简单模拟赛

Subway (subway, 2s, 512M)

Byteland 有 n 个城市和 $n!$ 辆私家车。由于交通不便，市民们怨声载道。今天，市长宣布 Byteland 多了 k 条高速地铁。每条高速地铁由若干段组成，每段连接两个城市里的对应地铁的地铁站，且只由每条高速地铁就可以从每个城市到达其他所有城市。

每个城市都有所有高速地铁对应的 k 个地铁站。根据设计不同，在每个城市进和出每个地铁站需要一定的时间，且进站需要付对应高速地铁的一张票。一旦进站了，在出站之前你可以任意使用**该高速地铁**对应的段进行穿梭。由于 Byteland 科技先进，你并不需要等待地铁，所以乘坐每段需要的时间是已知的。由于每条高速地铁由不同的公司承建，要从地铁 x 到地铁 y，你需要出站（走出地铁 x 的地铁站）再进站（走进**同一城市**中地铁 y 的地铁站）。

你现在处于城市 1。使用一些手段，你获得了 k 种高速地铁的票各一张，你希望使用高速地铁到达城市 n。注意你开始并不在任何地铁站中，且你不能在地铁站内结束旅途（即你需要在城市 n 出站后结束）。你想知道所需的最短时间。

输入格式

第一行两个正整数 n, k，表示城市数和高速地铁数。

对于每个高速地铁 $t \in [1, k]$，首先输入两行。第一行 n 个非负整数 $a_{t,i}$ 表示城市 i 进地铁 t 对应站所需时间，第二行 n 个非负整数 $b_{t,i}$ 表示城市 i 出地铁 t 对应站所需时间。接下来一行一个正整数 m_t，表示该地铁的段数。对于每一段输入一行三个非负整数 u, v, w，表示该段连接的两个地铁站所在城市 u, v（$u \neq v$）和乘坐该段所需时间 w。

保证仅使用每个高速地铁都能从任意一个城市到达其他所有城市，且每个高速地铁在每两个城市之间最多只有一个段。

输出格式

输出一个非负整数，表示从城市 1 到城市 n 所需的最短时间。

样例输入1

```
4 2
1 2 1 4
4 3 2 1
4
2 1 2
2 3 39
4 2 48
3 4 1
1 1 2 2
1 0 0 1
3
1 4 29
2 3 3
4 3 7
```

样例输出1

```
18
```

一种最优策略是 1 号线从城市 1 坐到城市 2，用时 1（城市 1 进站）+2（乘坐城市 1 到城市 2 的那段地铁）+3（城市 2 出站），再用 2 号线从城市 2 坐到城市 4，用时 1（城市 2 进站）+3（乘坐城市 2 到城市 3 的那段地铁）+7（乘坐城市 3 到城市 4 的那段地铁）+1（城市 4 出站），总用时 18。

样例输入2

```
2 2
0 0
0 0
1
2 1 1
0 0
0 0
1
1 2 2
```

样例输出2

```
1
```

不一定要用完所有的票。

样例输入3、样例输出3

见选手文件夹下 `ex_subway3.in` 和 `ex_subway3.ans`。

样例输入4、样例输出4

见选手文件夹下 `ex_subway4.in` 和 `ex_subway4.ans`。记得爆int。

数据范围

对于所有数据，$2 \le n \le 500$，$n-1 \le m_t \le 1000$，$1 \le k \le 8$，$0 \le a_{t,i}, b_{t,i}, w \le 10^9$。

对于 20% 的数据，$n \le 5$，$k \le 2$。

对于 30% 的数据，$n \le 100$，$k \le 3$。

对于 50% 的数据，$n \le 100$，$k \le 5$。

对于 70% 的数据，$k \le 6$。

对于另外 10% 的数据，$k = 1$。

Distanced (distanced, 2s, 512M)

Bytelane 上停着 n 辆车。文明城市评比就要开始了，作为 Bytecity 市长的你决定移除一部分车使得街道更文明（也可以什么也不移除）。

一个文明的街道首先应该整齐：车不应该停得过于散乱。我们称两辆车 x 和 y 属于同一个联通块当且仅当它们的距离不超过 t（“直接相邻”），或存在车 c 使得 x 和 c 属于同一个联通块且 c 和 y 属于同一个联通块（“间接相邻”）。街道的散乱度为联通块的数量（即两两不属于同一个联通块的车集合的最大可能大小）。

一个文明的街道还应该尽量美观：第 i 辆车有 a_i 的美观度，而街道的美观度是剩余车辆美观度的总和。由于部分车辆可能年久失修，a_i 可能为负。

由于你还没有决定好街道的整齐程度，对于每个 $s \in \{1, 2, \cdots, n\}$，你想要知道散乱度**不超过** s 的移除方案中美观度的最大值。

输入格式

第一行两个正整数 n, t，表示车辆数和直接相邻的距离上限。

第二行 n 个整数 $x_1, x_2, \cdots, x_n$，依次给定每辆车在数轴上的位置。车 a 和车 b 的距离为 $|x_a - x_b|$。

第三行 n 个整数 $a_1, a_2, \cdots, a_n$，依次给定每辆车的美观度。

输出格式

输出一行 n 个整数，第 i 个表示散乱度不超过 i 的移除方案中美观度的最大值。

样例输入1

```
5 2
5 1 2 4 8
5 3 4 -1 1
```

样例输出1

```
11 12 13 13 13
```

要求散乱度不超过 1 时一种最优方案是保留前四辆车。要求散乱度不超过 2 时一种最优方案是保留前三辆车。

样例输入2、样例输出2

见选手文件夹下 `ex_distanced2.in` 和 `ex_distanced2.ans`。

数据范围

对于所有数据，$1 \le n \le 10^5$，$1 \le t \le 10^9$，$|x_i|, |a_i| \le 10^9$，x_i 两两不同。

对于 20% 的数据，$n \le 20$。

对于 40% 的数据，$n \le 400$。

对于 70% 的数据，$n \le 2000$。

对于另外 10% 的数据，$a_i \ge 0$。

Color (color, 3s, 512M)

Bytejump 近期推出了一款新玩具：Bytecolor。在这个玩具中，玩家需要给 Bytecity 上色。

Bytecity 的结构可以由一棵树描述：有 $n - 1$ 条双向道路连接所有城市。调色板中有 k 种颜色，一开始某些城市已经上好色了（也是 k 种颜色中的），玩家需要用调色板中的颜色给剩余的每个城市上色（颜色可以重复），且相邻城市不能同色。

你想要知道玩家一共能得到多少种可能的树（即有多少种不同的染色方案），对 998244353 取模。

输入格式

第一行两个正整数 n, k，表示城市数和颜色数。

接下来 n 行，第 i 行一个非负整数 $c_i \in [0, k]$，表示城市 i 的开始颜色，0表示尚未上色，否则表示已经上好的颜色编号。

最后 $n-1$ 行描述树边。第 i 行两个正整数 u_i, v_i，表示第 i 条树边连接的两个城市。

输出格式

输出一行一个 $[0, 998244352]$ 的整数，表示玩家一共能得到多少种可能的树对 998244353 取模。

样例输入1

```
4 3
1 0 0 2
1 2
2 3
3 4
```

样例输出1

```
3
```

有 1212, 1232, 1312 三种上色方案。

样例输入2、样例输出2

见选手文件夹下 `ex_color2.in` 和 `ex_color2.ans`。

样例输入3、样例输出3

见选手文件夹下 `ex_color3.in` 和 `ex_color3.ans`。

数据范围

对于所有数据，$1 \le n \le 10^5$，$1 \le k \le 10^9$，$0 \le c_i \le k$。

对于 10% 的数据，$n, k \le 9$。

对于 20% 的数据，$n, k \le 50$。

对于 40% 的数据，$n, k \le 1000$。

对于 60% 的数据，$n \le 1000$。

对于另外 20% 的数据，$k \le 10^5$。

Grid (color, 3s, 512M)

Bytegrid 是 Byteland 最新潮的游戏。在这个游戏中，玩家在一个 $n \times n$ 的网格上辗转腾挪。

具体地，玩家可以从任意格子出发，一开始属于失能状态。第 i 行第 j 列的格子在时刻 $a_{i,j}$ 会充能，如果当时玩家处于该格子就可以花 0.1 秒转移到任意相邻（四联通）格，并再次变为失能状态。每到达一个格子玩家都会获得一个新的奇遇，因此玩家可能的移动方案数决定了每局游戏的丰富程度。

作为 Bytegrid 的开发者，你计划上线一个新版本，其中网格有一个格子不会被充能（但仍然可以出发或被移动到）。你有 q 种不同的更新计划，每次计划给定了 i, j，指定了第 i 行第 j 列的格子不会被充能（相当于将 $a_{i,j}$ 置为 $-\infty$），你希望获取在移除该格后玩家可能的移动方案数对 998244353，其中移动方案指的是依次经过的格子序列。**每次计划都是在原网格上进行改动，即计划两两独立。**

输入格式

第一行一个正整数 n，表示网格大小。

接下来 n 行，第 i 行 n 个正整数 $a_{i,1}, a_{i,2}, \cdots, a_{i,n}$，表示初始第 i 行每个格子的充能时刻。

接下来一行一个正整数 q，表示计划的个数。

接下来 q 行，每行三个正整数 x, y，表示该次计划使第 x 行第 y 列的格子无法被充能。

输出格式

输出 q 行，第 i 行一个整数，表示第 i 个计划中玩家的移动方案数，对 998244353 取模。

样例输入1

```
2
1 2
4 4
3
1 1
2 1
2 2
```

样例输出1

```
12
16
14
```

在第一次测试中，一种可能的移动序列为 $(1,2),(2,2),(2,1)$：第一次测试中玩家可以在 $(1,2)$ 出发，在 2 时刻被充能后转移到 $(2,2)$，在 4 时刻被充能后转移到 $(2,1)$。

注意 $(1,2),(2,2),(2,1),(1,1)$ 并不是可能的移动序列：到达 $(2,1)$ 时 4 时刻已经过去，所以玩家无法再在 $(2,1)$ 上被充能。

第一次测试中所有合法的移动序列：$[(1,1)]$, $[(1,2)]$, $[(1,2),(1,1)]$, $[(1,2),(2,2)]$, $[(1,2),(2,2),(1,2)]$, $[(1,2),(2,2),(2,1)]$, $[(2,1)]$, $[(2,1),(1,1)]$, $[(2,1),(1,2)]$, $[(2,2)]$, $[(2,2),(1,2)]$, $[(2,2),(2,1)]$，共 12 个。

样例输入2、样例输出2

见选手文件夹下 `ex_grid2.in` 和 `ex_grid2.ans`。

样例输入3、样例输出3

见选手文件夹下 `ex_grid3.in` 和 `ex_grid3.ans`。

数据范围

对于所有数据，$2 \le n \le 500$，$1 \le q \le 10^6$，$1 \le a_{i,j} \le 10^9$。

对于 20% 的数据，$n, q \le 40$。

对于 40% 的数据，$n, q \le 200$。

对于 60% 的数据，$q \le 500$。

对于另外 20% 的数据，输入中的所有时刻（a）两两不同。