**[Sort Characters By Frequency](https://leetcode.com/problems/sort-characters-by-frequency/)**

Given a string s, sort it in **decreasing order** based on the **frequency** of the characters. The **frequency** of a character is the number of times it appears in the string.

Return *the sorted string*. If there are multiple answers, return *any of them*.

**Example 1:**

**Input:** s = "tree"

**Output:** "eert"

**Explanation:** 'e' appears twice while 'r' and 't' both appear once.

So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

**Example 2:**

**Input:** s = "cccaaa"

**Output:** "aaaccc"

**Explanation:** Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.

Note that "cacaca" is incorrect, as the same characters must be together.

**Example 3:**

**Input:** s = "Aabb"

**Output:** "bbAa"

**Explanation:** "bbaA" is also a valid answer, but "Aabb" is incorrect.

Note that 'A' and 'a' are treated as two different characters.

**Constraints:**

* 1 <= s.length <= 5 \* 105
* s consists of uppercase and lowercase English letters and digits.

CODE :

class Solution {

public:

    string frequencySort(string s) {

        auto cmp = [](const pair<char, int>& a, const pair<char, int>& b) {

            return a.second < b.second;

        };

        priority\_queue<pair<char, int>, vector<pair<char, int>>, decltype(cmp)> pq(cmp);

        unordered\_map<char, int> hm;

        for (char c : s) {

            hm[c]++;

        }

        for (const auto& entry : hm) {

            pq.push(make\_pair(entry.first, entry.second));

        }

        string result = "";

        while (!pq.empty()) {

            pair<char, int> p = pq.top();

            pq.pop();

            result.append(p.second, p.first);

        }

        return result;

    }

};

Link : <https://leetcode.com/problems/sort-characters-by-frequency/?envType=daily-question&envId=2024-02-07>