第十七届东北地区大学生程序设计竞赛题解

北京邮电大学

BUPT

2023 年 5 月 14 日

Overview

- 首先为本场比赛出现的各种锅致以最诚恳的歉意

Overview

- 首先为本场比赛出现的各种锅致以最诚恳的歉意
- 本场比赛共有 178 支队伍参加

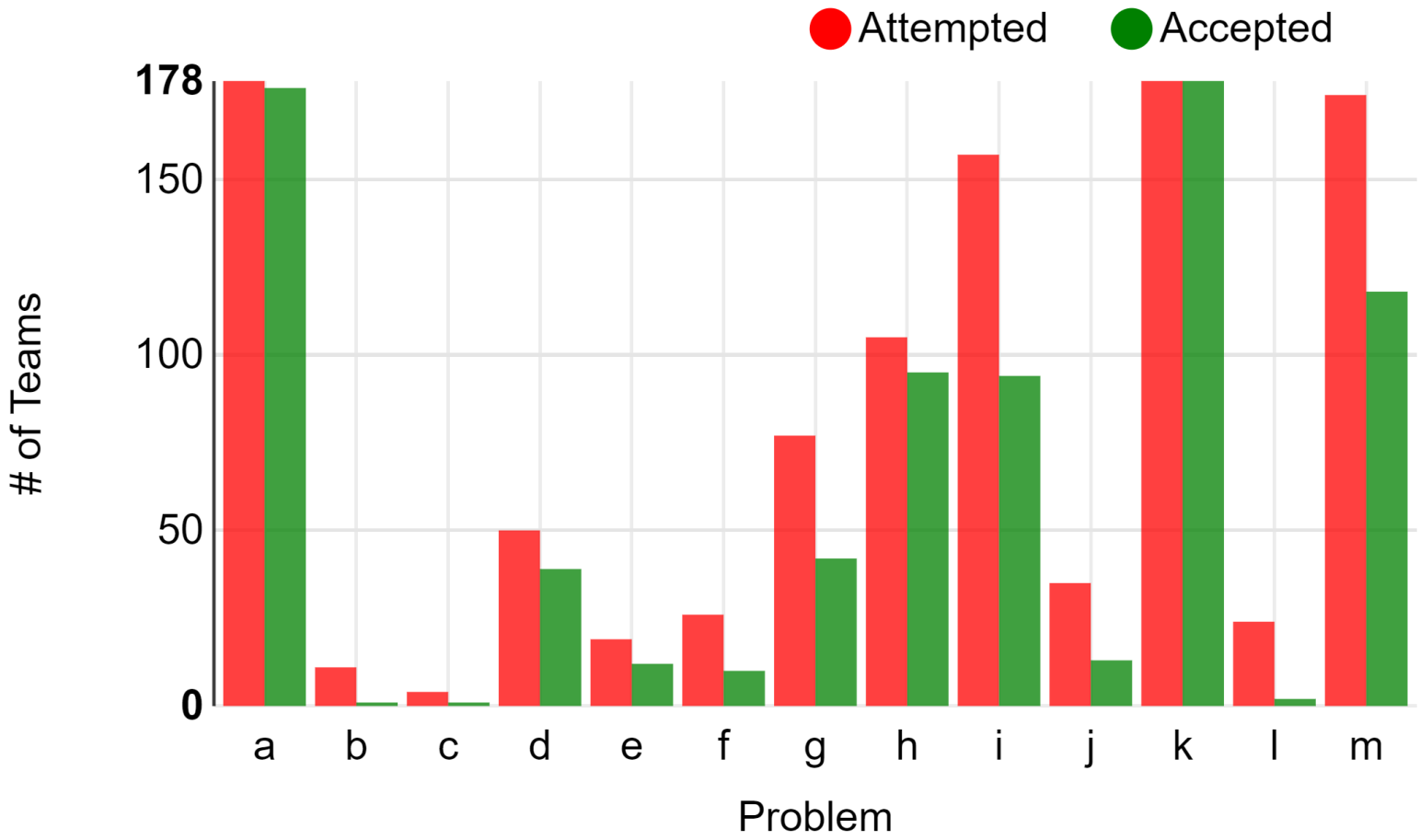
Overview

- 首先为本场比赛出现的各种锅致以最诚恳的歉意
- 本场比赛共有 178 支队伍参加
- **没有队伍爆零**

Overview

- 首先为本场比赛出现的各种锅致以最诚恳的歉意
- 本场比赛共有 178 支队伍参加
- **没有队伍爆零**
- 共产生 2879 个提交，783 个 Accepted 提交

Overview


Attempted
Accepted
178
150
100
50
0
# of Teams
a
b
c
d
e
f
g
h
i
j
k
l
m
Problem

A. Cask Effect

显然，将最长的木板给一部分给最短的木板即可，使给完之后两者长度相等。

因为不给最短的木板答案就不会更优，而显然拿最长的木板来给是最好的。如果最后两者长度不等，一定不会比让两者长度相等更优，因为答案只取决于最短的木板的长度。

另外，小数点后只可能是 $.0$ 或 $.5$，如果有精度问题，其实可以手动判断一下。

B. Problem B

我们根据 k 的大小不同分情况讨论。

首先，如果 $k \geq 2$，显然此时必然一个装置都不需要启动，方案数为 1.

一次拐弯时，可以发现，装置的设置方案满足题目要求的充要条件为：在所有 n 行中，至少有 $n-1$ 行，其中每一行都至少有一个装置启动了；在所有 m 列中，至少有 $m-1$ 列，其中每一列都至少有一个装置启动了。

换句话说，最多有一行是空的，且最多有一列是空的。

B. Problem B

考虑如何证明这件事。

不妨假设你当前从某一行的左端进入矩阵，那么讨论你的目标出口，有以下几种情况：

1. 出口在同一行正对面，那什么装置都不需要。
2. 出口在某一列的两端之一，那么也什么装置都不需要，因为转一次必然可以到达。
3. 出口在另一行的两端之一。

B. Problem B

稍加思考可以发现，对于不同的两行，只要任意一行中存在启动的装置，那么这两行的共 4 个端点出入口可以两两互相到达。

也就是说，对于任意两行，都至少有一行不为空。那么对所有 n 行来看，要么所有行都不为空，要么有且只有一行为空。对于列也同理。

于是我们可以讨论行和列每个方向分别空出来几行，分别都空在什么位置。具体需要根据 n 和 m 是否相等分别讨论。

B. Problem B

排除了空行空列之后，我们得到以下子问题：对于一个 n 行 m 列的矩阵，每行每列都至少有一个装置启动，且启动装置最少的方案数有多少?

设这个问题的答案为 $F(n, m)$，不妨假设 $n \geq m$，那么相当于每列至少开启一个装置，每行恰好开启一个装置，此时数量最少，为 n 个。

考虑每列开了多少个，假设 m 列分别开了 $a_1, a_2, \ldots, a_m$ 个，那么分配给每行排列组合的方案数就是 $\frac{n!}{a_1!a_2!\ldots a_m!}$，不难想到用 EGF 求解。

B. Problem B

设每列的 EGF 为 $f(x) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$，（注意，由于不能不开，所以没有 1 这一项），那么答案就是 $f^m(x)[n] \times n!$，即：

$$\begin{aligned} F(n,m) &= f^m(x)[n] \times n! = (e^x - 1)^m[n] \times n! \\ &= n! \left(\sum_{i=0}^{m} \binom{m}{i} (-1)^{m-i} e^{ix} \right) [n] \\ &= n! \sum_{i=0}^{m} \binom{m}{i} (-1)^{m-i} \frac{i^n}{n!} \\ &= \sum_{i=0}^{m} \binom{m}{i} (-1)^{m-i} i^n \end{aligned}$$

可以在 $O(m \log n)$ 复杂度下求解。

B. Problem B

将所有讨论的情况合并，那么最终答案为：

$$ans = \begin{cases} nF(n-1,m) + nmF(n-1,m-1) & n > m \\ n^2F(n-1,m-1) & n = m \end{cases}$$

B. Problem B

当小车一次都不能拐弯时，根据题目的要求，我们从任意行和列都应该能够到达其它任意的行和列，且从某一行或列到其他行或列的方法只有使用装置。所以我们可以把每一行每一列看成点，行于列之间的交叉（也就是每个单元格）看成将对应的行和列连接到一起的边。在此基础上，我们得到一张 n+m 个点的完全二分图，在这个图上求生成树计数即可。

完全二分图的生成树计数是一个经典问题，答案为 $n^{m-1}m^{n-1}$，可以用矩阵树定理或者 prufer 序列等多种方法证明。

B. Problem B

总时间复杂度 $O(n \log n)$。

C. Abstract Painting

从下到上扫描整个图形，在扫描过程中维护一个并查集以及与扫描线相交的竖直线段集合。

- 当扫描到竖直线段的下端点时，将该竖线加入集合，连通块编号为竖直线段的编号。
- 当扫描到竖线的上端点时，将该竖线从集合中移除。
- 当扫描到水平线段时，将集合中水平线段范围内的所有竖线的编号在并查集中与水平线段的编号合并。

C. Abstract Painting

然而上面的做法显然无法通过本题。

注意到将水平线段范围内的所有竖线与水平线段合并到一个连通块后可以将这些竖线作为一个整体看待，这启发我们维护竖线的连通块相同的区间，这样就可以将其作为一个整体来进行后续的合并。

一条竖线刚被加入时是单独的一个区间，其连通块编号为该竖线的编号。将水平线段范围内的所有竖线合并时将范围内的所有区间合并，其连通块编号为水平线段的编号。只有在一个区间的内部插入一条竖线时才会将一个区间拆成三个线段，因此自始至终只会出现 $O(n)$ 个区间，因此所有合并操作总共只涉及 $O(n)$ 个区间，从而保证了时间复杂度。

C. Abstract Painting

以上只是一个大致的思路，下面给出两种具体的实现方法。

C. Abstract Painting

以上只是一个大致的思路，下面给出两种具体的实现方法。

做法一：

离散化 + 线段树。将横坐标离散化后放入线段树。线段树的每个节点保存一个 color 值，color 值为 0 表示节点范围内没有竖线；值为 -1 表示节点范围内有属于不同连通块的竖线；值为正数表示节点范围内有竖线且全部属于编号为 color 的连通块。插入、删除竖线即为进行单点修改，而合并操作即为区间查询 + 修改，若遇到 color 值为 -1 的节点需要同时进入两棵子树。

C. Abstract Painting

做法二:

std::set。使用两个 set，一个维护集合中竖线的横坐标，另一个维护连通块编号相同的区间。插入竖线时向第一个 set 插入一个横坐标，向第二个 set 插入一个区间（有可能需要分割一个原有的区间)。删除竖线时将其横坐标从第一个 set 中删除，并检查其所在区间内是否还有竖线，如果没有竖线就将其所在竖线从第二个 set 中删除。合并操作只要在第二个 set 上修改即可。

上述两个做法的时间复杂度均为 $O(n \log n)$。

不管使用什么写法，本题都有比较多的细节，需要仔细思考，耐心实现。

D. Concrete Painting

对于一个位置，假如被 k 条线段覆盖，那么就会在 $2^k - 1$ 个子集内被统计到，贡献为区间长度乘以 $(2^k - 1) \times 2^{n-k}$。

D. Concrete Painting

对于一个位置，假如被 k 条线段覆盖，那么就会在 $2^k - 1$ 个子集内被统计到，贡献为区间长度乘以 $(2^k - 1) \times 2^{n-k}$。

将所有端点排序，对所有相邻两个端点之间的区域进行这样的判断，即可求出答案。

E. Triangle Pick

一道比较纯粹的计算几何题。

对于每次询问，暴力遍历所有三角形，求出交点并求出原点与交点的距离，统计出答案即可。

至于求交点和判断交点是否在三角形内，计算几何方法千千万，只要细心实现，就总能通过的。

F. MPFT

可以建立一个链表来记录当前在群聊中的人，每当有人加入群聊或发消息时将其移动到链表头部，这样链表尾部的人必然是最后一次发言时间最早的人，这就是经典的 LRU 链表算法。

同时对每个人记录一个最近 T 时长的时间段内的发言的窗口，在时间推进过程中维护这个窗口，一旦窗口大小超过了 K 就将此人从链表中删除，同时将窗口置空。时间复杂度 $O(M)$。

F. MPFT

可以建立一个链表来记录当前在群聊中的人，每当有人加入群聊或发消息时将其移动到链表头部，这样链表尾部的人必然是最后一次发言时间最早的人，这就是经典的 LRU 链表算法。

同时对每个人记录一个最近 T 时长的时间段内的发言的窗口，在时间推进过程中维护这个窗口，一旦窗口大小超过了 K 就将此人从链表中删除，同时将窗口置空。时间复杂度 $O(M)$。

事实上，用其他数据结构代替链表，只要写的不要太慢，以 $O(M \log M)$ 的时间复杂度同样可以通过本题。

F. MPFT

可以建立一个链表来记录当前在群聊中的人，每当有人加入群聊或发消息时将其移动到链表头部，这样链表尾部的人必然是最后一次发言时间最早的人，这就是经典的 LRU 链表算法。

同时对每个人记录一个最近 T 时长的时间段内的发言的窗口，在时间推进过程中维护这个窗口，一旦窗口大小超过了 K 就将此人从链表中删除，同时将窗口置空。时间复杂度 $O(M)$。

事实上，用其他数据结构代替链表，只要写的不要太慢，以 $O(M \log M)$ 的时间复杂度同样可以通过本题。

本题的实际过题情况不及预期。

G. Expected Sum

每一位对答案的贡献取决于右侧第一个加号的位置，分别考虑每一位对答案的期望贡献。记第 i 位数为 a_i，对答案的期望贡献为 E_i。

$$
\begin{aligned}
E_n &= a_n \\
E_{n-1} &= a_{n-1}\left(p_{n-1} + 10(1-p_{n-1})\right) \\
E_{n-2} &= a_{n-2}\left(p_{n-2} + 10(1-p_{n-2})p_{n-1} + 100(1-p_{n-2})(1-p_{n-1})\right) \\
E_{n-3} &= a_{n-3}(p_{n-3} + 10(1-p_{n-3})p_{n-2} + 100(1-p_{n-3})(1-p_{n-2})p_{n-1} + \\
&\quad 1000(1-p_{n-3})(1-p_{n-2})(1-p_{n-1})) \\
&\cdots
\end{aligned}
$$

容易发现上面的式子是可以递推的，于是可以在 $O(n)$ 的时间内求出答案。

H. Light the Street

显然对于两盏路灯之间，容易证明中点位置的亮度最小，证明如下：

对于两盏距离为 l 的路灯，离左边路灯 x 距离的位置亮度为

$$\frac{d}{x^2}+\frac{d}{(l-x)^2}=d\left(\frac{1}{x^2}+\frac{1}{(l-x)^2}\right)$$

对 x 求导得 $d\left(\frac{-2}{x^3}+\frac{2}{(l-x)^3}\right)$，令导数为 0 解得驻点 $x=\frac{l}{2}$，容易证明该驻点为极小值点。□

H. Light the Street

那么最暗的地方一定是两端以及相邻路灯中间，设相邻路灯距离为 l，那么两侧端点离路灯的距离就是 $\frac{\sqrt{2}}{2}l$，铺设总长为 $(k-1)l+\sqrt{2}l$。

要铺满的前提下 l 尽可能小，那就是 $l=\frac{n}{k-1+\sqrt{2}}$。然后算出来最暗处的亮度就好了。

H. Light the Street

或者可以二分一个 M，容易知道一盏路灯离边界的最大距离为 $\sqrt{\frac{d}{M}}$，两盏路灯间的最大距离为 $\sqrt{\frac{d}{2M}}$，可以 $O(1)$ 得出能铺出的最长距离，比较该距离与 n 的大小即可。时间复杂度 $O(T\log 10^9)$。

让 $k \le n$ 的意义在于让答案不会太大，避免精度问题。

当 $k = n$ 时差不多隔 1 距离放一个路灯，答案差不多是 $\frac{d}{0.5^2} = 4d$，以 $5d$ 为二分上界够用了。

I. Subsetting and Summing

其实只要确定 $|y_{S_1}|, |y_{S_2}|, |y_{S_3}|$ 最后贡献时究竟是正数还是负数，就能够确定每个向量对答案的贡献是多少，而这三者的正负性直接枚举即可。

枚举出来之后，就知道了每个向量的对答案的贡献，把贡献为正的加起来更新答案即可。

I. Subsetting and Summing

其实只要确定 $|y_{s_1}|, |y_{s_2}|, |y_{s_3}|$ 最后贡献时究竟是正数还是负数，就能够确定每个向量对答案的贡献是多少，而这三者的正负性直接枚举即可。

枚举出来之后，就知道了每个向量的对答案的贡献，把贡献为正的加起来更新答案即可。

但可能有这种情况，把贡献为正的向量加起来之后，不一定能保证 $y_{s_1}, y_{s_2}, y_{s_3}$ 的正负性和我们枚举的一样，不过这种情况其实不影响，因为这种情况下的答案肯定没有正负性和我们枚举的一样时答案更优。

时间复杂度 $O(2^3 n)$。

J. Less Time on the Road

首先使用 bfs 或 floyd 算法求出两两最短距离。

设 $dp[i][j][k][l]$ 表示完成了前 i 个请求，这次没走的人的位置在 j，A 走过的路程为 k，这次走的是 A 或 B （用 l 为 0 或 1 表示）的情况下 B 的最短路程。

由于这次走的人必然位于 x_i，故上面的定义足以完整表示状态。转移时枚举上一次谁走以及这一次谁走即可。

A 走过的总路程可能达到 nm 级别，故时间复杂度为 $O(n^2m^2)$。为了避免内存超限，需要将 dp 数组第一维做滚动处理。

K. The Secret Comparison

究极签到题，照做即可。

输出的时候建议复制题目所给，里面的 Eleven 是一个小坑。

L. Spatial Quantum Energy Theory

首先，我们用人话翻译一下原题：

原子由粒子组成，有 20 种粒子，原子能量的计算公式为

$$\left(\sum_{i=1}^{m} e_{a_i}\right)^m$$

我们假设 a_i 为第 i 个原子的粒子组成，E_i 为第 i 个原子的能量。

L. Spatial Quantum Energy Theory

在此基础上，求 “感应激发”(inductive excitation) 能量

$$\sum_{\substack{1\le i<j\le n \\ a_i|a_j=\max(a_i,a_j)}} E_iE_j$$

以及 “震荡激发”(oscillatory excitation) 能量

$$\sum_{\substack{1\le i<j\le n \\ a_i|a_j=\max(a_i,a_j)}} \left(\sum_{\substack{k\neq i,k\neq j \\ \max(a_i,a_j)|a_k=\max(a_i,a_j) \\ a_k|\min(a_i,a_j)=a_k}} E_iE_j\,\mathrm{popcount}(k) \right)$$

L. Spatial Quantum Energy Theory

先求震荡激发的能量，我们考虑交换求和顺序，即固定受激发原子，求有哪些感应激发的原子对使得该原子收到激发，以及其激发能量。

我们假设现在有三个原子 i, j, k, $(a_i \geq a_j, k \neq i, k \neq j)$，可以发现 i 和 j 能震荡激发 k 原子，当且仅当 $a_i | a_k = a_i, a_k | a_j = a_k$。所以将 $\mathrm{popcount}(k)$ 提出来后，我们只要求出所有粒子集合包含 a_k 以及所有包含于 a_k 的能量和，再将它们直接相乘（实际上还有一些细节需要处理）即可。

L. Spatial Quantum Energy Theory

于是现在问题转化为求

$$\sum_{a_j | a_i = a_i} E_j$$

以及

$$\sum_{a_j \& a_i = a_i} E_j$$

显然可以直接用 FWT（快速沃尔什变换）求解。

稍加处理，也可以由 FWT 的结果直接求出感应激发释放的能量。

M. Easy problem of prime

关于分解为质数的和，也许你会想到哥德巴赫猜想，其实这题就是用这个来做。

由于哥德巴赫猜想在题目所给的数据范围内正确，所以不妨当成定理使用。众所周知其欧拉版本：对于任意大于 2 的偶数，都一定能分解成两个质数之和。

于是 $f(n)$ 的 n 为偶数的情况就 $O(1)$ 解决了，奇数的情况也很简单。

1. 如果该奇数本身就是质数，那么显然 $f(n) = 1$；
2. 如果 $n - 2$ 是质数，那么 $f(n) = 2$。
3. 否则 $n - 3$ 一定是偶数，有 $f(n) = 3$。

加上一个 $O(n)$ 的欧拉筛，这题就解决了。

Thank you