

# 笔试-字节跳动-180812

共 5 道编程题

## Reference

- [官方题解](#)（只有思路）

## Index

- [1. 世界杯开幕式](#)
- [2. 文章病句标识](#)
- [3. 积分卡牌游戏](#)
- [4. 区间最大最小值 TODO](#)
- [5. 直播爱好者](#)

## 1. 世界杯开幕式

### ■ 题目描述

世界杯开幕式会在球场C举行，球场C的球迷看台可以容纳 $M*N$ 个球迷。在球场售票完成后，现官方想统计此次开幕式一共有多少个球队球迷群体，最大的球队球迷群体有多少人。

经调研发现，球迷群体在选座时有以下特性：

- 同球队的球迷群体会选择相邻座位，不同球队的球迷群体会选择不相邻的座位。（注解：相邻包括前后相邻、左右相邻、斜对角相邻。）
- 给定一个 $M*N$ 的二维球场，0代表该位置没有坐人，1代表该位置已有球迷，希望输出球队群体个数 $P$ ，最大的球队群体人数 $Q$ 。

### 输入描述：

第一行，2个数字， $M$ 及 $N$ ，使用英文逗号分割；  
接下来 $M$ 行，每行 $N$ 的数字，使用英文逗号分割；

### 输出描述：

一行，2个数字， $P$ 及 $Q$ ，使用英文逗号分割；  
其中， $P$ 表示球队群体个数， $Q$ 表示最大的球队群体人数

**示例1**

输入

复制

```

10,10
0,0,0,0,0,0,0,0,0,0
0,0,0,1,1,0,1,0,0,0
0,1,0,0,0,0,0,1,0,1
1,0,0,0,0,0,0,0,1,1
0,0,0,1,1,1,0,0,0,1
0,0,0,0,0,0,1,0,1,1
0,1,1,0,0,0,0,0,0,0
0,0,0,1,0,1,0,0,0,0
0,0,1,0,0,1,0,0,0,0
0,1,0,0,0,0,0,0,0,0

```

输出

复制

```

6,8

```

**备注:**

数据范围:

 $0 < M < 1000$ 
 $0 < N < 1000$ 
**思路**

- dfs 搜索联通区域
- 原题只要搜索 4 个方向，这里改为搜索 8 个方向

**Code(Python)**

```
M, N = list(map(int, input().split(',')))
```

```
book = []
```

```
for i in range(M):
```

```
    line = list(map(int, input().split(',')))
```

```
    book.append(line)
```

```
class Solution:
```

```
    def __init__(self, pos):
```

```
        self.pos = pos
```

```
        self.cnt = 0 # 记录当前区域的人数
```

```
        self.dp = [] # 保存所有区域的人数，返回其长度，及其中的最大值
```

```
def dfs(self, i, j):
    if 0 <= i < M and 0 <= j < N:
        if self.pos[i][j] == 1:
            self.cnt += 1
            self.pos[i][j] = 0 # 遍历过的点就置 0，避免重复搜索
            # 八个方向搜索
            self.dfs(i - 1, j)
            self.dfs(i + 1, j)
            self.dfs(i, j - 1)
            self.dfs(i, j + 1)
            self.dfs(i - 1, j - 1)
            self.dfs(i + 1, j + 1)
            self.dfs(i + 1, j - 1)
            self.dfs(i - 1, j + 1)

def solve(self):
    for i in range(M):
        for j in range(N):
            if self.pos[i][j] == 1:
                self.cnt = 0 # 每新找到一个区域就清零人数，重新计数
                self.dfs(i, j) # 深度优先搜索每个点
                if self.cnt > 0:
                    self.dp.append(self.cnt)
    return len(self.dp), max(self.dp)

s = Solution(book)
P, Q = s.solve()
print(str(P) + ',' + str(Q))
```

## 2. 文章病句标识

---

## ■ 题目描述

为了提高文章质量，每一篇文章（假设全部都是英文）都会有 $m$ 名编辑进行审核，每个编辑独立工作，会把觉得有问题的句子通过下标记记录下来，比如 $[1,10]$ ，1表示病句的第一个字符，10表示病句的最后一个字符。也就是从1到10这10个字符组成的句子，是有问题的。

现在需要把多名编辑有问题的句子合并起来，送给总编辑进行最终的审核。比如编辑A指出的病句是 $[1,10]$ ， $[32,45]$ ；B编辑指出的病句是 $[5,16]$ ， $[78,94]$ ，那么 $[1,10]$ 和 $[5,16]$ 是有交叉的，可以合并成 $[1,16]$ ， $[32,45]$ ， $[78,94]$

### 输入描述:

编辑数量 $m$ ，之后每行是每个编辑的标记的下标集合，第一个和最后一个下标用英文逗号分隔，每组下标之间用分号分隔

### 输出描述:

合并后的下标集合，第一个和最后一个下标用英文逗号分隔，每组下标之间用分号分隔。返回结果是从从小到大递增排列

### 示例1

输入

复制

```
3
1,10;32,45
78,94;5,16
80,100;200,220;16,32
```

输出

复制

```
1,45;78,100;200,220
```

### 备注:

数据范围:

$m < 2^{16}$

下标数值  $< 2^{32}$

每个编辑记录的下标  $< 2^{16}$ 组

## 思路

- 区间合并
- 排序 + 贪心

对  $[l1, r1]$ ,  $[l2, r2]$ , 如果  $r1 > l2$ , 则  $r1 = \max(r1, r2)$

## Code(Python)

```
# 输入处理
m = int(input())

tmp = []
for _ in range(m):
    line = [list(map(int, item.split(','))) for item in input().split(';')]
    tmp.extend(line) # 将所有病句存在一起

# 排序, 按每段病句 [l, r] 的第一个位置 l 排序
tmp = sorted(tmp, key=lambda x: x[0])

ret = [tmp[0]]
for item in tmp[1:]:
    if ret[-1][1] >= item[0]: # 贪心: 对 [l1,r1], [l2,r2], 如果 r1 > l2, 则 r1 = max(r1, r2)
        ret[-1][1] = max(ret[-1][1], item[1])
    else:
        ret.append(item)

# 输出处理
s = ''
for item in ret[:-1]:
    s += str(item[0]) + ',' + str(item[1]) + ';'
s += str(ret[-1][0]) + ',' + str(ret[-1][1])
print(s)
```

## 3. 积分卡牌游戏

## ■ 题目描述

小a和小b玩一个游戏，有n张卡牌，每张上面有两个正整数x, y。  
取一张牌时，个人积分增加x，团队积分增加y。  
求小a，小b各取若干张牌，使得他们的个人积分相等，且团队积分最大。

### 输入描述:

第一行n  
接下来n行，每行两个正整数x, y

### 输出描述:

一行一个整数  
表示小a的积分和小b的积分相等时，团队积分的最大值

### 示例1

输入

复制

```
4
3 1
2 2
1 4
1 4
```

输出

复制

```
10
```

### 备注:

数据范围:

$0 < n < 100$

$0 < x < 1000$

$0 < y < 1e6$

## 思路

- 动态规划
- **DP 定义:**  $d[i][j]$  := 前 i 张牌，两人所选择的牌的差值为 j 时的最大值

- 转移方程

$$d[i][j] = \max(d[i-1][j], d[i-1][j-x[i]] + y[i], d[i-1][j+x[i]] + y[i])$$

## Code (90%)

```
# 输入处理
n = int(input())
x, y = [], []
for i in range(n):
    _x, _y = list(map(int, input().split()))
    x.append(_x)
    y.append(_y)

xy = list(zip(x, y))
xy = sorted(xy, key=lambda t: t[1])

ret = 0
if sum(x) % 2 == 0: # 如果所有 x 的和为偶数
    print(sum(y)) # 直接输出所有 y 的和
else:
    for i in range(len(xy)):
        if xy[i][0] % 2 == 1: # 去掉 x 中为奇数的那一项
            ret = sum([xy[j][1] for j in range(len(xy)) if j != i])
            print(ret)
            break
```

- 这段代码能过 90% 真是运气

## 4. 区间最大最小值 TODO

### ■ 题目描述

两个长度为 $n$ 的序列 $a, b$

问有多少个区间 $[l, r]$ 满足

$\max(a[l, r]) < \min(b[l, r])$

即 $a$ 区间的最大值小于 $b$ 区间的最小值

数据范围:

$n < 1e5$

$a_i, b_i < 1e9$

输入描述:

第一行一个整数 $n$

第二行 $n$ 个数, 第 $i$ 个为 $a_i$

第三行 $n$ 个数, 第 $i$ 个为 $b_i$

输出描述:

一行一个整数, 表示答案

### 示例1

输入

复制

```
3
3 2 1
3 3 3
```

输出

复制

```
3
```

## 5. 直播爱好者



## ■ 题目描述

小明在抖音里关注了 $N$ 个主播，每个主播每天的开播时间是固定的，分别在 $s_i$ 时刻开始开播， $t_i$ 时刻结束。小明无法同时观看两个主播的直播。一天被分成了 $M$ 个时间单位。请问小明每天最多能完整观看多少场直播？

### 输入描述:

第一行一个整数，代表 $N$

第二行一个整数，代表 $M$

第三行空格分隔的 $N*2$ 个整数，代表 $s, t$

### 输出描述:

一行一个整数，表示答案

### 示例1

输入

复制

```
3
10
0 3 3 7 7 0
```

输出

复制

```
3
```

### 示例2

输入

复制

```
3
10
0 5 2 7 6 9
```

输出

复制

```
2
```

**备注:**

数据范围:

 $1 \leq N \leq 10^5$  $2 \leq M \leq 10^6$  $0 \leq s_i, t_i < M$  ( $s_i \neq t_i$ ) $s_i > t_i$  代表时间跨天, 但直播时长不会超过一天。**思路**

- 贪心选择结束时间最早的直播

Code: 未测试

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

int main() {
    int n, m;
    cin >> n >> m;

    vector<pair<int, int>> book;
    for(int i=0; i<n; i++){
        int l, r;
        scanf("%d%d", &l, &r);
        if(l > r)           // 坑点: 可能存在第二天的情况
            r += m;
        book.push_back({r, l}); // 把结束时间存在首位, 排序时避免重新定义比较方法
    }

    sort(book.begin(), book.end()); // 按结束时间排序

    int ret = 0;
    int r = 0; // 保存当前结束时间
    for (int i=0; i<n; i++) {
        if (book[i].second > m)           // 只能在当天看完
            continue;
        if (r < book[i].second) {        // 如果当前直播在上一个直播结束之后开始
            ret += 1;
            r = book[i].first;           // 更新结束时间
        }
    }

    cout << ret << endl;
    return 0;
}
```

