第十五届“商汤杯”北京航空航天大学程序设计竞赛决赛题解

北航 ACM 集训队

A　极巨团体战

枚举喵头目的数量 x，答案为 $\max_{0\le x\le n}\{(100x+200(n-x))\cdot 1.1^x\}$。时间复杂度 $\mathcal{O}(n)$。

也可以求导，时间复杂度 $\mathcal{O}(1)$。

B　国士无双

观察“国士无双”要求的牌型，有两个要求，一个是 $1,2,\ldots,13$ 各出现一次，另一个是不允许出现其他的数字。而一共只有 14 个数字，也就是一定恰有一个 $1,2,\ldots,13$ 的数字会出现两次。这意味着一共只有 13 种合法的牌型。

那么可以选择枚举这个数字是多少，然后对比原始牌型与目标牌型之间的距离。

当然，也可以直接统计每个数字出现次数，然后查看有几个数字出现了 1 次，几个数字出现了 2 次及以上，几个数字出现了 0 次。若有出现了 2 次的，答案为出现了 0 次的总个数，否则为出现了 0 次的个数 +1。（因为需要把一个其他的变为 2 次）

时间复杂度 $\mathcal{O}(1)$，空间复杂度 $\mathcal{O}(1)$。

C　亦或骗子

这是一个骗子题，甚至不用读 a_i。

最大值当每个元素各自分为一段就可以取到，最小值全部分在一起就可以取到。

简单证明：分一起只会让分数和不变或变小，切开来只会让分数和不变或变大。

给一个 Python 的代码：

```python
n = int(input())
a = list(map(int, input().split())) # 可以不写
print(*range(1, n + 1))
print(*[1] * n)
```

时间复杂度 $\mathcal{O}(n)$。

D 树上路径

设小 A 在 $m-1$ 次移动中移动的距离分别为 $x_1, x_2, \cdots, x_{m-1}$，则 $x_1 + x_2 + \cdots + x_{m-1} = n - 1$，且 $x_1, x_2, \cdots, x_{m-1} \geq k$。所求答案为该不定方程解数的奇偶性。

不妨设 $x_i = y_i + k - 1$,则方程变为 $y_1 + y_2 + \cdots + y_{m-1} = n - 1 - (m-1)(k-1)$，$y_1, y_2, \cdots, y_{m-1} \geq 1$，是经典的隔板法问题，答案为 $\binom{n-2-(m-1)(k-1)}{m-2}$。

该组合数的规模达到了 10^9 级别，因此用定义直接计算将超时。注意到判断奇偶性，等价于求该组合数模 2 的余数，而 2 是一个质数，因此可以用 lucas 定理进行化简。

时间复杂度 $\mathcal{O}(T \log n)$。

E 群体狂乱

这一题是一个按照题目的要求直接模拟的题。

枚举攻击顺序，然后按照题目要求去检查剩余存活的，枚举攻击谁的情况，这样的总体时间复杂度是 $\mathcal{O}(n! \cdot (n-1)^n)$，因为 $n \leq 6$，这并不会超时。

注意枚举情况的时候需要带上其概率，因为每一个在分支死亡随从的分布不一定相同，最终的概率也可能不相等，所以不能用 $\frac{\text{存活次数}}{\text{总次数}}$ 的方式来得到答案。

F woafrnraetns 与正整数

容易想到，将所有数进行排序，检查所有相邻的数是否满足要求即可。如果相邻的数对之间都无解，那么其它数对之间更不可能有解（差距更大）。但是排序的时间复杂度为 $\mathcal{O}(n \log n)$，不能承受。

这里有一个结论：在本题的数据范围下，至多 $T = 120,896$ 个数中一定有解。

设一组输入无解，则有 $a_1 \ge 1$，$a_i \ge \frac{q}{p} a_{i-1} (2 \le i \le n)$。于是 $a_n \ge \left(\frac{q}{p}\right)^{n-1}$，而 $a_n \le 10^9$。因此 $n-1 \le \log_{\frac{q}{p}} 10^9$。在本题的数据范围下，$n-1$ 最大可取 $\log_{\frac{10000}{9999}} 10^9 \approx 207,222.297$。事实上由于 a_i 均为整数，因此严格的界为前一段所述的数字。

因此，只需将前 $\min(n, T)$ 个数排序，在其中寻找解即可。

时间复杂度为 $\mathcal{O}(n + T \log T)$。

G　林克与宝箱咒语

枚举 u, v 的 lca d，那么显然 u 到 d 路径上的串要包含给定串 s 的一段前缀；d 到 v 路径上的串要包含给定串 s 的一段后缀，前缀和后缀的长度和需要大于等于 k。

设 $dp_{u,i,0}$ 表示 u 子树的所有结点中，有多少个到 u 路径上的串至多与 s 的前缀匹配 i 位；同理 $dp_{u,i,1}$ 表示后缀至多匹配 i 位。

采用树上背包的方式更新 dp 并计算答案。假设当前 dp_u 已经计算了 u 的一些子树，考虑一棵新子树 v。现在需要先计算 u 和 v 产生的答案，随后将 v 的子树加入进来。首先用 $u \to v$ 这条边更新 dp_v，这很容易在 $\mathcal{O}(k)$ 内做到。答案则需要加上 $\sum_{i=0}^{k} \sum_{j=k-i}^{k} dp_{u,i,0} dp_{v,j,1}$。最后再将 dp_v 加到 dp_u 中。

可以证明这样的时间复杂度为 $\mathcal{O}(nk)$，参见 cf 1097G。

H　宝可梦与分支进化

设 dp_i 表示序列前 i 个元素中必选第 i 个的前提下的最长子序列。转移方程为 $dp_i = \max_{1 \le j < i, a_j \text{ ancestor } a_i} \{dp[j]\} + 1$，答案为 $\max_{1 \le i \le n} \{dp_i\}$。直接做的时间复杂度为 $\mathcal{O}(n^2)$ 或 $\mathcal{O}(n^2 \log n)$，不能通过。

使用一些数据结构维护 dp 值即可加速转移。例如，使用树链剖分及线段树即可在 $\mathcal{O}(n \log^2 n)$ 的时间复杂度内完成转移。其中，更新某个点的 dp 值所需时间为 $\mathcal{O}(\log n)$，而查询所有祖先时，由于需要跳 $\mathcal{O}(\log n)$ 段重链，因此所需时间为 $\mathcal{O}(\log^2 n)$。常数小的话可以通过。

如果将序列反过来，问题将变得更加简单：这要求后一项在前一项的子树中，可以在 dfs 序上使用线段树维护，时间复杂度为 $\mathcal{O}(n \log n)$。

在原问题上也可以做到 $\mathcal{O}(n\log n)$ 的复杂度。方法是每次计算得到 dp_i 后，用它更新子树中所有结点的答案，同样可以在 dfs 序上使用线段树维护。

I Poison AND^OR Affection

I.1 做法 1

有显然 dp：设前 i 个数分成 k 段的最大总影响力为

$$g(i,k)=\max_{j=1}^{i-1}\{g(j,k-1)+f(j+1,i)\}$$

对一个固定的 r，$f(l,r)$ 随着 l 减小而递减，且连续 and 和连续 or 只有不超过 $\mathcal{O}(\log\max\{a_i\})$ 种取值，因此 $f(l,r)$ 对固定的 r 只有 $\mathcal{O}(\log\max\{a_i\})$ 种取值。

上面的式子可以枚举所有 f 可能的转移位置，用数据结构维护一下每种 k 下的 g 区间最大值，时间复杂度 $\mathcal{O}(nk\log\max\{a_i\}\log n)$。

I.2 做法 2

注意到按位考虑时，$f(l,r)=1$ 当且仅当 $[l,r]$ 中这一位有 0 也有 1，因此在已经分好的某一段后添加一个新的数，并不会让结果变差。于是对于做法 1 中的分界点，选择分界点处的位置转移就是最优解。

时间复杂度 $\mathcal{O}(nk\log\max\{a_i\})$。

I.3 做法 3

按位考虑，尝试证明分成 k 段的价值 $g_i(n,k)$ 关于 k 是凸的。对于当前最左端的端点 l，找一个最小的 r，使得 $[l,r]$ 之间既有 0 也有 1，显然此时分段能够让异或出来的价值多 1。当 $[1,n]$ 已经全部划分完毕后，此时的价值达到最大。对此的简单证明是，考虑找出原序列的 01 交替最长子序列，然后把这东西套到区间上，必然是最大的（因为不可能存在一个更长的 01 划分方式，进而得到更长的价值了）。

在此基础上，每段区间都是一个 $00\ldots(1)$ 或 $11\ldots(0)$ 的样式，这时无论怎么进行重新划分，都必然会导致一个段落里 01 被划分开（也可能不会，但是肯定不会比之前更优），因此价值非增。当每个区间都只选一个元素时，价值为 0。

综上，按位考虑的 $g_i(n,k)$ 关于 k 是凸的。同时，将所有位都考虑进价值，价值是每一位的一个线性组合，求二阶导并不改变正负性，因此 $g(n,k)$ 关于 k 也是凸的。

利用带权二分，本题可以做到 $\mathcal{O}(n\log k\log\max\{a_i\})$。

J 括号序列

将左括号视为 $+1$，将右括号视为 -1，会形成一个新序列 $\{a_i\}$，记它的前缀和为 $\{s_i\}$。

一个合法的括号序列需要满足 $s_i \geq 0\ (\forall 1 \leq i \leq n)$，即不会有匹配不上的右括号；还需要满足 $s_n = 0$，即左括号和右括号数量相等，每个左括号都能有一个右括号匹配。

修改 l,r 区间的操作就是把 a_i 的那一段正负号翻一下，可以分解为两次将 a_i 从某一个位置 x 开始正负号翻一下的操作。对应到 s_i 上就是从某一位置 x 开始都算一下 $s_i \leftarrow -(s_i - s_{x-1}) + s_{x-1}$。

修改后的查询操作就是检查 s_i 总体的最小值，以及单点求一下 s_n 的值。

以上的这些操作可以使用支持区间加、乘修改，支持区间查询最大最小值的线段树来完成。

时间复杂度 $\mathcal{O}(n+m\log n)$。