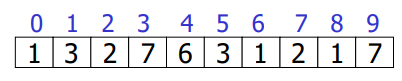
No counting sort vc vai inicialmente ver qual q é o maior valor de todos.

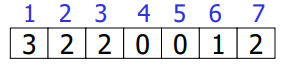
Após isso vc vai cirar um vetor que vai de 0 até o maior valor, de tal forma q esse vetor vai conter quantas vezes aquele valor se repete no vetor original.

Com isso vc consegue ordenar o vetor de forma a adicionar os valores de 0 a max quantas vezes eles aparecem no vetor que guarda quantas vezes ele se repetiu no original.

Sendo assim, se o vetor fosse, por exemplo:



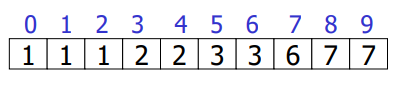
O vetor de contagem seria:



OBS: era para ter o o index 0 com o valor 0

E dessa forma o seu código vai (no vetor original)

adicionar o 0 0 vezes, depois adicionar o 1 3 vezes, adicionar o 2 duas vezes, adicionar o 3 3 vezes, adicionar o 4 0 vezes, adicionar o 5 0 vezes, aidiconar o 6 uma vez e depois adicionar o 7 duas vezes, ordenando assim o vetor:



Código:

void CountingSort(int \*v, int n){

int max = 0;

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

if (v[i] > max) max = v[i];

}

int contagem[max + 1];

for (int i = 0; i < max + 1; i++){

contagem[i] = 0;

}

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

contagem[v[i]] ++;

}

int contador = 0;

for (int i = 0; i < max + 1; i++){

for (int j = 0; j < contagem[i]; j++){

v[contador] = i;

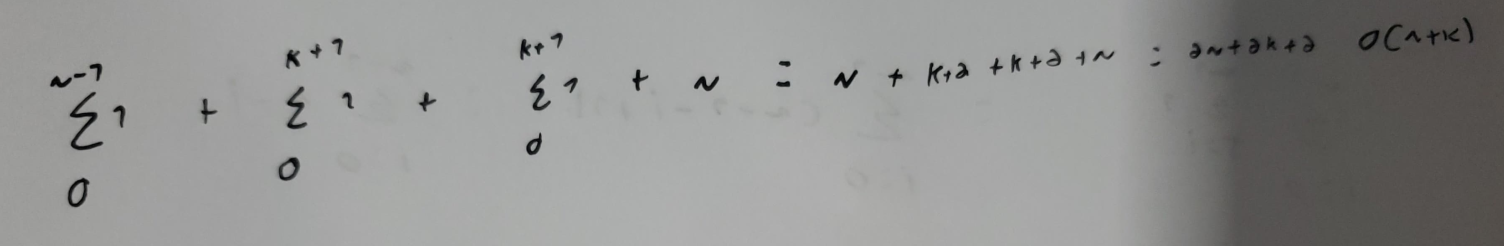
contador++;

}

}

}

Complexidade (como n se tem comaparações, a complexidade vai ser dada pela soma da complexidade de cada for):



A parte do código:

for (int i = 0; i < max + 1; i++){

for (int j = 0; j < contagem[i]; j++){

v[contador] = i;

contador++;

}

}

Será sempre N, pois a quantidade de vezes q esse código vai rodar vai ser equivalente a quantidade de vezes que o for de baixo vai rodar que será exatamente N, que o número de elementos no array original.

Ele consegue ser tão bom assim, pois ele é um método de ordenação que não é baseado em comparação.