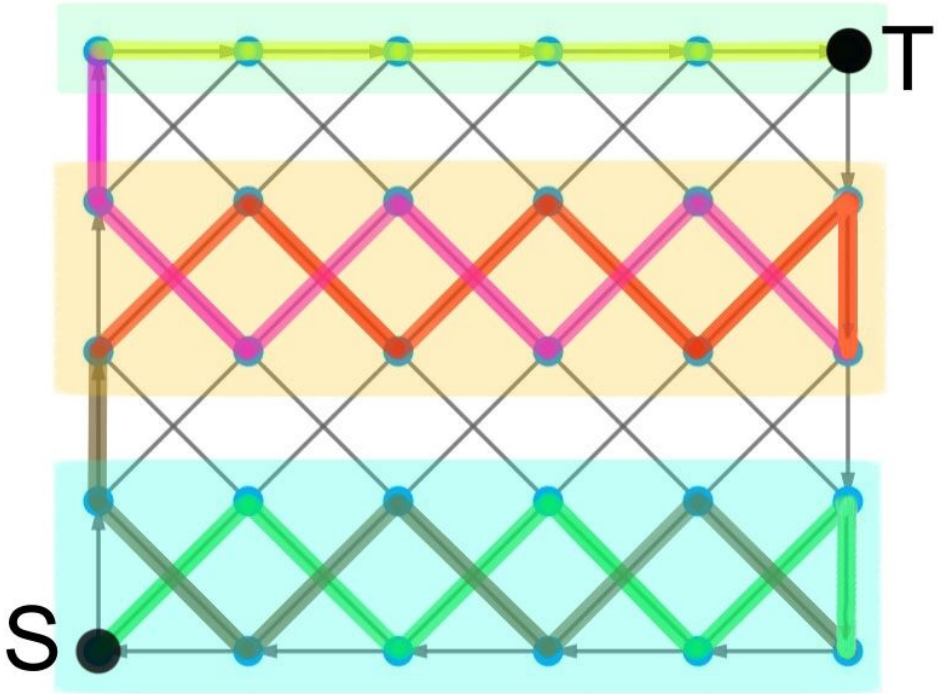
题解

A. 核酸检测

下面我们将说明，存在一条路线经过每一个隔离点恰好一次。因为每个隔离点至少要经过一次，所以不会存在比这更快的路线。总共有 $n(n+1)$ 个隔离点，该方案花费的时间是 $n(n+1)-1$ 个单位。

构造这样一条满足条件的路线的方法有很多，下面介绍一种较为简洁的方法。



上图展示了 $n=5$ 的一条路线（从隔离点 S 出发，依次经过绿色、棕色、红色、紫色、黄色，最后到达隔离点 T）。

如果 n 是奇数，那么一行含有偶数个隔离点，一共有奇数行。对于相邻两行的全部隔离点，可以从（这两行的）左下角出发，经过其余隔离点各一次，到达（这两行的）左上角，接着乘坐地铁来到上一行。重复这个过程，你将经过除第一行外的所有隔离点各一次，并到达整个方阵的左上角。接着乘坐地铁按顺序经过第一行的各个隔离点。

如果 n 是偶数，那么你可以想象将整个方阵顺时针旋转 90 度，旋转后的方阵满足一行含有偶数个隔离点、一共有奇数行，且拥有与原来相同的道路结构。这样就可以使用上一段描述的方法来完成构造。

当然，还有很多其他构造方法，它们都能被判定为正确。

B. 齐心抗疫

首先我们简单地描述一下题意：给定一棵树，找出两个点 x, y，设 $\mathrm{dis}(x, y)$ 表示 x 与 y 之间的距离，最大化 $\max(a_x, a_y) \cdot \mathrm{dis}(x, y)$。

由于 $\max(a_x, a_y) \cdot \mathrm{dis}(x, y) = \max(a_x \cdot \mathrm{dis}(x, y), a_y \cdot \mathrm{dis}(x, y))$，所以该问题等价于：合理选择 x, y，最大化 $a_x \cdot \mathrm{dis}(x, y)$。

于是我们只需要计算出距离每一个点最远的点有多远即可。

设树的直径两端点分别为 u, v，那么 u 或 v 必然距离点 x 最远。所以我们只需要求出直径后从直径两端点分别 DFS 求出每个点到直径两端点距离值即可。

时间复杂度为 $\mathcal{O}(n)$。

事实上，由于数据范围较小，这里也允许 $\mathcal{O}(n \log n)$ 的算法通过。

P.S. 关于树的直径：树的直径两端点是树上距离最远的两点，求直径时，只需要用 DFS 求出距离任意一个点最远的点 x，再找出距离 x 最远的点 y，即得到树的一对直径端点 x、y。

C. 病毒研究

可以发现，无论什么时候，我们都只能将病毒的活性确定在一个区间内。这让我们联想到以活性所在的区间为状态的动态规划算法。

设 $f(l,r)$ 为现在可以确定活性在 l 到 r 之间，需要的期望代价乘上 $r-l+1$ 的值。

转移如下：

- 若区间 $[l,r]$ 中的数属于多个不同的状态，那么 $f(l,r)$ 就等于 $[l,r]$ 中每种状态所在的区间的 f 值之和。
- 若区间 $[l,r]$ 中的数都属于 1 状态，那么 $f(l,r)=0$。
- 否则枚举前 m 种操作，可得 $f(l,r)=\min\{f(l-w_i,r-w_i)+(r-l+1)\times v_i\}$。

设活性的最大值为 L，那么答案为 $f(1,L)$，复杂度为 $\mathcal{O}(L^2m)$，无法通过此题。

我们先对所有操作跑一次完全背包，这样我们就可以使用一次操作来替代连续的多次操作。进行了完全背包以后，我们获得了新的 v_i 和 w_i。

这样之后，我们对于从 $f(l-w_i,r-w_i)$ 到 $f(l,r)$ 的转移，就可以强制让其满足 $[l-w_i,r-w_i]$ 中的数属于多个不同的状态。

那么如果用以上限制跑一遍记忆化搜索，就可以发现我们合法的状态数是$\mathcal{O}(L)$的。因为我们会用到的状态只有以下一种：对于所有的 $1\le p\le L$，设 i 为活性为 p 时所处的状态，所有的 $f(a_{i-1}+1,p)$ 和 $f(p,a_i)$。

如果我们在转移的时候使用前缀和优化，那么就可以以 $\mathcal{O}(Lm)$ 的复杂度解决这道题。

D. 抗疫斗争

首先有一个暴力算法。记 $A(i,j)$ 为当前还有 i 点行动值，上轮的行动值为 j 时的胜败情况，可以使用动态规划 $\mathcal{O}(n^2)$ 计算出所有的 h_m。这样就能够通过**前两档部分分**。

接下来让我们来考虑一下如何计算 h_m。

首先，如果 m 为奇数，那么人类方只要在第一轮中只付出一点行动值就能获胜，因为接下来双方都只能每轮付出一点行动值。

而如果 m 为偶数，那么人类方第一轮的行动值必须为偶数，否则轮到病毒方时剩余行动值就会变成奇数。由此我们得出，如果 m 为偶数，双方每次的行动值都一定为偶数。那么只要把每两点行动值合并，我们就能得到 $h_{2m}=2h_m$。

综上所述，我们可以得到

$$h_m=\begin{cases}1,\text{ if } m=2n+1,\\ h_n,\text{ if } m=2n.\end{cases}$$

那么不难得出 $h_m=\max_{2^k|m}2^k$，即 $h_m=\operatorname{lowbit}(m)$。

这里直接 $\mathcal{O}(n)$ 计算出所有 h_m，就能通过**第三档部分分**。

下面我们来考虑计算 $f_n=\sum_{m|n}h_m$ 的前缀和，记为 $S(n)$。我们有

$$S(n)=\sum_{i=1}^{n}h_i\left\lfloor\frac{n}{i}\right\rfloor$$

这里没有达到标程复杂度的数论求和算法可以通过**第四档部分分**。例如一个做法是对 $\left\lfloor\frac{n}{i}\right\rfloor$ 整除分块。但因为此类算法繁多，本题解并不加以描述。

我们考虑枚举 h_i 的取值，并统一计算其贡献。方便起见，我们记 $g(n)=\sum_i\left\lfloor\frac{n}{i}\right\rfloor$。

$$S(n)=g(n)+\sum_{k\geq1}(2^k-2^{k-1})g\left(\frac{n}{2^k}\right)$$

由于计算单个 $g(n)$ 的复杂度为 $\mathcal{O}(\sqrt{n})$，那么通过计算我们可以得到该算法的复杂度

$$\sum_{k\geq0}\sqrt{\frac{n}{2^k}}=\sqrt{n}\sum_{k\geq0}\left(\frac{1}{\sqrt{2}}\right)^k=(2+\sqrt{2})\sqrt{n}$$

即复杂度为 $\mathcal{O}(\sqrt{n})$。

当然，计算单个 $g(n)$ 时可以套用 SPOJ DIVCNT1 的算法，做到 $\mathcal{O}(n^{\frac{1}{3}}\log n)$ 的复杂度。由于意义不大，并没有针对这个算法设置一档部分分。

E. 名垂青史

储水量等于 `上轮廓` 减 `下轮廓`。`下轮廓` 只需要求区间和。

`上轮廓` 看作左右两个方向的阶梯。以下就左半边的阶梯阐述做法。

一段长度为 l 的区间，规定一段区间的 `左阶梯` 指的是前缀 $\max$ 。随着时间的推移，其 `左阶梯` 只会变 $\mathcal{O}(l)$ 次，因为每个数比左边所有数都大的时间是一个区间。

所以我们想要对序列建线段树，线段树的每个结点是个区间，对于每个线段树上的区间求出其 `左阶梯` 的变化。

求每个数在每个线段树节点上比所有左边的数都大的时间区间比较麻烦，可以直接拖动态半平面交的板子，也可以写在线段树每个结点建凸包二分。如果你选择后者，此时需要注意线段树上询问的时候定位的结点不要重复，看上去每个点有 $\mathcal{O}(\log n)$ 次单点对区间的凸包询问，但实际上将这些询问差分之后只会对应到线段树上共 $\mathcal{O}(\log n)$ 个结点。

然后，线段树上每个结点再开内层线段树维护一下左阶梯的变化。

区间询问的时候，从左到右访问线段树上的结点，每个结点上的内层线段树二分一下阶梯的拼接。

总时间复杂度 $\mathcal{O}(n \log^2 n)$。