

Tenzing and Tsondu 题解

题意

两个阵营玩卡牌游戏，轮流操作，操作的一方可以选择自己的 x 和对方的一个 y 。 x 变成 $x - y$ ， y 变成 $y - x$ 。当值 ≤ 0 就死亡，查询谁会赢。

题解

可以把 $x - y, y - x$ 看作 $x - \min(x, y), y - \min(x, y)$ ，所以只需要比较 $\sum a_i$ 和 $\sum b_i$ 即可。

Tenzing and Books 题解

观察位运算 OR 的性质：当某一个二进制位变成 1 后，其永远是 1。

这个性质告诉我们，如果有一本难度为 y 的书，且 $x|y \neq x$ ，由于选上它会使一个不需要的二进制位变成 1，所以这本难度为 y 的书不能被选。

我们可以对于每一个栈，找到一个最长的合法前缀。我们可以贪心的选择前缀中的所有书（理由：选择一个 $x|y = x$ 的书不会使答案变得更劣），计算出这些前缀中所有难度的 OR，检查是否为 x 即可。

复杂度 $O(n)$ 。

Tenzing and Balls 题解

假设最终被删除的是一些区间 $[l_1, r_1], [l_2, r_2], \dots, [l_m, r_m]$ ，则一定需要满足 $a_{l_i} = a_{r_i}$ 。

可以用 DP 求解最少需要保留多少个，设 dp_i 表示考虑前缀 $1, 2, 3, \dots, i$ 最少需要保留多少个，转移为 $dp_i = \min(dp_{i-1} + 1, \min \{dp_j | a_{j+1} = a_i, j + 1 < i\})$ 。

这个 DP 可以在 $O(n)$ 的时间复杂度内算出答案，只需要对每个 x 维护 $a_{j+1} = x$ 时 dp_j 的最大值即可。

Tenzing and His Animal Friends

把限制 u, v, y 看作 u, v 之间有一条权值为 y 的边，显然最终答案不会超过 1 到 n 的最短路。

可以通过以下方式构造，集合初始是 $\{1\}$ 。

每次只添加一个点，假设 i 在集合中的时间为 T_i ，则需要满足 $|T_u - T_v| \leq y$ ，这个是一个差分约束问题，可以使用最短路构造。

具体的，按照 1 到 i 的距离的顺序从小到达加点，每个集合的时间即为排序后相邻两个数到 1 的距离的差。

Tenzing and Triangle 题解

可以注意到，最优解不会存在相交的三角形，因为合并它们一定是不劣的，所以任意两个三角形是相离的。

三角形的斜边是直线 $y = k - x$ 上的一段区间，因此我们用区间 $[L, R]$ 表示左上角为 $(L, k - L)$ ，右下角为 $(R, k - R)$ 的三角形。

先假设所有点都会产生代价，用三角形覆盖一个点后能减少代价。设 $f(L, R)$ 表示被三角形 $[L, R]$ 覆盖的点的代价之和减去 $A(R - L)$ 。我们需要找若干个没有公共部分的区间 $[l_1, r_1], [l_2, r_2], \dots, [l_m, r_m]$ ，最大化 $\sum f(l_i, r_i)$ 。

记 dp_i 表示考虑前缀 $[1, i]$ ， $\sum f(l_i, r_i)$ 的最大值，有两种转移：

- i 不被任何区间覆盖， $dp_i \leftarrow dp_{i-1}$ 。
- i 被区间 $[j + 1, i]$ 覆盖， $dp_i \leftarrow dp_j + f(j + 1, i)$ 。

从小到大枚举 i ，维护 $g_j = dp_j + f(j + 1, i)$ ，当 $i - 1 \rightarrow i$ 时， g 会这样变化：

- $g_{0 \dots i-1}$ 全部减去 A 。
- 对于每个点 $(x, k - i)$ ， $g_{0 \dots x}$ 加上点的代价。

g 需要支持区间加，全局最大值（假设不合法的位置为 0），使用线段树即可。

时间复杂度 $O((n + k) \log k)$ 。

Tenzing and Tree 题解

设 $root$ 为所有黑点的重心，以 $root$ 为根，设 $size_i$ 为子树 i 里的黑点个数。

则这种方案的值为 $\sum_{i \neq root} k - 2 \cdot size_i = (n - 1) \cdot k - 2 \cdot \sum_{i \neq root} size_i$ 。

考虑把 i 这个点涂黑，它会对它的所有祖先产生贡献，也就是 $2 \cdot depth_i$ ，其中 $depth_i$ 表示 $root$ 到 i 的边数。

因为要最大化，所以可以贪心地选择 $depth_i$ 最小的。

但是怎么保证选择其他黑点之后 $root$ 恰好是重心呢？其实只需要对于所有的 $root$ 取 \max 就好，因为如果一个点不是重心，那么一些边的权值就是负数，一定比最优的答案劣。

枚举所有的 $root$ ，然后使用 bfs 按照到 $root$ 的距离加点，复杂度为 $O(n^2)$ 。

Tenzing and Random Operations 题解

在本题的做法开始前，有两点基本性质：

- 对于两个完全独立的随机变量 x_1, x_2 ，有 $E(x_1 x_2) = E(x_1) E(x_2)$ 。
- 对于 $(a + b) \times (c + d)$ ，有 $E((a + b) \times (c + d)) = E(ac) + E(ad) + E(bc) + E(bd)$ 。

回到本题，设 $x_{i,j}$ 为一个随机变量：当第 i 次操作将 $a_j := a_j + v$ 的时候，它的值是 v ，否则是 0。

则注意到答案就是 $\prod_{i=1}^n (a_i + \sum_{j=1}^m x_{j,i})$ 的期望。

运用上述的第二条性质将这个乘积拆开，则每项都是一些 a_i 和 x 的乘积。

具体而言，每一项有 n 个因子，对于每个 $i \in [1, n]$ ，要么 a_i 作为这一项的一个因子，要么某个 $x_{j,i}$ 作为这个项的一个因子。

然后来研究具体一项的期望，注意到若 $i_1 < i_2$ ，则 $E(x_{j,i_1} \times x_{j,i_2}) = E(x_{j,i_1}) \times v$ ，也就是如果 x_{j,i_1} 为 0 那么整个乘积就是 0，而当 x_{j,i_1} 是 v 的时候 x_{j,i_2} 一定是 v 。

因此对于一项的所有 x 因子而言，我们按照第一维下标分类，也就是把所有 $x_{j,\dots}$ 归入第 j 类，则对于每一类我们只关注第一个变量，如果它为 v ，则剩下的变量取值都是 v ，否则结果自然是 0。而注意到不同类的 x 变量取值完全独立（因为它们是两轮不同的操作决定取值的），所以两类变量乘积的期望可以拆成两类变量内部乘积的期望的结果再相乘。

我们的目标是求所有项的期望和，这个过程可以很好地和 dp 结合起来：

$dp(i, j)$ 表示我们确定了每一项的前 i 个因子，有 j 个类已经至少出现了一个数（如果位置 $i+1$ 的变量加入这些类带来的贡献是 v ，否则带来的贡献是 $\frac{i+1}{n} \times v$ ），转移容易 $O(1)$ 计算，讨论位置 $i+1$ 是放入 a_{i+1} 进去还是 $x_{\dots, i+1}$ 进去，如果是后者，再讨论它所属的类是 j 个已经出现过的还是 $m-j$ 个未出现过的。

时间复杂度 $O(n \times \min\{n, m\})$ 。

Tenzing and Random Real Numbers 题解

假设没有变量等于 0.5，因为等于 0.5 的概率为 0，把值小于 0.5 的变量称做白点，大于 0.5 的称做黑点，每种黑白染色是等概率的，所以我们可以对每种黑白染色计算合法的概率，再取平均值。

对于两个小于 0.5 的变量， ≤ 1 的条件一定满足， ≥ 1 的条件一定不满足，因此我们不需要考虑同色点之间的条件。白点 u 和黑点 v 之间的条件 $x_u + x_v \leq 1$ 满足当且仅当 $x_u \leq 1 - x_v$ ，记

$y_u = \min(x_u, 1 - x_u) = \begin{cases} x_u & (u \text{ is white}) \\ 1 - x_u & (u \text{ is black}) \end{cases}$ ，那么 y_u 可以看做 $[0, 0.5)$ 的随机变量，对于

≤ 1 的条件，白点的 y 要小于等于黑点的 y ，我们从白点向黑点连边，对于 ≥ 1 的条件，我们从黑点向白点连边。

我们得到了一张有向图，限制了 $y_{1\dots n}$ 的大小关系，设 $y_{1\dots n}$ 从小到大排序是 $y_{p_1}, y_{p_2}, \dots, y_{p_n}$ ，那么每种排列 p 都是等概率的，并且这个 p 有贡献当且仅当它是拓扑序，所以合法概率就是拓扑序数量除以 $n!$ 。

现在问题转化为一个计数问题，对于每种染色，统计拓扑序数量之和。现在我们不枚举染色，而直接枚举拓扑序，即枚举了一个排列 p 使得 $y_{p_1} < y_{p_2} < \dots < y_{p_n}$ ，统计能满足条件的染色数量，可以发现 ≤ 1 的条件限制了 p 中位置靠前的变量 < 0.5 ，而对另一个没有限制， ≥ 1 的条件限制了 p 中位置靠前的变量 > 0.5 。

接下来就状压 DP 了，设 dp_{mask} 表示拓扑序中已经加入了 $mask$ ，转移就枚举新加入点 u ，如果 $mask$ 中包含了所有和它之间有 ≤ 1 条件的变量，那么它可以染成黑色，如果 $mask$ 中包含了所有和它之间有 ≥ 1 条件的变量，那么它可以染成白色。

复杂度 $O(2^n n)$ 。

Tenzing and Necklace 题解

题意

你需要把一个环切成若干部分，切断一个位置需要花费一定的时间，查询最少时间满足每个部分的大小 $\leq k$ 。

题解

添加一个限制：“必须删掉 m 条边”。

考虑从小到大枚举最小的割边，

设枚举的最小的割边是 a_1 ，接下来的最优方案是 a_2, a_3, \dots, a_m 。

如果再枚举一个最小的割边： b_1 ，之后的最优方案是 b_2, b_3, \dots, b_m 。

假设 $a_i < a_{i+1}, b_i < b_{i+1}, b_1 > a_1$ 。

1. 可以只调整 b_2, b_3, \dots, b_m ，满足 $\forall 1 \leq i \leq m, b_i \geq a_i$ ，同时调整后的总花费不变。

调整方式如下：

找到最小的 i 满足 $b_i < a_i$ ，找到 i 后第一个 j 满足 $b_j \geq a_j$ ，若不存在则令 $j = m + 1$ 。

可以发现可以将 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 替换成 $(a_i, a_{i+1}, a_{i+2}, \dots, a_{j-1})$ ，任然是合法方案，同时将 $(a_i, a_{i+1}, a_{i+2}, \dots, a_{j-1})$ 替换成 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 也是合法方案，因为 $b_{i-1} \geq a_{i-1}, b_j \geq a_j$ 。

由于 a 和 b 都是固定 a_1, b_1 的最优方案，所以 $w_{b_i} + w_{b_{i+1}} + \dots + w_{b_{j-1}} = w_{a_i} + w_{a_{i+1}} + \dots + w_{a_{j-1}}$ ，因此将 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 替换成 $(a_i, a_{i+1}, a_{i+2}, \dots, a_{j-1})$ 总花费不变。

一直进行上述调整直到不存在 $b_i < a_i$ 。

同理可证可以只调整 a_2, a_3, \dots, a_m ，满足 $\forall 1 \leq i \leq m, b_i \geq a_i$ ，同时调整后的总花费不变

2. 若已满足 $\forall 1 \leq i \leq m, b_i \geq a_i$ ，则可以只调整 b_2, b_3, \dots, b_m ，满足 $\forall 1 \leq i < m, a_i \leq b_i \leq a_{i+1}$ ，同时调整后的总花费不变。假设 $a_1 < b_1 \leq a_2$ 。

调整方式如下：

找到最小的 i 满足 $b_i > a_{i+1}$ ，找到 i 后第一个 j 满足 $b_j \leq a_{j+1}$ ，不妨设 $a_{m+1} = +\infty$ 。

可以发现可以将 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 替换成 $(a_{i+1}, a_{i+2}, a_{i+3}, \dots, a_j)$ ，任然是合法方案，同时将 $(a_{i+1}, a_{i+2}, a_{i+3}, \dots, a_j)$ 替换成 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 也是合法方案，因为 $b_{i-1} \leq a_i, b_j \leq a_{j+1}$ 。

由于 a 和 b 都是固定 a_1, b_1 的最优方案，所以 $w_{b_i} + w_{b_{i+1}} + \dots + w_{b_{j-1}} = w_{a_{i+1}} + w_{a_{i+2}} + \dots + w_{a_j}$ ，因此将 $(b_i, b_{i+1}, b_{i+2}, \dots, b_{j-1})$ 替换成 $(a_{i+1}, a_{i+2}, a_{i+3}, \dots, a_j)$ 总花费不变。

同理可证可以只调整 a_2, a_3, \dots, a_m ，满足 $\forall 1 \leq i < m, a_i \leq b_i \leq a_{i+1}$ ，同时调整后的总花费不变。

3. 若 $b_1 > a_2$ ，则可以调整 b_1, b_2, \dots, b_m ，满足 $b_1 \leq a_2$ ，且总花费不会变大

调整方式如下：

找到最小的 j 满足 $b_j \leq a_{j+1}$ ，不妨设 $a_{m+1} = +\infty$ 。

可以发现可以将 $(a_2, a_3, a_4, \dots, a_j)$ 替换成 $(b_1, b_2, b_3, \dots, b_{j-1})$ ，任然是合法方案，同时将 $(b_1, b_2, b_3, \dots, b_{j-1})$ 替换成 $(a_2, a_3, a_4, \dots, a_j)$ 也是合法方案，因为 $b_j \leq a_{j+1}$ 。

由于 a 是固定 a_1 的最优方案，所以 $w_{b_1} + w_{b_2} + \dots + w_{b_{j-1}} \geq w_{a_2} + w_{a_3} + \dots + w_{a_j}$ ，因此将 $(b_1, b_2, b_3, \dots, b_{j-1})$ 替换成 $(a_2, a_3, a_4, \dots, a_j)$ 总花费不会变大。

结合以上结论，我们可以得到一个必须割掉 m 条边的做法：

令 $a_1 = 1$ ，找到一组最优解 $a_1, a_2, a_3, \dots, a_m$ 。

之后可以假设, 所有的 b_i 都满足, $a_i \leq b_i \leq a_{i+1}$ 。

得到一个分治的做法, 令 $solve((l_1, r_1), (l_2, r_2), (l_3, r_3), \dots, (l_m, r_m))$ 表示当前需要算出所有 $l_i \leq b_i \leq r_i$ 的最优解。

若 $l_1 > r_1$ 则结束, 否则令 $x = \lfloor \frac{l_1 + r_1}{2} \rfloor$, 可以在 $O(\sum r_i - l_i + 1)$ 的复杂度内使用 dp 算出 $b_1 = x$ 的答案与方案, 递归计算 $solve((l_1, b_1 - 1), (l_2, b_2), (l_3, b_3), \dots, (l_m, b_m))$ 和 $solve((b_1 + 1, r_1), (b_2, r_2), (b_3, r_3), \dots, (b_m, r_m))$ 。

时间复杂度分析: $\sum r_i - l_i + 1 = (\sum r_i - l_i) + m$ 。如果相邻两个部分的总和 $\leq k$, 则可以合并, 一定不是最优方案, 所以不妨设 $m \leq 2 \lceil \frac{n}{k} \rceil$ 。假设第一段的长度是 $r_1 - l_1 + 1 = O(k)$, 时间复杂度为 $O(n \log k + mk) = O(n \log k)$ 。

接下来我们需要对于所有可能的 m 算出答案, 取 \min 就是最终的答案。

割去第一条边后如果最优方案需要割掉 m' 条边, 与之前的证明类似, 可以调整全局最优方案, 使得 $|m - m'| \leq 1$ 且总代价不会变大。