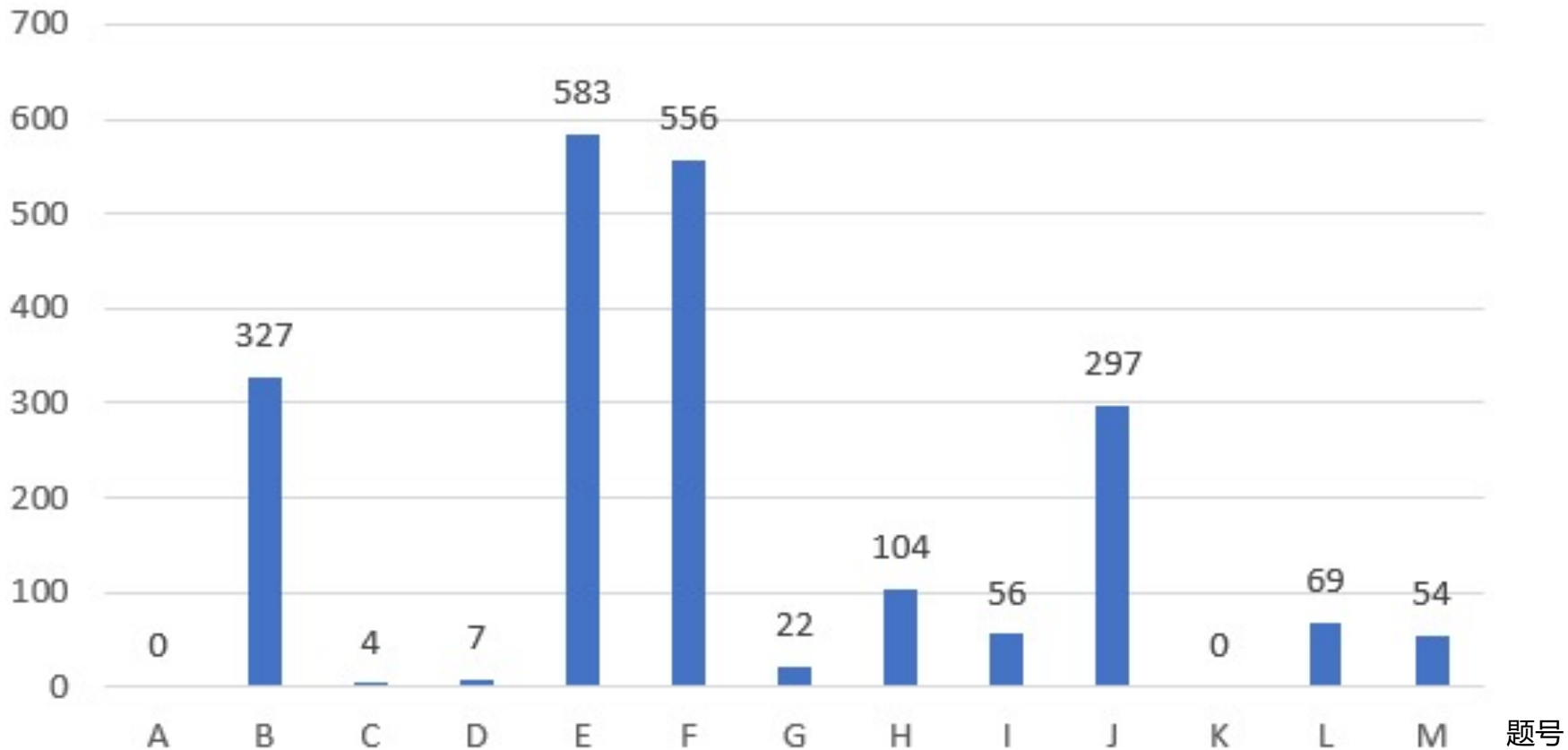
ICPC2021沈阳站题解

-------沈阳站命题组

通过人数


700
600
500
400
300
200
100
0
0
327
4
7
583
556
22
104
56
297
0
69
54
A
B
C
D
E
F
G
H
I
J
K
L
M
题号

A A Bite of Teyvet

- 考虑维护每个圆能提供贡献的区间
- 也就是对每个x计算$\underset{1\le i\le n}{\operatorname{argmax}}(r^2-(x-x_i)^2)$
- 此时x可以看做常数，也就是计算$\underset{1\le i\le n}{\operatorname{argmax}}(2x_ix+(r^2-x_i^2))$
- 使用动态半平面交维护直线$2x_ix+(r^2-x_i^2)$的上包络
- 在每条直线作为上包络的一部分出现时，用直线对应圆在对应x区间内的面积更新答案
- 复杂度$O(nlogn)$

B Bitwise Exclusive-OR Sequence

- 考虑逐位确定序列，则若干个限制可以看成一条(u,v,w)的边，其中w为0或1
- 序列合法等价于存在一种集合划分方案，其中w为1的边跨越集合，而w为0的边不跨越集合，这与二分图定义等价
- 因此只需要按给出的限制二分图染色即可
- 此种方法需要30次dfs，常数比较大，需要注意常数

- 注意到二分图染色的过程并不需要逐位考虑，可以钦定一个点为0然后按照限制确定其余位置，由刚才的分析可知这样是等价的。此种方法只需要1次dfs，能轻松通过此题

C Cards of Magic

- 复制卡可以复制复制卡，n=5的样例需要使用这个策略来在手里有一张水人卡和两张复制卡时打出5的伤害而不是4的伤害
- 最优策略是固定的
  - 在没有水人的情况下
    - 摸到复制卡和摸到火球卡都先存着
    - 除非手里有火球卡且复制卡复制火球卡之后和现有火球卡一起打出去可以打败怪物
  - 在有水人的情况下
    - 水人卡可以直接打出造成1伤害，火球卡可以直接打出造成3伤害
    - 对于第一张复制卡，如果没有打出过火球卡就先存着，除非怪物血量≤2，则可以复制水人卡直接打出2伤害打败怪物，如果已经打出过火球卡，则可以复制火球卡并打出4伤害
    - 如果再摸到第二张复制卡，就可以两张复制卡相互复制打出无限伤害

C Cards of Magic (Cont'd)

- 考虑将过程分为两阶段，摸到水人卡之前和摸到水人卡之后，然后枚举哪一回合摸到第一张水人卡
- 对于摸到水人卡之前，f[k][0/1][0/1/2]表示摸了k张牌，其中有0/≥1张火球卡，有0/1/≥2张复制卡的概率
- 对于摸到水人卡之后，g[n][0/1][0/1/2]表示怪物血量是n，已经用过0/≥1张火球卡，没有摸到复制卡/手里有一张没打出的复制卡/已经打出过复制卡时，打败怪物的期望回合数
- 那么就可以枚举第一次摸到水人卡的回合，使用上述两组信息进行贡献的计算，需要注意回合数较大只能是一直摸复制卡，在摸了很多张复制卡之后，任意摸到一张非复制卡就能打败怪物

- 另解，利用$E(X) = \sum_{k>0} P(X \geq k)$计算摸k-1张牌还不能打败怪物的概率，当k较大时只有一直复制牌无法打败怪物，否则枚举k进行计算，可能需要维护一些组合数的后缀和，此处略去细节

D Cross the Maze

- 假设答案已知，则原问题变为判定性问题
- 对于该问题，考虑用网络流解决。对于每个位置(x,y)拆分成T个点以满足按时间序进行的条件；同时再拆成两个点以满足同时刻只能有一个人的条件。在新图上判断是否满流，即可判定T是否为解
- 回到原问题，寻求最少时间，只需要二分答案或者枚举答案并依次添加新边即可
- 对于方案的输出，只需要每个起点分别dfs寻找一条满流的合法路径即可。可以证明，此种方法一定能找到一个合法方案

E Edward Gaming, the Champion

- 签到题

F Encoded Strings I

- 根据题目描述把所有前缀编码得到n个字符串
- 对这n个字符串排序输出最大字典序的即可

G Encoded Strings II

- 答案串一定是非递增的，并且所有前 出现的字符种类数 个小写字母都有至少一个
- 在这个前提下，答案串字典序最大等价于从前往后每次选择一种字符，并且每步都要选到最多的字符
- 记f[S]表示从后往前选S集合里的字符，至少要多长的后缀才能保证S里的字符都能选出来
- 这样从前往后贪心的时候，就需要从除去f[S]长度的后缀的部分里选S集合外的字符
- 于是dp[S]表示从前往后贪心选了S集合里的字符，最后一个字符位置的最小值
- 按照|S|从小到大转移，相当于从前往后确定每种字符的出现次数，每一种都需要选到当前最大的次数，选不到最大的转移丢弃即可
- 复杂度$O(|\Sigma|2^{|\Sigma|})$

H Line Graph Matching

- 线图的最大权匹配就是原图的最大权边匹配
  - 边匹配：两条边有公共端点则可以匹配
- 保证原图是连通图，只有一个连通块
- 当边数是偶数时，所有边都可以被匹配完
  - 原图用DFS展开成一棵树，在树上自底向上贪心匹配即可
- 当边数是奇数时，恰好有一条边不会被匹配
  - 如果是非割边，这条边可以不被匹配
  - 如果是割边，且割去这条边之后两侧边数均为偶数，则这条边也可以不被匹配

I Linear Fractional Transformation

- 做法一：分两种情况解方程
- c=0
  - 此时f(z)=(az+b)/d是线性变换，不妨设d=1，只有两个变量
  - 选两个点解出a,b之后代入第三个点检查是否满足即可
- c≠0
  - 由齐次性不妨设c=1，此时有三个变量，解出a,b,d即可

- 做法二：分式线性变换保交比
  - $\frac{w_0-w_1}{w_0-w_2}\Big/\frac{w_3-w_1}{w_3-w_2}=\frac{z_0-z_1}{z_0-z_2}\Big/\frac{z_3-z_1}{z_3-z_2}$，其中$w_0=f(z_0)$

J Luggage Lock

- 从状态 abcd -> efgh 可以规约成从 0000 开始变换
- 只需要从 0000 这个状态 BFS 一下所有 10^4 种方案

K Matrix Operations

- 本题等价于矩形加，查矩形最大值，允许离线
- 离线处理后，可将长度 m 的操作序列按块大小 B 分块
- 对每个块，首先按块内的矩形将平面划分为 $B \times B$ 的网格，然后用算法 1 求出在这个块之前的操作完成后，每个格子的最大值，将块内的问题转化为 $B \times B$ 网格上的 B 矩形加，查矩形最大值，使用算法 2 求解

K Matrix Operations (Cont'd)

- 算法1
- 问题转化为 $O(m)$ 次矩形加，然后查 B^2 次矩形最大值，查询构成 $B \times B$ 网格

- 使用扫描线扫平面的一维，线段树支持 $O(m)$ 次区间加，维护另一维的前缀历史最大值，每扫完一格进行 B 次查询并清空历史最值。线段树建树时使得每个查询对应于一个节点，则批量 B 次查询只需 $O(B)$ 时间，而单次修改是正常的 $O(logm)$。于是可以在 $O(mlogm + B^2)$ 时间内完成

K Matrix Operations (Cont'd)

- 算法2
- 使用 k-d 树或者其它离线分治方法可以做到 $O(B^2)$ 时间
- 总时间 $T(m) = O(\frac{m}{B}(mlogm + B^2)) = O(m\sqrt{mlogm})$

- Timothy Chan 关于 Klee's measure 的论文里提到的 Klee's measure on depth 和这个问题差不多，他使用的做法应该和我们实现的也差不多

L Perfect Matchings

- 考虑有n个条件，每个条件为第i条边不能选
- 考虑k条边一定选，考虑容斥
- 求任何k个条件 即k条边一定选其他边任选的方案数
- 这些条件构成了一棵树
- 考虑树形dp，dp[i][j][0/1]表示 以i为根的子树，有j个匹配根节点i，选或不选的方案数，转移就是背包转移

M String Problem

- 考虑在线处理每加入一个字符之后的最大子串，显然最大的子串一定是一个后缀
- 考虑新加入一个字符c，一定是上一步最大子串的路径退回到某个节点之后走出一步c，这个节点显然是路径上最早的能够使字典序变大的节点，也就是所有当前出边比c小的节点中，离起点最近的一个点
- 所以在动态构建sam的过程中，维护最大子串路径和路径上的点的深度和出边即可。由于每次最多往外走一条边，所以往回走的总边数也不会超过n

M String Problem (Alternate)

- 字典序最大的子串一定是某个后缀，记长为r的前缀里字典序最大的后缀为ans[r]
- ans[r+1]一定是ans[r]的某个border结尾添加第r+1个字符
  - 一个串的border是指既是某个前缀又是某个后缀的串
- 找出最短的border，在这个border结尾加上第r+1个字符之后比ans[r]大，那么这个border结尾添加第r+1个字符之后就是长为r+1的前缀的答案
- 长为n的字符串里所有border的长度可以划分为O(logn)个的等差数列，这些等差数列里只有首项和末项是有用的
- 使用kmp算法快速枚举这O(logn)个有用的border即可