HASHING DATA STRUCTURE

- Sort Array by Increasing Frequency

- nums = [1,1,2,2,2,3]

  We have to sort these Elements According to their Frequencies

  It means, We have to do something Like,

  We Have to store the Numbers with their Frequencies

1 -> 2

2 -> 3

3 -> 1

- For this, We have Data structure -> unordered_map

  Let's say unordered_map<int , int > umap ;

Ans: 3 1 1 2 2 2

Hello World

HASHING DATA STRUCTURE

- Sort Array by Increasing Frequency

- nums = [2,3,1,3,2]

  Ans: 1 3 3 2 2

2 -> 2

3 -> 2

1 -> 1

Hello World

HASHING DATA STRUCTURE

- 🟧 **Lambda Function in c++**

- 🟧 **[&] (int a , int b) { return Expression }**

[&] (parameters) { return Expression }

It's using a lambda function during sort. The lambda function specifies how to sort:

1. **if the two numbers have different frequencies, the one with smaller frequency goes first.**
2. **Otherwise, the one that is lexicographically greater goes first.**

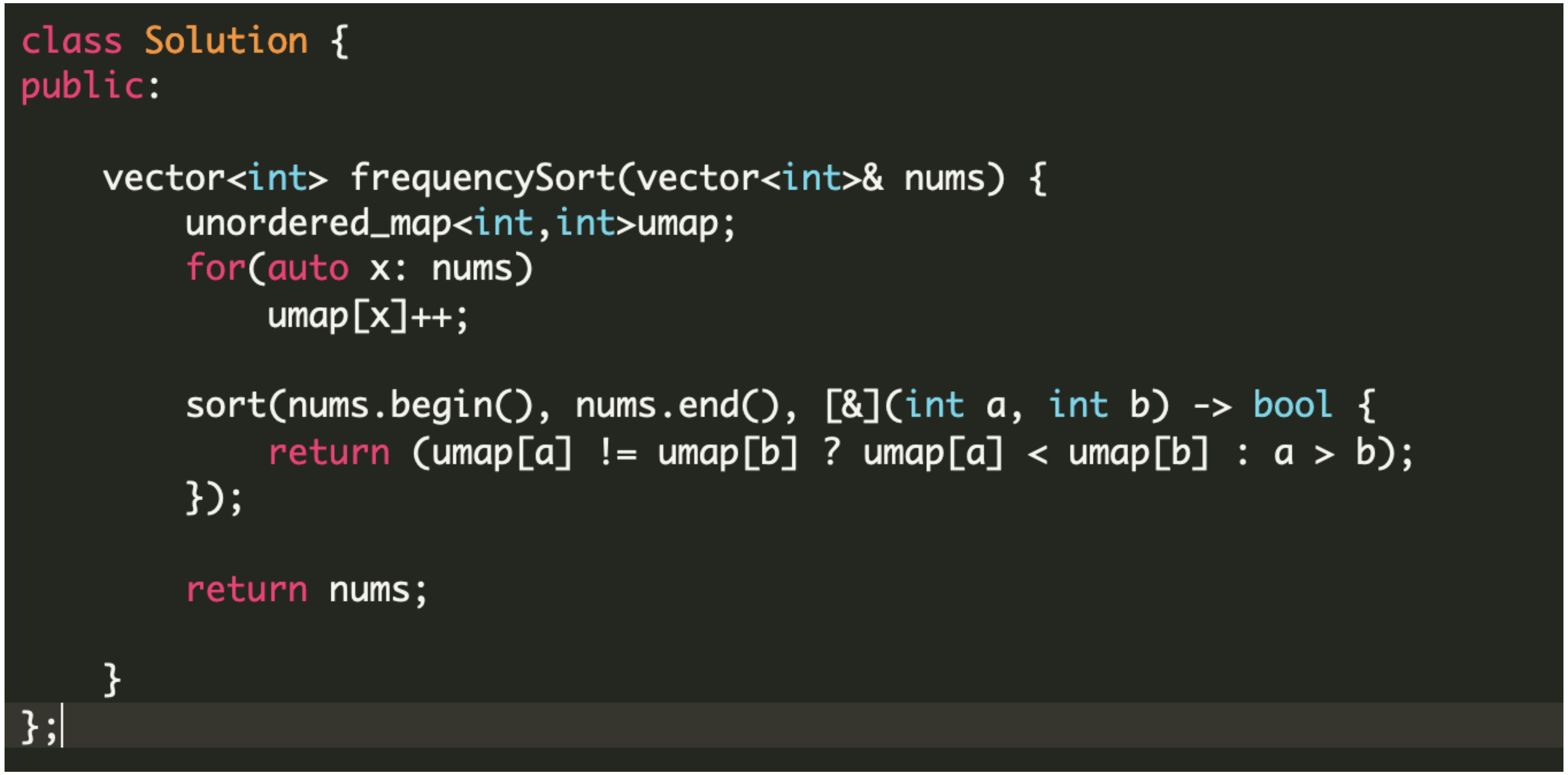
Hello World

HASHING DATA STRUCTURE

- **Lambda Function in c++**

- **[&] (int a , int b) { return Expression }**

  [&] (parameters) { return Expression }



```cpp
class Solution {
public:

    vector<int> frequencySort(vector<int>& nums) {
        unordered_map<int,int>umap;
        for(auto x: nums)
            umap[x]++;

        sort(nums.begin(), nums.end(), [&](int a, int b) -> bool {
            return (umap[a] != umap[b] ? umap[a] < umap[b] : a > b);
        });

        return nums;

    }
};
```

Hello World