1 [CSP-J 2025] 拼数 / number

题目描述

小 R 正在学习字符串处理。小 X 给了小 R 一个字符串 s，其中 s 仅包含小写英文字母及数字，且包含至少一个 $1 \sim 9$ 中的数字。小 X 希望小 R 使用 s 中的任意多个数字，按任意顺序拼成一个正整数。注意：小 R 可以选择 s 中相同的数字，但每个数字只能使用一次。例如，若 s 为 `1a01b`，则小 R 可以同时选择第 $1,3,4$ 个字符，分别为 $1,0,1$，拼成正整数 101 或 110；但小 R 不能拼成正整数 111，因为 s 仅包含两个数字 1。小 R 想知道，在他所有能拼成的正整数中，最大的是多少。你需要帮助小 R 求出他能拼成的正整数的最大值。

输入输出格式

输入：第一行，一个字符串 s，表示小 X 给小 R 的字符串。

输出：一行，一个正整数，表示小 R 能拼成的正整数的最大值。

输入示例	输出示例
5	5
290es1q0	92100

样例解释

样例1：字符串 $s =$ "5"，只有一个数字5，所以最大正整数就是5。

样例2：字符串 $s =$ "290es1q0"，包含数字 $2,9,0,1,0$。将这些数字从大到小排列得到 $9,2,1,0,0$，拼成的最大正整数为92100。

算法分析

1. 问题转化

   从字符串中提取所有数字字符，然后将这些数字按照从大到小的顺序排列，这样拼接起来的数字就是最大的正整数。

2. 关键步骤

   （1） 遍历字符串，统计每个数字出现的次数

   ```
   for (int i = 0; i < s.size(); i++) {
       if (s[i] >= '0' && s[i] <= '9') a[s[i] - '0']++;
   }
   ```

   （2） 从数字9到数字0，依次将每个数字拼接到结果字符串中

   ```
   for (int i = 9; i >= 0; i--) {
       while (a[i]-- > 0) res += i + '0';
   }
   ```

   （3） 输出结果字符串

3. 算法正确性证明

（1） 贪心策略的正确性

要得到最大的正整数，首先就是位数要尽可能多，意味着每一个数字都要用到，其次应该让高位数字尽可能大。因此，将数字从大到小排列是最优策略。

参考实现

```cpp
#include "bits/stdc++.h"
using namespace std;

using u64 = unsigned long long;
using i64 = long long;

// std::mt19937_64 rng {std::chrono::steady_clock::now().
    time_since_epoch().count()};
#define int long long
#define endl "\n"
constexpr i64 inf = 1e18;

void slu() {
    string s;
    cin >> s;
    vector<int> a(10, 0);
    for (int i = 0; i < s.size(); i++) {
        if (s[i] >= '0' && s[i] <= '9') a[s[i] - '0']++;
    }
    string res = "";
    for (int i = 9; i >= 0; i--) {
        while (a[i]-- > 0) res += i + '0';
    }
    cout << res << endl;
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);

    int t = 1;
    // cin >> t;

    while (t--) slu();
    return 0;
}
```

拓展思考

- 如果题目修改为输出最小正整数，该如何实现？