```
/*
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。
来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/sort-characters-by-frequency
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:
方法一:map+priority_queue

- 首先遍历一次字符串,用map数据结构存储所有出现过的字符和他们出现的次数;
- 利用priority_queue的大顶队数据结构,重载其传入数据类型和比较函数,建立大顶堆;
- 根据大顶堆的特性,依次取得大根堆的堆顶元素来组成返回值的子字符串,直到大顶堆为空.

方法一:C++_Map+priority_queue

```
class Solution
{


    /*大顶堆*/
    struct cmp
```

```
    {
        template<typename T, typename U>
            bool operator()(T const& left, U const& right)
            {
                if (left.second < right.second)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
    };


    public:
        string frequencySort(string s)
        {
            map<char,int>   mii                 ;
            int             i       =   0       ;
            string          ret_val             ;
            char            ch      =   0       ;
            int             fre     =   0       ;

            /*统计各字符串出现频次*/
            for(i = 0; i < s.size() ; i++)
            {
                if(mii.count(s[i]))
                {
                    mii[s[i]]++;
                }
                else
                {
                    mii[s[i]] = 1;
                }
            }

            priority_queue<pair<char,int>,vector<pair<char,int>>,cmp>
 pq(mii.begin(),mii.end());
            pair<char,int> pci;
            while(!pq.empty())
            {
                pci = pq.top();
                ret_val += string(pci.second,pci.first);
                pq.pop();
            }

            return ret_val;

        }
};

/*
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 cpp 提交中击败了88.90% 的用户
```

```
内存消耗 :11.2 MB, 在所有 cpp 提交中击败了25.16%的用户
*/
```

AlimyBreak
2019.11.02