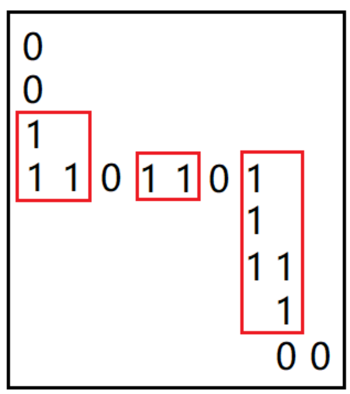
石门中学2021联赛模拟 Day2
解题报告

command_block 2021.10.6

1 跑路

考虑若已经选定一条从左上角到右下角的路径，如何用最小的操作使得该路径合法。

不难发现，对于路径的每一段连续的 1 ，恰能用一次子矩形翻转将其处理。



若尝试使用一次子矩形翻转同时处理两段 1，则必然干扰到中间的 0 。不难证明这已经最优方案。

于是，一条路径的“代价”为该路径上 1 的连续段数目，也就是连续出现的 01 的数目。

我们使用简单的二维 DP 求出路径的最小代价。

具体地，记 $dp[i][j]$ 表示左上角到 (i,j) 的路径的最小代价，则有转移

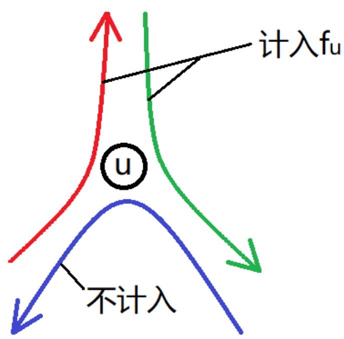
$$dp[i][j]=\min\Big(dp[i-1][j]+[B_{i-1,j}=0][B_{i,j}=1],dp[i][j-1]+[B_{i,j-1}=0][B_{i,j}=1]\Big)$$

答案为 $dp[n][m]$ ，复杂度为 $O(n^2)$。

2 清扫

首先任取一个不是叶子的点为根。若 $n = 2$ 则都是叶子，特判。当 $n \geq 3$ 则必然存在一个非叶节点。

记 f_u 表示方案中穿过（自下而上或自上而下）u 的路径数目，对于恰好在 u 处弯折的路径则不算在内。对于叶节点，有 $f_u = a_u$。



对于非叶节点，记 $s_u = \sum\limits_{u \to v} f_v$ ，即从儿子延伸上来的路径条数。

其中 f_u 条穿过 u ，其余在 u 处（来自不同分支）两两配对形成弯折。弯折数目为 $\dfrac{s_u - f_u}{2}$ 。

可以列出方程：

$$a_u = \frac{s_u - f_u}{2} + f_u = \frac{s_u + f_u}{2}$$

则

$$f_u = 2a_u - s_u$$

于是可以用一次 dfs 求出所有的 f, s。

然后考虑如何判定，一个显然的条件是 $f_u, s_u \geq 0$。

此外，f_u, s_u 各有上界。记 c_u 为子树内**叶节点的权值和**，则：

- $f_u \leq c_u$ ：穿过 u 的路径不可能多于子树内的“路径端点”总数。
- $\dfrac{s_u - f_u}{2} \leq s_u - \max\limits_{v\text{为}u\text{的儿子}} f_v$ ：当某一个儿子提供的路径过多，不同分支配对时会余下一些。若满足这个条件，则一定有合法的配对方案：每次令最多的分支与最少的分支配对。

当 u 点为根时，额外要求 $f_u = 0$。

若满足上述条件，不难归纳证明可以构造出符合要求的方案。

复杂度 $O(n)$。

3 定向

3.1 分析

对于边 $u \to v$ ，若将其反向:

情况一：若 u, v 在同一个强连通分量内。

强连通分量个数不可能增加，但该强连通分量可能被破坏。

若图中存在一条 $u \to v$ 的路径，且不含该边，则不会破坏，反之则会。

情况二：若 u, v 不在同一个强连通分量内。

若图中存在一条 $u \to v$ 的路径，且不含该边，则会增加一个强连通分量，反之无变化。

记 $f(u,v)$ 表示“u, v 在同一个强连通分量内”，$g(u,v)$ 表示图中存在 $u \to v$ 的路径，且不含边 $u \to v$。

综上所述，只有 $f(u,v), g(u,v)$ 两者之一成立，边 $u \to v$ 的答案才为 1 ，否则为 0。

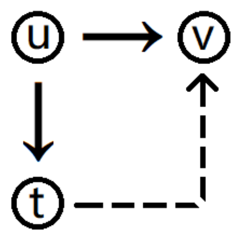
对于 f ，容易使用 Tarjan 算法求出。问题的核心在于 g 的求解。

3.2 算法一：$O(nm)$

考虑对于每个 u 分别求出 $g(u, \_)$。

从 u 开始 dfs ，将 u 的边表顺序翻转，再次 dfs。

若点 v 在两个 dfs 树中深度均为 1 ，则说明 $g(u,v) = 0$ ，否则 $g(u,v) = 1$。



证明：若 $g(u,v) = 0$ ，则点 v 在两个 dfs 树中深度显然均为 1 。

下证点 v 在两个 dfs 树中深度均为 1 时，$g(u,v) = 0$ 。即 $g(u,v) = 1$ 时 v 在两个 dfs 树中深度不可能同时为 1 。

$g(u,v) = 1$ 时，存在另一条不走 $u \to v$ 的路径能从 u 到 v ，记该路径的第一条边为 $u \to t$。

在两次 dfs 中，必然恰有一次先考虑 $u \to t$ 而后考虑 $u \to v$ ，由于 dfs 的特性，一定会从 t 进而访问到 v 。后面考虑 $u \to v$ 时，就不会将其加入 dfs 树。

□

3.3 算法二：$O(n^3/w+m)$

仍然考虑对于每个 u 分别求出 $g(u,\_)$。

对于某个 $i \neq u$ ，$g(u,i)=1$ 当且仅当存在 $j \neq i,u$ ，满足存在简单路径 $u \to j \rightsquigarrow j$。

将 u 点删除，将所有直接与 u 相连的点记为 $v_1, v_2, ..., v_k$。

顺次考虑 $i=1\sim k$ ，从 v_i 开始搜索（不重复遍历之前搜过的点），所有搜到的点 t（除了 v_i 本身）都能满足 $g(u,t)=1$。

为了处理 $u \to v_j v_i$ $(i<j)$ 的情况，还需反过来再跑一次。

这样的搜索复杂度是 $O(n+m)$ 的，瓶颈在于，无论目的地是否被标记过，我们都要再次检查出边。

考虑利用 bitset 进行优化，记目前还未发现 $g(u,t)=1$ 的 t 的集合为 S ，点 u 的出边的集合为 G_u 。我们搜索时，只需考虑集合 $S \cap G_u$ 内的元素。

下面的代码段可以用 $O(|T|n/w)$ 的代价找出 bitset T 内的所有元素。

```
for(int i=T._Find_first();i!=T.size();i=T._Find_next(i))
```

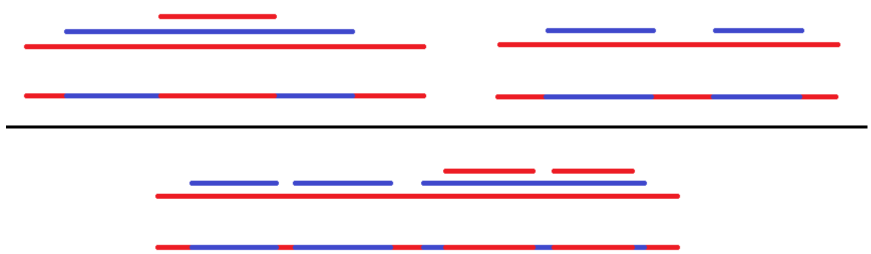
每个点只会被搜索到一次，故单次搜索复杂度为 $O(n^2/w)$。

4 染色

4.1 判定某种颜色序列是否能被生成

定义一个颜色序列 C 的**最简操作序列**为：该序列 P 能染出 C，但 P 的任意子序列都无法染出 C。

先考虑颜色序列不存在黑球，即只有红蓝球的情况。观察可能有怎样的染色方案，如下图：



不难发现，若有 k 个蓝色段，则所需的染色次数为 $k+1$ 次，红色和蓝色的染色次数不定，在 $[1,k]$ 之间，但红色和蓝色都必须各自至少染一次。

（特殊地，若没有蓝色，则只需染一次红色）

在此基础上，还有染色顺序的限制。若我们决定染 a 次蓝色，最好的方案是：先染一次红色，再染 $a-1$ 段蓝色，然后染一大段蓝色，再在这段蓝色上染完剩下的红色。

对最简操作序列的（充要）要求：满足个数 $=k+1$，最前面是 $\mathtt{r}$，第二个是 $\mathtt{b}$。

接着考虑由黑球分隔的多段有色区间。

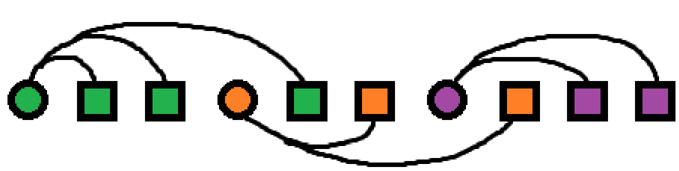
先不考虑纯红色段，设有 c 个杂色（红蓝都有）段。我们要在整个操作序列中选出若干个子序列，分别作为每个有色区间的最简操作序列。

我们先要选出 c 个子序列 $\{\mathtt{r},\mathtt{b}\}$ 作为 c 个最简操作序列的开头。

假设我们已经选定了 c 个子序列 $\{\mathtt{r},\mathtt{b}\}$，考虑如何判定。接下来我们要为每个最简操作序列添加指定数目的字符，且满足字符在开头的 $\{\mathtt{r},\mathtt{b}\}$ 右侧。

记第 i 个 $\{\mathtt{r},\mathtt{b}\}$ 后面剩余的未选择字符数目为 t_i，对应的最简操作序列还需要 s_i 个字符。

一个经典的贪心策略是，按 t_i 从大到小（即按开头位置从左到右）排序，依次考虑，每次尽量靠左取字符。



（图中圆形代表一个 {r,b}，方形代表其他字符）形式化地，有解的充要条件为

$$\forall_i,\ t_i \ge \sum_{j=i}^{c} s_j$$

进一步考虑如何排列 s 的顺序更优。不难发现，若将 s 降序排列，则每个 $\sum_{j=i}^{c} s_j$ 都取得最小值，此时最优（最可能满足条件）。

再考虑有 a 个纯红色段的情况，根据经典贪心，取出 a 个尽量靠前的 r 处理纯红色段即可。

4.2 DP 计数

枚举 a,c，然后可以确定 $t_{1\sim c}$。

先考虑在确定 s 的情况下，如何计算颜色序列的方案数。

先考虑颜色段之间的排列顺序。首先是杂色段和纯红色段混合的方案数，为 $\binom{a+c}{a,c}$。

然后是杂色段内部顺序数，即 s 序列的不同排列数，这是典型的多重排列。记 $c_i = \sum[s_j = i]$ 则方案数为 $(\sum s_i)!\big/\prod c_i!$

接下来考虑确定顺序后的方案数。

各个“颜色段”长度不定，可以看做装球的盒子。

我们发现，对于一个杂色段对应的 s_i（注意，由于总字符数为 s_i+2，则蓝色段个数为 s_i+1），会产生 $2s_i+1$ 个至少有一个球的颜色段，以及 2 个可以为空的颜色段（两端的红球段）。

对于 a，会产生 a 个至少有一个球的（纯红）颜色段。

对于黑色的部分，会产生 $a+c-1$ 个至少有一个球的颜色段，以及 2 个可以为空的颜色段（两端的黑球段）。

综上，非空盒的个数为 $2\sum s_i + c + 2a - 1$ 个，而可空盒为 $2c+2$ 个，总球数为 n。这是经典组合数学问题，容易用组合数求出方案，具体计算略去。

接下来对所有不同的 s 序列进行统计。

设 $dp[i,j,k]$ 表示填写了 $s_{i\sim c}$，目前 $\sum s = j$，且 $s_i = k$ 的方案数（不同排列数）。

枚举在 s 中选多少个 k，则有转移：

$$dp[i,j,k] = \sum_{p\ge 1}\binom{c-i+1}{p}\sum_{g<k} dp[i+p, j-pk, g]$$

式子中 $\binom{c-i+1}{p}$ 用于处理多重排列，是将 p 个 k 归并到目前的 s 中的方案数。

前文中条件 $\forall_i,\ t_i \ge \sum_{j=i}^{c} s_j$ 在这里体现为要求 $j \le t_i$ 且 $\forall p' \in [1,p],\ j - p'k \le t_{i+p'}$。

最终的答案是 $dp[1,\_,\_]$，根据第二维的 $\sum s$（用组合数）计算贡献系数。

使用前缀和优化 DP，即可做到 $O(n^3 \log n)$。

算上枚举 a,c 的复杂度，总复杂度为 $O(n^5 \log n)$。由于常数非常小，可以轻松通过。