A ideia por traz do merge sort é dividir o array em 2 até que ele tenha tamanho 1, ou seja, é um vetor ordenado, ai ele usa um algoritmo para juntar dois vetores ordenados (que é mais simples do q ordenar um vetor), dessa forma ele vai voltando com os arryas ordenados e juntando eles.

void merge(int \*v, int inicio, int meio, int fim){

int n1 = meio - inicio + 1;

int n2 = fim - meio;

int v1[n1], v2[n2];

for (int i = 0; i < n1; i++){

v1[i] = v[i + inicio];

}

for (int i = 0; i < n2; i++){

v2[i] = v[i + meio + 1];

}

int i = 0;

int j = 0;

int k = inicio;

while (i < n1 && j < n2){

if (v1[i] < v2[j]){

v[k] = v1[i];

i++;

}else{

v[k] = v2[j];

j++;

}

k++;

}

while (i < n1){

v[k] = v1[i];

i++;

k++;

}

while (j < n2){

v[k] = v2[j];

j++;

k++;

}

}

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

if(inicio >= fim) return;

int meio = (inicio + fim) / 2;

mergesort(v, inicio, meio);

mergesort(v, meio + 1, fim);

merge(v, inicio, meio, fim);

}

O merge sort sempre vai ser O(n. logn), isso pois, considerando que o mergesort sempre consegue dividir o vetor em dois subvetores de tamanho n/2:

