Group D

赛前FLAG

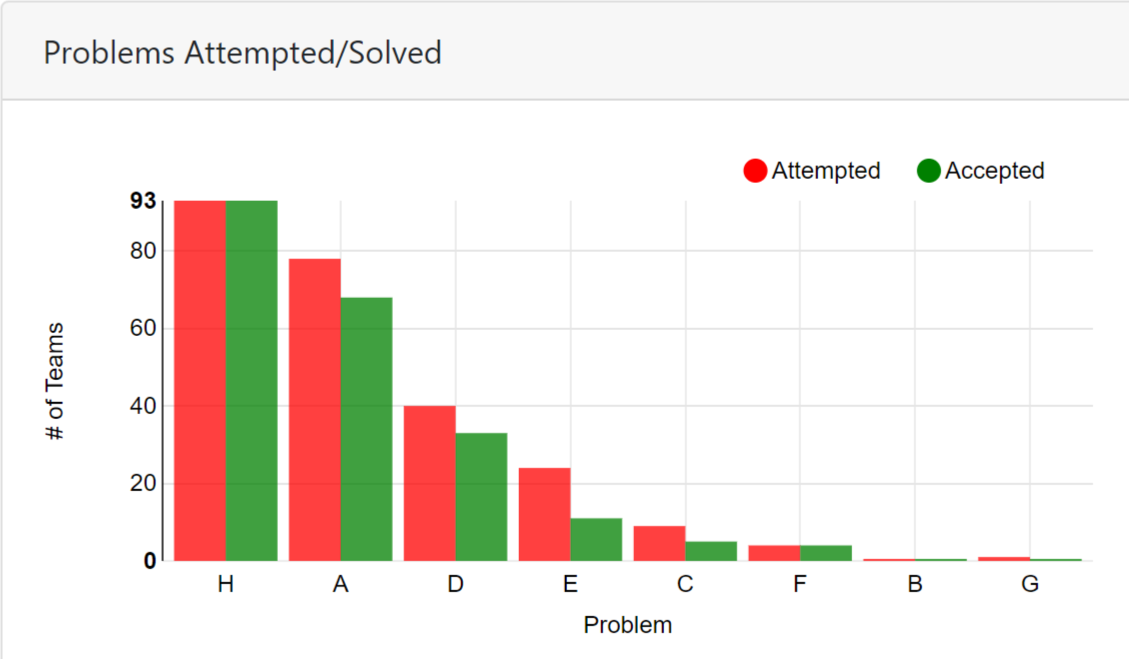
- 题面不出锅
- 数据不出锅
- 10分钟之内通过第一个题
- 有大于等于5个题被开出来
- 第一名题数大于等于4个题
- 春季集训队成员没有人爆零
- Sky_miner真的能BG我们一次

赛前FLAG

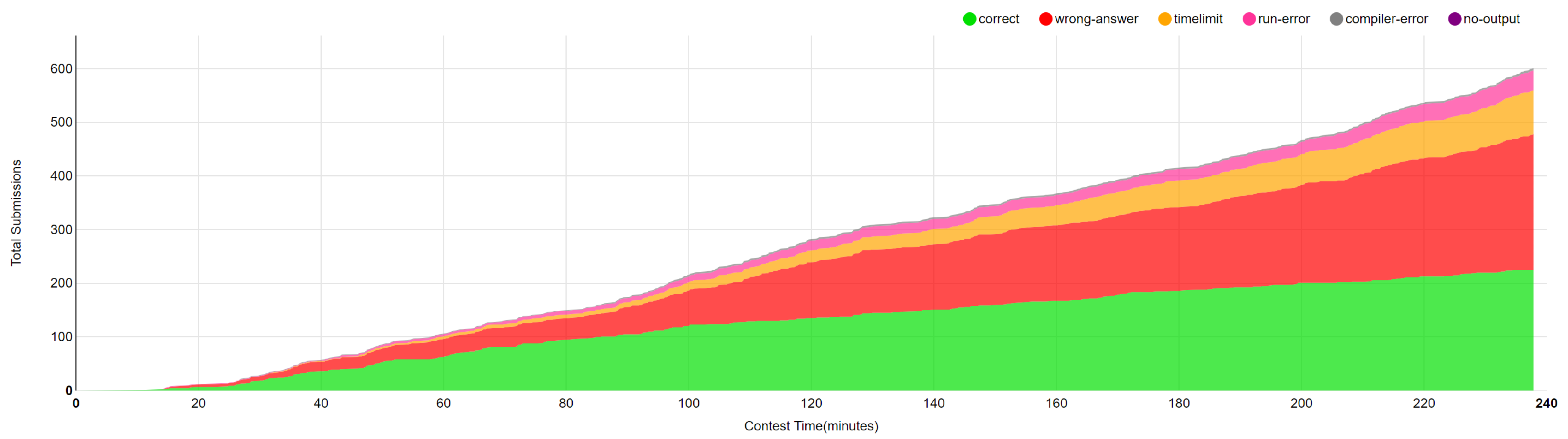
- 题面不出锅
- 数据不出锅
- 10分钟之内通过第一个题
- 有大于等于5个题被开出来
- 第一名题数大于等于4个题
- 春季集训队成员没有人爆零
- Sky_miner真的能BG我们一次

题目难度

- 预测
  - H < A < C=D=E < B=F=G
- 实际



Submissions Over Time



Last blood

- A - 顾哲涵
- C – 陈威志
- D – 李昌栋
- E – 杨沛霖
- F - 张　鑫
- H – 周屹赫

Sky_miner and Odious Number

容易发现，当前面的数位一定时，xxx0与xxx1中有且仅有一个是odious number。故f(i) ~ 2i。通过建立映射关系，可知f(i)的取值仅为2i-2与2i-1中的一者。于是只要检验2i-2是否合法即可。

本题还有其他做法，如搜索等，也可利用其他公式算出。

Sky_miner and Course Selection

- 首先假设M门课都开，这时候学生人数最多的课为P，并有Q位学生。那么最终答案至多是Q.
- 考虑答案<Q的情况:
- 如果P课开设,不论剩下的M-1课开不开设,不会影响已经选上P课的同学,因此答案不会比Q小.
- 所以我们确定P课不选,再对剩余的M-1门课重复相同的的操作,取每次操作得到的选课人数最多的课的人数的最小值即可.

Sky_miner and Binary Trees

- 考虑到颜色具有传递性
- 我们把颜色间的传递关系连边，建成一张有向图。
- 通过观察我们可以发现，某种颜色的根节点，深度为x的儿子个数，恰好为这个有向图上从这个颜色结点出发，长度为x的路径条数。
- 这就变成了一个经典问题
- 用A表示最初的有向图，A^k表示深度为k的所有颜色数
- 问题就变成了求$\sum A^k$

E.Sky_miner and Count Connecting Block

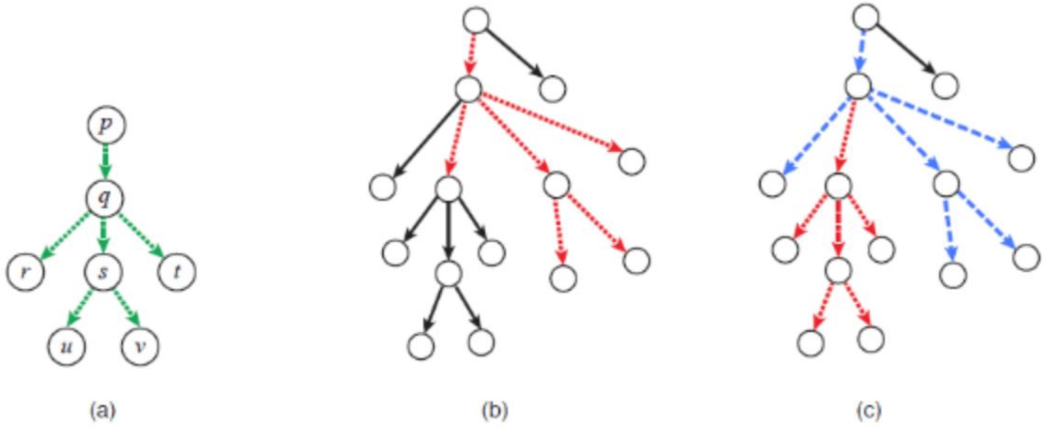
题意简述：给出一个黑白棋盘，求黑色联通块个数。

题解：一行一行处理，使用并查集。处理某行时，标记上一行的区域合并到了哪块，在处理一个区域时，把上一行所有与该区域有交的区域合并。

C.Problem Sky_miner and Stick-Man

简单题

题意：将一棵树划分出最多的“火柴人”



C.Problem Sky_miner and Stick-Man

简单题

贪心（树形DP）

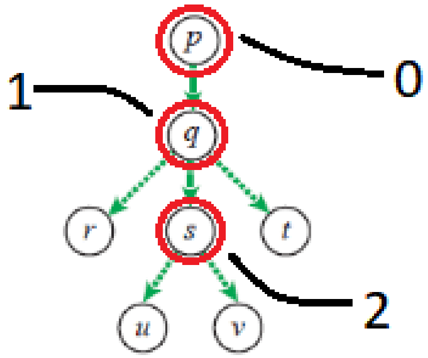
贪心：贪心的在树上由底自根考虑每一个“火柴人”是否能放上去。

C.Problem Sky_miner and Stick-Man

简单题

贪心（树形DP）

树形DP：将“火柴人”划分为3个状态进行考虑即可



Sky_miner and Continued Fraction

- 维护一个长为 n 的序列，q 次操作：单点修改，或询问一段区间组成的连分数的值（对 998244353 取模）。

- 1 <= n, q <= 10^5
- 2 <= a < 998244353

Sky_miner and Continued Fraction

- 考虑连分数 $[a_1; a_2, \dots a_n]$，记 δ_i 为取前 i 项的分数，考虑其分子分母，算一下前三项，可以发现一些东西：

$$\delta_1 = \frac{P_1}{Q_1} = a_1 = \frac{a_1}{1}$$
$$\delta_2 = \frac{P_2}{Q_2} = a_1 + \frac{1}{a_2} = \frac{a_2 a_1 + 1}{a_2} = \frac{a_2 P_1 + 1}{a_2 Q_1}$$
$$\delta_3 = \frac{P_3}{Q_3} = a_1 + \frac{1}{a_2 + \frac{1}{a_3}} = a_1 + \frac{a_3}{a_3 a_2 + 1} = \frac{a_3 a_2 a_1 + a_3 + a_1}{a_3 a_2 + 1} = \frac{a_3 P_2 + P_1}{a_2 Q_2 + Q_1}$$

Sky_miner and Continued Fraction

- 事实上有:

$$
\begin{aligned}
\delta_k &= \frac{P_k}{Q_k} = [a_1; a_2, \cdots, a_{k-1}, a_k] = [a_1; a_2, \cdots, a_{k-1} + \frac{1}{a_k}] = [a_1; a_2, \cdots, \frac{a_k a_{k-1} + 1}{a_k}] \\
&= \frac{\frac{a_k a_{k-1}+1}{a_k} P_{k-2} + P_{k-3}}{\frac{a_k a_{k-1}+1}{a_k} Q_{k-2} + Q_{k-3}} = \frac{(a_k a_{k-1} + 1) P_{k-2} + a_k P_{k-3}}{(a_k a_{k-1} + 1) Q_{k-2} + a_k Q_{k-3}} = \frac{a_k (a_{k-1} P_{k-2} + P_{k-3}) + P_{k-2}}{a_k (a_{k-1} Q_{k-2} + Q_{k-3}) + Q_{k-2}} \\
&= \frac{a_k P_{k-1} + P_{k-2}}{a_k Q_{k-1} + Q_{k-2}}
\end{aligned}
$$

$$\therefore\ P_k = a_k P_{k-1} + P_{k-2},\ Q_k = a_k Q_{k-1} + Q_{k-2}$$

- 用线段树维护二维的递推矩阵即可

B-暴力

- 考虑暴力dijkstra
- 枚举每个未使用的点，找到全局最小值，拿这个点更新其余所有点。注意存边需要一组一组存/使用邻接矩阵，否则会MLE，复杂度O(n^2)

B-正解

- 观察到暴力做法需要支持的操作只有以下三种:
- 1.区间对等差数列取min
- 2.查询全局最小值及其位置
- 3.删除一个点

B-正解

- 于是可以用线段树进行维护，每个节点上存一个一次函数，修改时若新一次函数在整个区间内都比当前节点上存的一次函数小则直接替换，若新一次函数在整个区间内都比当前节点上存的一次函数大则return， 否则算出两个一次函数的交点，用只对当前节点所代表的区间少于一半位置有作用的那条一次函数向下递归。

B-正解

- 可以证明每次向下传的区间至少会减短一半，所以每次将一个一次函数加到一个节点上耗时O(logn)，而区间对等差数列取min至多将一次函数加到O(logn)个节点上，所以单次操作总复杂度O(log^2 n)
- 简称李超树
- 删除节点操作只需额外维护每个节点上最左和最右的未被删除位置即可
- 总复杂度O(nlog^2 n)

G

这是个二维博弈，需要用到Nim积与Nim和

计算初始情况，$A = \bigoplus_{i=1}^{n} x_i \otimes y_i$

列方程，$X \otimes X \oplus A = 0$，解出X即为答案

$$M = 2^{2^k} \leq B < 2^{2^{k+1}}$$

$$B = a * M + b$$

$$B \otimes B = a \otimes a \otimes M \oplus a \otimes a \otimes (M/2) \oplus b \otimes b$$

先计算a再同样方法计算b