题解

T1 tree

结合样例可得一个贪心做法：每次从最大点权的点开始，删除与其相连的所有边，这样才能使权值大的点对答案的贡献尽可能少，尽可能早地将最大点权分离出来。

题目要求进行删边操作，而对此我们不好维护，可以想到将操作倒序，从而维护加边的操作。

我们需要实现一个支持合并两个集合，查询集合最大值的数据结构，使用并查集即可。

复杂度 $O(n\log n)$ 。

T2 permutation

题目即判断 $x=\prod_{i=a}^{b} i$ 是否是 $y=\prod_{i=c}^{d} i$ 的因数。

显然是存不下的，而且模意义下也并没有整除相关的性质。

考虑唯一分解定理，一个数 a 是 b 的因数，当且仅当 a 每个素因子的指数都不超过 b 的对应素因子指数。

因此我们对 $[a,b],[c,d]]$ 每个数分解质因数， 结合线性筛，可以得到 40 分。

考虑贡献，如何求一个素数 p 在区间的指数？我们可以快速求出区间内有多少 p 的倍数，即 $\lfloor\frac{b}{p}\rfloor-\lfloor\frac{a-1}{p}\rfloor$ ，同时还要再计算 p^2 的倍数，这些对答案贡献是 $2(\lfloor\frac{b}{p^2}\rfloor-\lfloor\frac{a-1}{p^2}\rfloor)$ ，但因为 p^1 在前面已经统计过，所以不需要再乘上 2 的系数。对于 $p^k\le\max(b,d)$ 均进行计算，总时间复杂度为 $O(n)$。

T3 game

题目是最优化问题，可以考虑动态规划等算法。

可以设计状态 $dp[i][x][y]$ 表示考虑第 i 轮，一个人位于 (i,x) 的位置，另一个人位于 (i,y) 的位置，复杂度 $O(n^3)$ 。

显然这样并不优秀，因为显然第 i 轮必有一个人位于 (i,a_i) ，改状态设计为 $dp[i][x]$ 表示另一个人在 (i,x) 的位置。我们可以分类讨论，列出转移方程：

$$dp_{i,j}=\begin{matrix}dp_{i-1,j}+|a_{i-1}-a_i|,j\ne a_{i-1}\\ Min\{dp_{i-1,k}+|k-a_i|\},j=a_{i-1}\end{matrix}$$

观察到第一行的转移是 $O(1)*n$ ，而第二行的转移是 $O(n)*1$ ，这启发我们用数据结构优化转移。

- 对于第一行的转移，可以认为是进行两个区间加法操作。
- 对于第二行的转移，再进行一步分类讨论：
    - 对于 $k<a_i$ ，相当于查询 $\{dp_{i,k}-k\}$ 的最小值，再加上 a_i；
    - 对于 $k\ge a_i$ ，相当于查询 $\{dp_{i,k}+k\}$ 的最小值，再减去 a_i。
- 因此，我们可以用线段树分别维护 $\{dp_j\},\{dp_j+j\},\{dp_j-j\}$ ，支持区间加法，单点修改，查询区间最值。

时空复杂度为 $O(n\log m)$ 。