1636. Sort Array by Increasing Frequency

Given an array of integers `nums`, sort the array in **increasing** order based on the frequency of the values. If multiple values have the same frequency, sort them in **decreasing** order.

Return the *sorted array*.

Example 1:

```
Input: nums = [1,1,2,2,2,3]
Output: [3,1,1,2,2,2]
Explanation: '3' has a frequency of 1, '1' has a frequency of 2, and '2' has a frequency of 3.
```

Example 2:

```
Input: nums = [2,3,1,3,2]
Output: [1,3,3,2,2]
Explanation: '2' and '3' both have a frequency of 2, so they are sorted in decreasing order.
```

Example 3:

```
Input: nums = [-1,1,-6,4,5,-6,1,4,1]
Output: [5,-1,4,4,-6,-6,1,1,1]
```

Constraints:

- `1 <= nums.length <= 100`
- `-100 <= nums[i] <= 100`

1636. Sort Array by Increasing Frequency

```
/*
    array sortiing: Lambda on array of pairs
   Time complexity: O(nlogn)
   Extra space complexity: O(n+2n+logn)
*/
class Solution {
public:
   vector<int> frequencySort(vector<int>& nums) {
      int n=nums.size();

      std::unordered_map<int,int> freq;
      for(auto& val: nums) freq[val]++;

      std::vector<std::pair<int,int>> nums_freq;
      for(auto& val: nums){
         int f=freq[val];
         nums_freq.push_back({val,f});
      }

      std::sort(nums_freq.begin(),nums_freq.end(),[](const std::pair<int,int>& p1,const
std::pair<int,int>& p2){
                                          if(p1.second==p2.second) return p1.first>p2.first;
                                          return p1.second<p2.second;
                                       });

      for(int i=0;i<n;i++) nums[i]=nums_freq[i].first;

      return nums;
   }
};
```

1636. Sort Array by Increasing Frequency

```
/*
   array sortiing: Lambda on array of frequency
   Time complexity: O(nlogn)
    Extra space complexity: O(n)
*/
class Solution {
public:
    vector<int> frequencySort(vector<int>& nums) {
        int n=nums.size();

        std::vector<int> freq(201,0);
        for(auto& val: nums) freq[val+100]++;

        std::sort(nums.begin(),nums.end(),[&](const int& a,const int& b){
                                            return freq[a+100]==freq[b+100]?a>b:freq[a+100]<freq[b+100];
                                        });

        return nums;
    }
};
```

1636. Sort Array by Increasing Frequency

```
/*
   Counting sort+relative sort
   Time complexity: O(n+m)
   Extra space complexity: O(n+m)
*/
class Solution {
   public:
      vector<int> relative_sort(vector<int>& arr1, vector<int>& arr2) {
         int mx=*std::max_element(arr1.begin(),arr1.end());
         std::vector<int> count(mx+1,0),count_remaining(mx+1,0);
         for(auto& v: arr1) {
            count[v]++;
            count_remaining[v]++;
         }

         for(auto&v: arr2) count_remaining[v]=0;

         int n=arr1.size();
         int m=arr2.size();

         count[arr2[0]]--;
         for(int i=1;i<m;++i) count[arr2[i]]+=count[arr2[i-1]];

         int x=count[arr2[m-1]];
         for(int i=0;i<=mx;++i){
            if(count_remaining[i]!=0){
               count[i]+=x;
               x=count[i];
            }
         }

         std::vector<int> ans(n);
         for(int i=n-1;i>=0;--i) ans[count[arr1[i]]--]=arr1[i];


         return ans;
      }
```

```
std::vector<int> frequencySort(std::vector<int>& nums) {
        int n=nums.size();
        int mi=*std::min_element(nums.begin(),nums.end());
        if(mi<0) for(int i=0;i<n;++i) nums[i]+=abs(mi);
        int mx=*std::max_element(nums.begin(),nums.end());
        std::vector<int> nums_freq(mx+1,0);
        for(auto& val: nums) nums_freq[val]++;

        std::vector<std::pair<int,int>> freq_arr;
        for(int i=0;i<=mx;i++) {
           int f=nums_freq[i];
           if(f!=0) freq_arr.push_back({f,i});
        }

        // Perform count sort on the frequency array freq_arr
        auto p=*std::max_element(freq_arr.begin(),freq_arr.end());
        int mxf=p.first;

        std::vector<int> freq(mxf+1,0);
        for(auto& p: freq_arr) freq[p.first]++;

        freq[0]--;
        for(int i=1;i<=mxf;++i) freq[i]+=freq[i-1];
        int m=freq_arr.size();
        std::vector<std::pair<int,int>> sorted_freq_arr(m);
        for(int i=0;i<m;++i) {
           sorted_freq_arr[freq[freq_arr[i].first]--]=freq_arr[i];
        }

        std::vector<int> ref_arr;
        for(auto& p: sorted_freq_arr){
           ref_arr.push_back(p.second);
        }

        vector<int> ans=relative_sort(nums,ref_arr);
        if(mi<0) for(int i=0;i<n;++i)ans[i]-=abs(mi);

        return ans;

    }
};
```