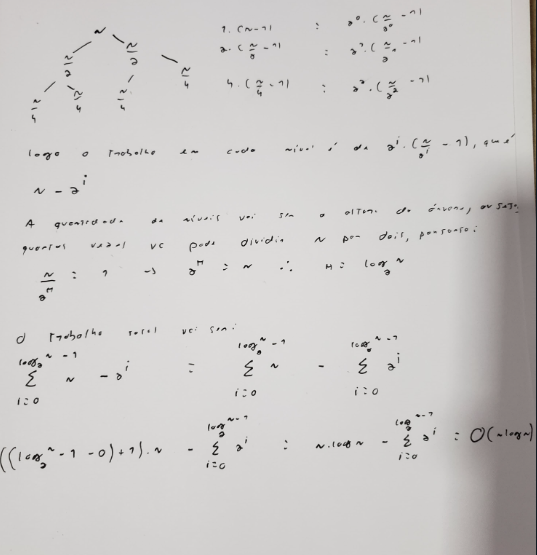
}

O pior caso seria quando vc sempre escolhe o pior pivo, que seria ou o maior ou o menor elemento do array, isso, pois, por exemplo, se sempre escolhermos o pivo como o menor elemento do array, a sua posição final será o início, logo, o vetor de tamanho n será subidividido em dois vetores, um de tamanho 0 e outro de tamanho n - 1, e se você continuar fazendo isso você terá:



O somatório é até n - 1, pois se vc tem um vetor de tamanho n, o maior número de comparações que vc terá que fazer vai ser de n - 1. Logo, quando o tamanho é n o trabalho é n - 1.

O melhor caso é quando vc sempre consegue dividir o vetor de tamanho n em dois vetores de tamanho n/2.



Vai até logn - 1, pois vc começa em 0