# 题目

给定一个字符串，请将字符串里的字符按照出现的频率降序排列。

**示例 1:**

输入:

"tree"

输出:

"eert"

解释:

'e'出现两次，'r'和't'都只出现一次。

因此'e'必须出现在'r'和't'之前。此外，"eetr"也是一个有效的答案。

**示例 2:**

输入:

"cccaaa"

输出:

"cccaaa"

解释:

'c'和'a'都出现三次。此外，"aaaccc"也是有效的答案。

注意"cacaca"是不正确的，因为相同的字母必须放在一起。

**示例 3:**

输入:

"Aabb"

输出:

"bbAa"

解释:

此外，"bbaA"也是一个有效的答案，但"Aabb"是不正确的。

注意'A'和'a'被认为是两种不同的字符。

# 分析

## 方法一：堆/优先队列

**思路：**

1、使用无序map存储字符和出现次数映射关系

2、在vector中存储pair对，利用vector自定义排序

**代码：**

class Solution {

public:

string frequencySort(string s) {

unordered\_map<char, int> ump;

for (const auto &c : s) {

++ump[c];

}

vector<pair<char, int>> vec;

for (const auto &m : ump) {

vec.push\_back(m);

}

sort(vec.begin(), vec.end(), [](const pair<char, int> &p1, const pair<char, int> &p2) { return p1.second > p2.second; });

string ret;

for (const auto &v : vec) {

ret += string(v.second, v.first);

}

return ret;

}

};

## 方法二：vector排序

**思路：**

**代码：**

class Solution {

public:

string frequencySort(string s) {

unordered\_map<char,int> mp;

for(auto ch : s)

{

mp[ch]++;

}

priority\_queue<pair<int,char>> pri\_que;

for(auto m : mp)

{

pri\_que.push(make\_pair(m.second,m.first));

}

string ret;

while(!pri\_que.empty())

{

auto ch = pri\_que.top();

pri\_que.pop();

ret.append(ch.first,ch.second);//拷贝的字符个数和首地址

}

return ret;

}

};