蓝桥杯国赛简单题

题目描述：

题目描述

L 星球游乐园非常有趣，吸引着各个星球的游客前来游玩。小蓝是 L 星球游乐园的管理员。

为了更好的管理游乐园，游乐园要求所有的游客提前预约，小蓝能看到系 统上所有预约游客的名字。每个游客的名字由一个大写英文字母开始，后面跟 00 个或多个小写英文字母。游客可能重名。

小蓝特别喜欢递增的事物。今天，他决定在所有预约的游客中，选择一部 分游客在上午游玩，其他的游客都在下午游玩，在上午游玩的游客要求按照预 约的顺序排列后，名字是单调递增的，即排在前面的名字严格小于排在后面的 名字。

一个名字 AA 小于另一个名字 BB 是指：存在一个整数 ii，使得 AA 的前 ii 个字母与 BB 的前 ii 个字母相同，且 AA 的第 i+ 1i+1 个字母小于 BB 的第 i+ 1i+1 个字母。（如果 AA 不存在第 i + 1i+1 个字母且 BB 存在第 i + 1i+1 个字母，也视为 AA 的第 i + 1i+1 个字母小于 BB 的第 i + 1i+1 个字母）

作为小蓝的助手，你要按照小蓝的想法安排游客，同时你又希望上午有尽 量多的游客游玩，请告诉小蓝让哪些游客上午游玩。如果方案有多种，请输出 上午游玩的第一个游客名字最小的方案。如果此时还有多种方案，请输出第一 个游客名字最小的前提下第二个游客名字最小的方案。如果仍然有多种，依此 类推选择第三个、第四个.....游客名字最小的方案。

输入描述

输入包含一个字符串，按预约的顺序给出所有游客的名字，相邻的游客名 字之间没有字符分隔。

其中有 ，每个名字的长度不超过 1010 个字母，输入的总长度不超 过 10^6106 个字母。

输出描述

按预约顺序输出上午游玩的游客名单，中间不加任何分隔字符。

输入输出样例

示例

> 输入

```
WoAiLanQiaoBei
```

> 输出

```
AiLanQiao
```

运行限制

- 最大运行时间：1s
- 最大运行内存: 128M

题目分析：本质是求最长递增子序列

贪心+二分查找+回溯(超出了运行时内存限制)

内存限制为**160M**；猜测可能是**dp**动态数组使之超出了内存；

关于**dp**动态数组的降维优化：

dp【i】表示当递增子序列最长为**i**时最后一各数的最小值

增加**pos[i]:**表示以字符**i**结尾的最长序列的长度

若要回溯求最长长度子序列中的最小值：即将**pos[i]==len**的值输出即可

```
#include <iostream>
#include<string>
#include<vector>
#include<unordered_map>
#include<algorithm>
#include<limits.h>
using namespace std;
```

```
class Solution {
    //哈希映射
    unordered_map<int, string>map1;
    unordered_map<string, int>map2;
    vector<int>datas;
    //dp动态规划
    vector<vector<int>>dp;//表示长度为i,递增长度为len的最后一个字符串的值
    vector<int>min_dp;


private:
    //适用于sort的字符串比较函数
    //比较函数中的等于好切记摸棱两可
    bool static cmp1(const string& s1,const string& s2) {
        int n1 = s1.size(), n2 = s2.size();
        int i = 0, j = 0;
        while (i < n1 && j < n2) {
            if (s1[i] == s2[j]) {
                ++i;
                ++j;
            }
            else {
                if (s1[i] < s2[j])return true;
                else return false;
            }
        }
        if (i == n1&&j!=n2)return true;
        else return false;
    }

public:
    string main_func(vector<string>strs) {
        vector<string>temp = strs;//不知此处编译是否有问题
        sort(temp.begin(), temp.end(), cmp1);


        for (int i = 0; i < (int)temp.size(); i++) {
            //map1[i] = temp[i];
            map2[temp[i]] = i;
        }
        for(auto&v:map2){
          map1[v.second]=v.first;
        }
        //将字符串转换成数字-以减少空间消耗
        for (auto& str : strs) {
            datas.push_back(map2[str]);
        }
```

```
    //动态规划初始化
    int n = strs.size();
    //dp.resize(n);
    min_dp.push_back(datas[0]);
    //dp[0].push_back(datas[0]);
    //动态规划开始
    for (int i = 1; i < n; i++) {
        //首先更新min_dp-用于求最长子序列的长度
        if (datas[i] > min_dp.back())min_dp.push_back(datas[i]);
        else {
            int begin = 0, end = min_dp.size() - 1, tag = -1, mid;
            while (begin <= end) {
                mid = (begin + end) >> 1;
                if (min_dp[mid] >= datas[i]) {
                    tag = mid;
                    end = mid - 1;
                }
                else begin = mid + 1;
            }
            min_dp[tag] = datas[i];


        }
        dp.push_back(min_dp);

    }


        //求结果
        int len = min_dp.size()-1;
        vector<int>ans;
        int target = min_dp.back();
        ans.push_back(target);
        for (int i = n - 1; i >= 0 && len >= 0; i--) {
            if (datas[i] == target) {

                if (len > 0 && i > 0) {
                    target = dp[i - 1][len - 1];
                    ans.push_back(target);
                }

                --len;
            }
        }

        reverse(ans.begin(), ans.end());
        string res;
        for (auto& data : ans) {
```

```
                res += map1[data];
            }
            return res;



    }

};
int main()
{
    //处理输入
    string input;
    cin >> input;
    //将输入分隔成字符串
    vector<string>strs;
    string temp;
    int n = input.size(), i = 0;
    while (i < n) {
        temp.clear();
        temp.push_back(input[i++]);
        while (i < n && input[i] <= 'z' && input[i] >= 'a') {
            temp.push_back(input[i++]);
        }
        strs.push_back(temp);
    }
    //建立实例-返回结果
    Solution A;
    cout << A.main_func(strs);
    // 请在此输入您的代码
    return 0;
}
```

启示代码：

```
#include<iostream>
#include<cmath>
#include<algorithm>
#include<cstring>
#include<string>
#include<map>
#include<set>
#include<vector>
#include<queue>
#define ll long long
#define maxn 1000010
using namespace std;
string inp;
string s[maxn];
```

```
string dp[maxn];
int lenn;
int pos[maxn];
int main()
{
    cin>>inp;
    int len=1;
    for(int i=0;i<inp.size();)
    {
        if(isupper(inp[i]))
        {
            do
            {
                s[len]+=inp[i++];
            }while(islower(inp[i]));
            len++;
        }
    }
    dp[1]=s[1];
    pos[1]=1;
    lenn=1;
    for(int i=1;i<len;i++)
    {
        if(s[i]>dp[lenn])
        {
            dp[++lenn]=s[i];
            pos[i]=lenn;
        }
        else
        {
            int x=lower_bound(dp+1,dp+1+lenn,s[i])-dp;
            dp[x]=s[i];
            pos[i]=x;
        }
    }
    string maxx="zzzzzzzzzz";
    vector<string> v;
    for(int i=len-1;i>=1;i--)
    {
        if(len==0)
        break;
        if(pos[i]==lenn && maxx>s[i])
        {
            lenn--;
            maxx=s[i];
            v.push_back(s[i]);
        }
    }
    for(int i=v.size()-1;i>=0;i--)
```

```
    {
        cout<<v[i];
    }
    cout<<endl;
}
```

代码优化：

```
#include <iostream>
#include<string>
#include<vector>
#include<unordered_map>
#include<algorithm>
#include<limits.h>
using namespace std;

class Solution {
private:
    //哈希映射
    unordered_map<int, string>map1;
    unordered_map<string, int>map2;
    vector<int>datas;
    //dp动态规划
    vector<int>min_dp;
    vector<int>pos;
public:
    string main_func(vector<string>strs) {
        vector<string>temp = strs;//不知此处编译是否有问题
        sort(temp.begin(), temp.end());
        //哈希
        for (int i = 0; i < (int)temp.size(); i++) {
            //map1[i] = temp[i];
            map2[temp[i]] = i;
        }
        for(auto&v:map2){
          map1[v.second]=v.first;
        }
        //将字符串转换成数字-以减少空间消耗
        for (auto& str : strs) {
            datas.push_back(map2[str]);
        }
        //动态规划初始化
        int n = strs.size();
        min_dp.push_back(datas[0]);
        pos.push_back(0);
        int len=0;
        //动态规划开始
        for (int i = 1; i < n; i++) {
            //首先更新min_dp-用于求最长子序列的长度
```

```
            if (datas[i] > min_dp.back()){
              min_dp.push_back(datas[i]);
              pos.push_back(++len);
            }
            else {
                int begin = 0, end = min_dp.size() - 1, tag = -1, mid;
                while (begin <= end) {
                    mid = (begin + end) >> 1;
                    if (min_dp[mid] >= datas[i]) {
                        tag = mid;
                        end = mid - 1;
                    }
                    else begin = mid + 1;
                }
                min_dp[tag] = datas[i];
                pos.push_back(tag);
            }

        }


            //求结果
            len = min_dp.size()-1;
            vector<int>ans;
            int target =INT_MAX;
            for (int i =n-1; i >= 0; i--) {
              if(len<0)break;

              if (pos[i]==len&&datas[i]<INT_MAX) {
                ans.push_back(datas[i]);
                target=datas[i];
                --len;
              }


            }

            reverse(ans.begin(), ans.end());
            string res;
            for (auto& data : ans) {
                res += map1[data];
            }
            return res;



    }

};
```

```
int main()
{
    //处理输入
    string input;
    cin >> input;
    //将输入分隔成字符串
    vector<string>strs;
    string temp;
    int n = input.size(), i = 0;
    while (i < n) {
        temp.clear();
        temp.push_back(input[i++]);
        while (i < n && input[i] <= 'z' && input[i] >= 'a') {
            temp.push_back(input[i++]);
        }
        strs.push_back(temp);
    }
    //建立实例-返回结果
    Solution A;
    cout << A.main_func(strs);
    // 请在此输入您的代码
    return 0;
}
```

反思总结：

1. 蓝桥杯的竞赛模式和力扣不一样
2. 需要自己写出所有的代码，包括核心函数，输入输出
3. 调试的时候不一定会给出所有的示例，有些示例是不可见的
4. 调试难度比较大

- 记主常用函数的头文件；**C**库，**STL**等。
- 同时考虑时间和空间的优化
- 牛啊！！！

蓝桥杯2020年A组省赛题：

第一部分：A,B,C三道简单题；

试题 A: 门牌制作

本题总分：5 分

【问题描述】

小蓝要为一条街的住户制作门牌号。

这条街一共有 2020 位住户，门牌号从 1 到 2020 编号。

小蓝制作门牌的方法是先制作 0 到 9 这几个数字字符，最后根据需要将字符粘贴到门牌上，例如门牌 1017 需要依次粘贴字符 1、0、1、7，即需要 1 个字符 0，2 个字符 1，1 个字符 7。

请问要制作所有的 1 到 2020 号门牌，总共需要多少个字符 2?

【答案提交】

这是一道结果填空的题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

枚举**1**到**2020**，分别计算每个数所含**2**的个数并相加！！！

试题 B: 既约分数

本题总分：5 分

【问题描述】

如果一个分数的分子和分母的最大公约数是 1，这个分数称为既约分数。

例如，$\frac{3}{4}$, $\frac{5}{2}$, $\frac{1}{8}$, $\frac{7}{1}$ 都是既约分数。

请问，有多少个既约分数，分子和分母都是 1 到 2020 之间的整数（包括 1 和 2020）?

【答案提交】

这是一道结果填空题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

学习知识点：快速求最大公约数和最小公倍数——辗转相除法

相关博客地址：https://blog.csdn.net/Little1Pudding/article/details/70578465?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216348813041678026252146 6%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163488130416780262521466&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-

70578465.first_rank_v2_pc_rank_v29&utm_term=%E6%80%8E%E4%B9%88%E5%BF%AB%E9%80%9F%E6%B1%82%E6%9C%80%E5%A4%A7%E5%85%AC%E7%BA%A6%E6%95%B0&spm=1018.2226.3001.4187

```
#include <stdio.h>
int main()
{
    int a,b,c,d,m,n;
    scanf("%d %d",&a,&b);
    m=a;n=b;
    while(1){//循环的辗转相除法
        c=a%b;
        a=b;
        b=c;
        if(b==0){
            break;
        }
    }
    //最小公倍数等于:把这两个数用最大公约数除得到的商相乘,再乘以最大公约数,就得到最小公倍数;
    c=m/a;d=n/a;//最小公倍数
    n=c*d*a; //的算法
    printf("%d %d\n",a,n);//循环的辗转相除法的结果
    return 0;
}
```


该题暴力枚举所有可能即可！！！

试题 C: 蛇形填数

本题总分：10 分

【问题描述】

如下图所示，小明用从 1 开始的正整数“蛇形”填充无限大的矩阵。

$$\begin{array}{cccccc} 1 & 2 & 6 & 7 & 15 & \ldots \\ 3 & 5 & 8 & 14 & \ldots & \\ 4 & 9 & 13 & \ldots & & \\ 10 & 12 & \ldots & & & \\ 11 & \ldots & & & & \\ \ldots & & & & & \end{array} \tag{1}$$

容易看出矩阵第二行第二列中的数是 5。请你计算矩阵中第 20 行第 20 列的数是多少？

【答案提交】

这是一道结果填空的题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

模拟即可！！！（建议写代码的时候通过加注释来帮助自己理清思路）

```
#include <iostream>
#include<vector>
using namespace std;
int main()
{
  vector<vector<int>>martix(40,vector<int>(40,0));
  martix[0][0]=1;
  bool turn=true;//true时从上到下
  int num=2;
  int flag=1;
  int i=0,j=1;//起始位置
  while(flag<40){
    martix[i][j]=num++;
    if(turn){
      //当从上到下
      if(i==flag){
        //到头了
        turn=!turn;//转向
        ++flag;
        i=flag;//加一操作
      }else{
        //没有到头
        ++i;
```

```
            --j;
        }
      }else{
        //从下到上
        if(j==flag){
          //到头了
          turn=!turn;//转向
          ++flag;
          j=flag;
        }else{
          //没有到头
          --i;
          ++j;
        }
      }
    }
    cout<<martix[19][19];
    // 请在此输入您的代码
    return 0;
}
```

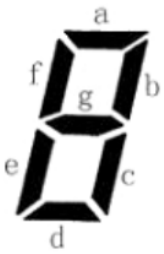
第二部分：中等题（开始上难度了）

试题 D: 七段码

本题总分：10 分

【问题描述】

小蓝要用七段码数码管来表示一种特殊的文字。



上图给出了七段码数码管的一个图示，数码管中一共有 7 段可以发光的二极管，分别标记为 a, b, c, d, e, f, g。

小蓝要选择一部分二极管（至少要有一个）发光来表达字符。在设计字符的表达时，要求所有发光的二极管是连成一片的。

例如：b 发光，其他二极管不发光可以用来表达一种字符。

例如：c 发光，其他二极管不发光可以用来表达一种字符。这种方案与上一行的方案可以用来表示不同的字符，尽管看上去比较相似。

例如：a, b, c, d, e 发光，f, g 不发光可以用来表达一种字符。

例如：b, f 发光，其他二极管不发光则不能用来表达一种字符，因为发光的二极管没有连成一片。

请问，小蓝可以用七段码数码管表达多少种不同的字符？

思路：将**a,b,c,d,e,f**分别看成图中的一个点，并将该七段码抽象成图；枚举所有可能的子集，并检验子集构成的子图是否时连通图

```
#include <iostream>
#include<vector>
#include<unordered_set>
#include<unordered_map>
#include<queue>
using namespace std;

int main()
{
    //暴力解法
    //构图--a，b,c,d,e,f分别代表一个结点
    //检查所有的子图中连通子图的个数
    int res = 0;
    vector<unordered_set<int>>edges(7);
    edges[0].insert(1);
    edges[0].insert(5);
```

```
    edges[1].insert(0);
    edges[1].insert(6);
    edges[1].insert(2);

    edges[2].insert(6);
    edges[2].insert(3);
    edges[2].insert(1);

    edges[3].insert(2);
    edges[3].insert(4);

    edges[4].insert(3);
    edges[4].insert(6);
    edges[4].insert(5);

    edges[6].insert(2);
    edges[6].insert(4);
    edges[6].insert(1);
    edges[6].insert(5);

    edges[5].insert(0);
    edges[5].insert(6);
    edges[5].insert(4);

    //暴力枚举所有可能并检查
    //没有结点的入度为0并不代表着时联通图
    for (int i = 1; i < (1 << 7); i++) {
        //cout << "序号: " << i << endl;
        vector<int>t;
        for (int j = 0; j < 7; j++) {
            if ((1 & (i >> j)) == 1) {
                t.push_back(j);
                //cout << j << " ";
            }
        }
        cout << endl;
        //建立子图
        int n = t.size();
        //cout << "n:" << n << endl;
        unordered_map<int, unordered_set<int>>map;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (edges[t[i]].count(t[j])) {
                    map[t[i]].insert(t[j]);
                    map[t[j]].insert(t[i]);
                }
            }
        }
        //深度优先遍历
```

```
            unordered_map<int, bool>m;
            for (int i = 0; i < n; i++)m[t[i]] = false;
            int begin = t[0];
            queue<int>que;
            que.push(begin);
            int num = 0;
            while (!que.empty()) {
                int b = que.front();
                que.pop();
                if (m[b])continue;
                m[b] = true;
                ++num;
                for (auto v : map[b]) {
                    if (m[v])continue;
                    que.push(v);

                }

            }
            //cout << "num:" <<num<< endl;
            if (num == n)++res;


        }

        cout << res;
        // 请在此输入您的代码
        return 0;
}
```

试题 E: 平面分割

本题总分：15 分

【问题描述】

20 个圆和 20 条直线最多能把平面分成多少个部分？

【答案提交】

这是一道结果填空的题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

这道题有点偏数学推导了：说实话有点难度！！！

```
import java.util.Scanner;
// 1:无需package
```

```
// 2: 类名必须Main, 不可修改

public class Main {
    public static void main(String[] args) {

      int result = doFunction(20,20);
       System.out.println(result);
    }


    //m个圆 n条直线
    private static int doFunction(int m,int n){

        //设有k个圆，当增加第k+1个圆时，最多2k个交点，故增加2k个部分，累加求和
        //知m个圆有m(m-1)+2个部分（包含最开始的区域）

        //第k条直线最多与原来包括原和k-1条直线的部分相交2m+k-1个交点，故增加2m+k个
        //区域，累加求和知增加了2mn+n（n+1）/2-1个区域，注意到当k=1时增加的是2m个
而不是2m+1个（因为此时
        //k-1=0）

        //相加即为结果
        return m*(m-1)+1+2*m*n+n*(n+1)/2;
    }
}
```

填空题部分总结：

1. 基本的知识还是要熟练掌握：如求最大公约数和最小公倍数等。
2. 填空题部分编程要严谨，由于没有题目示例，只能自己思忖着程序的对错，所以很容易会出错。
3. 填空题的最后一题还是有难度的；第四题的话，还可以有点优化

万能头文件：#include <bits/stdc++.h>

环境配置：

https://blog.csdn.net/zhangjiaji111/article/details/117551495?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216424968341678026193664 6%2522%252C%2522scm%2522%253A%25222014071 3.130102334.pc%255Fall.%2522%257D&request_id=164249683416780261936646&biz_id=0&utm_medium=distribute.

pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-117551495.first_rank_v2_pc_rank_v29&utm_term=%E8%93%9D%E6%A1%A5%E6%9D%AF%E7%9A%84%E7%8E%AF%E5%A2%83%E9%85%8D%E7%BD%AE&spm=1018.2226.3001.4187