题目描述：

某车站为了方便管理，决定根据目的地以及出发时间的不同对车辆时刻表进行分组管理。要求：给定一个时刻表，同一个目的地的车辆必须分配在同一组内，分组不能打乱时刻表的车次顺序，即各个分组之间出发时间有序。请对时刻表尽可能多的分组，按出发时间早晚作为输出顺序。

输入

时刻表内容：aabbcddc，a,b,c,d为目的地，字母出现的先后顺序为出发时间的先后顺序

输出

输出各个分组的长度，以空格相隔，输出顺序与时刻表的出发顺序一致

样例输入

aabbcddc

样例输出

2,2,4

提示

aabbcddc 可分为aa,bb,cddc三组，目的地相同的车辆分配在了一组，同时，aa分组出发时间早于bb分组，bb分组早于cddc分组，所以输出结果为2，2，4。若分为一组，aabbcddc，则不满足题目中尽可能多的分组这一要求。输入不为空

规则

请尽量在全场考试结束10分钟前调试程序，否则由于密集排队提交，可能查询不到编译结果
点击“调试”亦可保存代码
编程题可以使用本地编译器，此页面不记录跳出次数

```
#include <bits/stdc++.h>
using namespace  std;

// aabbcddc
int main()
{
    string s;
    cin >> s;
    int last = -1;
    int last_idx_arr[300];
    vector<int> ans;
    for(int i = 0; i < s.size(); i++) {
        last_idx_arr[s[i]] = i;
```

```
    }
    int pos = 0, cnt = 0;
    while(pos < s.size()) {
        cnt = 1;
        last = max(last, last_idx_arr[s[pos]]);
        while(pos < last) {
                        ++pos, ++cnt;
            last = max(last, last_idx_arr[s[pos]]);
        }
        ans.push_back(cnt);
                ++pos;
        cnt = 1;
    }
    cout << ans[0];
    for(int i = 1; i < ans.size(); ++i) {
        cout << "," << ans[i];
    }
    cout << endl;
    system("pause");
    return 0;
}
```

题目描述：

ROC-AUC是一种常用的模型评价指标，它是ROC曲线下的面积。现在已知样本数据集的真实值(1为正样本，0为负样本)和某二分类器在该样本数据集上的预测值（属于正样本的概率，且各不相同），求ROC-AUC，精确到小数点后两位。

输入

第1行为训练样本的数量N。
第2到N+1行的每行为样本的实际类别与预测值，以空格分隔。

输出

计算ROC-AUC，精确到小数点后两位。

样例输入

```
10
1 0.90
0 0.70
1 0.60
1 0.55
0 0.52
1 0.40
0 0.38
0 0.35
1 0.31
0 0.10
```

样例输出

```
0.68
```

提示

可以按定义计算，也可以计算随机挑选一对正负样本，正样本的预测值大于负样本的预测值的概率。

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


int main()
{
        vector<double> p_score, n_score;
        double x;
        int n, y;
        cin >> n;
        for (int i = 0; i < n; ++i) {
                cin >> y >> x;
```

```
                if(y) p_score.push_back(x);
                else n_score.push_back(x);
        }
        sort(p_score.begin(), p_score.end());
        sort(n_score.begin(), n_score.end(), greater<double>());
        int cnt = 0, i, j;
        for(i = 0; i < p_score.size(); i++) {
                for(j = 0; j < n_score.size(); j++) {
                        if(p_score[i] > n_score[j]) {
                                break;
                        }
                }
                cnt += n_score.size() - j;
        }
        printf("%.2lf\n", cnt * 1.0 / p_score.size() / n_score.size());
    //system("pause");
    return 0;
}
// 10
// 1 0.90
// 0 0.70
// 1 0.60
// 1 0.55
// 0 0.52
// 1 0.40
// 0 0.38
// 0 0.35
// 1 0.31
// 0 0.10
```

题目描述：

用户出行旅游通常都是一个闭环，即从某地出发游览各个景点，最终回到出发地。已知各个景点间的通行时间。请你设计一套算法，当用户给定出发地以及景点后，给出一条游览路线，使得用户能够不重复的游历每个景点并返回出发地的总通行时间最短，计算该最短通行时间。假设所有用户的出发地都是0

输入

第一行：出发地与景点的总个数 n
第二行：景点间的直达路线的个数m
其后m行：各个景点的通行时间a　b　t
表示a地与b地之间的通行时间是t。

输出

不重复游览完所有景点并返回出发地的最短游览时间T
若不存在这样的游览路线，返回-1

样例输入

```
4
6
0 1 4
0 2 3
0 3 1
1 2 1
1 3 2
2 3 5
```

样例输出

```
7
```