2020 ICPC 小米邀请赛网络赛第一场
试题分析

某不愿透露姓名的出题人

比赛链接：https://ac.nowcoder.com/acm/contest/7501

A. Intelligent Warehouse

设 $Dp[i]$ 表示当前选的所有数都是 i 的约数且符合题意的情况下能选的数的个数的最大值。最后答案就是所有 Dp 值中的最大值。

一个非常直观的转移就是用 $Dp[i]$ 去更新 i 的所有倍数的 Dp 值，但是这样复杂度是 $O(W \log W)$ 的，可能会 TLE~~（但实际上跑得和正解一样快）~~。实际上只用枚举 i 的素数倍就可以了，因为合数总可以被若干个素数的乘积给凑出来，就没必要再枚举了，复杂度和埃拉托斯特尼筛法的复杂度是一样的。

时间复杂度：$O(W \log\log W)$

B. Intelligent Robot

首先有用的点只有 $O(k)$ 个，然后问题在于判断两点之间是否可以直达，这个就 $O(k)$ 地枚举每堵墙看有没有挡住线路，具体来说就看两点组成的线段与墙对应的线段师是否严格相交。最后再用 Dijkstra 算法求最短路即可。

时间复杂度：$O(k^3)$

C. Smart Browser

按照题意模拟。

D. Router Mesh

这题乍一眼看上去像是个时间分治 + 可撤销并查集（按秩合并）的数据结构题，但复杂度是 $O(m + n\log^2 n)$ 的，可能会 TLE。

实际上这题有图论做法，只需要对每个点 i 求出其所在的点双连通分量的个数 c_i，当然也要求出初始的连通块个数 C，那么 i 的答案就是 $C + c_i - 1$。

时间复杂度：$O(n + m)$

E. Phone Network

先考虑维护一个 $R_{i,l}$，表示以 l 为左端点，包含 $1 \sim i$ 中所有数字的最小右端点。那么当 i 转移到 $i+1$ 时，不妨设数字 $i+1$ 的位置从左到右分别为 $p_1, p_2, \cdots, p_k$，那么可知对于 $[p_{j-1}+1, p_j]$ 中的 l，其 $R_{i+1,l} = \max\{R_{i,l}, p_j\}$，就变成区间求 max 了。特别地，对于 $[p_k+1, n]$ 的这一段 $R_{i+1,l}$，可以令其变成 $+\infty$。

注意到 $R_{i,1\sim n}$ 是单调的，所以我们可以在 $[p_{j-1}+1, p_j]$ 中找到 $R_{i+1,l} < p_j$ 的一段 l，这样就变成区间赋值了。然后考虑再维护一个 $R_{i,l} - l + 1$ 的值，每次取其中的最小值即为答案。

时间复杂度：$O(n \log n)$

F. Design Problemset

这题可能有个坑点，就是所有题的 l_i, r_i 的限制与 L, R 的限制冲突，即 $\sum r_i < L$ 或者 $R < \sum l_i$，这样的话答案自然是 0。

首先一套题的题目个数肯定是越少越好，所以就取 $\max\{L, \sum l_i\}$ 作为一套题的题目数量 P。然后二分答案，不妨设当前答案为 A，考虑检验其合法性。首先看所有种类的题目是否满足 $a_i \geq A \times l_i$，如果不满足则肯定就不合法，否则每套题还需要 $P - \sum l_i$ 道题来充数，总共就需要 $A \times (P - \sum l_i)$ 道充数题，然后每种题目可以拿出 $\min\{a_i - A \times l_i, A \times (r_i - l_i)\}$ 道题来充数，看已有的可充数题是否不少于所需充数题即可。

时间复杂度：$O(n \log W)$

G. Tree Projection

做法很多，标程的做法如下（记 $PosA[i]$ 为 i 在 A 中的出现位置）：

```
puts("YES");
for (int i = 2, cur = B[1]; i <= n; i ++) {
    printf("%d %d\n", cur, B[i]);
    if (PosA[B[i]] < PosA[cur])
        cur = B[i];
}
```

大概是说从左到右枚举排列 B，对于一个 $B[i]\ (i > 1)$，找到 $B[1 \sim i-1]$ 中在排列 A 中出现位置最靠前的一个并与其连边。

可以验证上述算法运行结果就是一个合法解。

- 考虑排列 A：对于每一条边 $(cur, B[i])$，可知谁在排列 A 中更靠前，谁就离 $A[1]$ 更近，也就是说在 A 中靠前的点是靠后的点的父亲。所以当 $A[1]$ 为根时，A 是 T 的一个合法拓扑序。

- 考虑排列 B：当 $B[1]$ 为根时，对于一个点 $B[i]$，可知要么这个点是叶子节点，要么后面所有的点 $B[i+1 \sim n]$ 都是这个点的子孙，也就是说每个点的子树都在排列 B 的某个子区间中。所以当 $B[1]$ 为根时，B 是 T 的一个合法 dfs 序。

H. Grouping

一个分组方案权值为 $\frac{1}{n}\sum\limits_{g\in m} w_g^2 - (\frac{1}{n}\sum\limits_{g\in m} w_g)^2$，总答案为 $\frac{1}{|M|}\sum\limits_{m\in M}(\frac{1}{n}\sum\limits_{g\in m} w_g^2 - (\frac{1}{n}\sum\limits_{g\in m} w_g)^2)$。其中 M, m, g, w_g 分别代表所有的分组方案，某个分组方案，分组方案 m 中的某个组，以及 g 这个组的权值。下同。

首先考虑 $\frac{1}{|M|}\sum\limits_{m\in M}\frac{1}{n}\sum\limits_{g\in m} w_g^2 = \frac{1}{|M|n}\sum\limits_{m\in M}\sum\limits_{g\in m} w_g^2$。因为所有组的出现次数相同，所以上式可以化成 $\frac{1}{|G|}\sum_{g\in G} w_g^2$，记作 (1) 式，其中 G 代表所有可能的组的集合。把所有数从小到大排序然后线性扫一遍，维护一下前缀和以及前缀平方和就可以算出 $\sum_{g\in G} w_g^2$，再取个平均即可。

然后考虑求 $\frac{1}{|M|}\sum\limits_{m\in M}(\frac{1}{n}\sum\limits_{g\in m} w_g)^2$，记作 (2) 式。首先可以求出把 $2n$ 个数分成 n 组的方案数 $T_n = (2n-1)!!$，这个可以用归纳法来证明，然后式子就可以化成 $\frac{1}{T_n\times n^2}\sum_{m\in M}(\sum_{g\in m} w_g)^2$。考虑每两组数 $(i,j),(k,l)$ $(1\le i,j,k,l\le 2n)$ 对 $\sum_{m\in M}(\sum_{g\in m} w_g)^2$ 的贡献：

- 如果 $(i,j) = (k,l)$，则其会在 T_{n-1} 种分配方法中产生贡献，总贡献为 $T_{n-1}\times(a_i - a_j)^2$
- 否则如果 $i = k$ or $i = l$ or $j = k$ or $j = l$，则其不可能带来贡献，因为一个数字在一种分配方法中出现一次，同一个方案的不同两组数字是不可能有重复标号的数字的
- 否则会在 T_{n-2} 种分配方法中产生贡献，总贡献为 $2\times T_{n-2}\times|a_i - a_j|\times|a_k - a_l|$

所以就把上面这些贡献累加，然后除以 $T_n\times n^2$ 即可得到 (2) 式。其中 $(i,j) = (k,l)$ 部分的实际上就是 T_{n-1} 乘以 (1) 式。考虑第三个部分的贡献，首先可以先求出 $(\sum_{g\in G} w_g)^2$，即全集，然后枚举重复的数字 i，并减去 $(\sum_{j\neq i}|a_i - a_j|)^2$，还要再加回一个 $\sum_{g\in G} w_g^2$，因为形如 $(i,j),(i,j)$ 的会被减两次（i 那里一次，j 那里一次），加加减减完之后再乘上一个 T_{n-2} 即得到第三部分的贡献。至于数字大小 10^6 的限制，只是拿来迷惑一下大家，并无特殊作用。

时间复杂度：$O(n\log n)$

I. Walking Machine

把图倒过来建，然后从迷宫外开始 BFS，看能搜到多少个格子即可。

时间复杂度：$O(nm)$

J. Matrix Subtraction

考虑 $M_{1,1}$ 只能被 $(1,1)-(a,b)$ 的子矩阵处理，所以 $(1,1)-(a,b)$ 的选择次数是确定的，同理可以求出 $(1,2)-(a,b+1)$ 以及其他所有 $a\times b$ 的子矩阵的选择次数。

记 $C_{i,j}$ 为 $(i,j)-(i+a-1,j+b-1)$ 的选择次数，可知有：

$$C_{i,j} = M_{i,j} - \sum_{u=0}^{a-1}\sum_{v=0}^{b-1}[u\neq 0 \text{ or } v\neq 0]C_{i-u,j-v}$$

这个用二位前缀和 + 二维差分就可以在 $O(1)$ 的时间内算出来。最后看是否所有的 $C_{i,j}$ 都为非负以及 M 是否恰好会变成全 0 即可。

时间复杂度：$O(\sum nm)$

K. Sqrt Approaching

这个题的本质就是：给定一个分数 $X\,(X=\frac{A}{B})$，要求找到另一个分数 $Y\,(Y=\frac{C}{D})$，使得 Y 在 X 与 $\sqrt{n}$ 之间。

一个直观的思路是把 A,B 扩大若干倍，比如就往后面加零直到长度为 99999，得到 C,D，然后再根据 A^2 和 nB^2 的大小关系来给 C,D 进行微调，但这样是会 WA 的，如果输入的 A,B,n 满足 $|A^2-nB^2|=1$，那么有 $|\frac{A}{B}-\sqrt{n}|=\frac{1}{B(A+\sqrt{n}B)}$，这样的话误差就是 $10^{-199980}$ 的级别，然而一般的微调都会产生 $10^{-100000}$ 的浮动，就算套上二分也很难达到 $10^{-199980}$ 的精度。

于是我们想要一个通解，标程的做法是：

$$C=(n+1)A+2nB,\ D=2A+(n+1)B$$

可以验证满足题目要求的条件。易证从略。

ps. 这个题是怎么出出来的？首先本人从某个地方看到了一个题：

> 是否存在一组正整数 n,m，使得 $(3+5\sqrt{2})^n=(5+3\sqrt{2})^m$ ？

这题本人一眼看上去就是不存在，但想了很久不会证，就寻思写个程序打表找规律，这里不妨把 $(a+b\sqrt{2})^n$ 表示成 $A_n+B_n\sqrt{2}$，然后发现 $\frac{A_n}{B_n}$ 会越来越趋近于 $\sqrt{2}$，并且似乎对于所有的 a,b 都有这个性质，本人就尝试证明了一下，进而发现对于其他的根号也有类似的性质，然后就想到出这样一个根号数值逼近的题了。