# 毕业照

**时间限制：** 1.0 秒

**空间限制：** 512 MiB

刷新

**相关文件：** 下发文件 (/staticdata/225.dftkobl1MKKCKIDP.pub/a13mLhgS7QPNhvQl.down.zip/down.zip)

## 题目描述

毕业季快到了，小A、小B、小C、小D、小E 五个人准备一起去拍毕业照。

他们准备排成一列拍照，但大家也有自己的一点小心思。比如小A 想和 小B 站相邻位置拍照，小C 不想和小D 站相邻位置拍照。

给出同学之间谁和谁站相邻位置拍照，谁不和谁站相邻位置拍照的限制，问有多少种满足条件的拍照站位方案。

## 输入格式

从标准输入读入数据。

输入的第一行包含一个整数 $m$，表示有 $m$ 条限制。

接下来的 $m$ 行，每行格式为 `id x y` 。其中

- `id` 为 $1$，表示 `x` 和 `y` 要站相邻位置拍照； `id` 为 $2$，表示 `x` 和 `y` 不能站相邻位置拍照。
- `x` 和 `y` 为 `A` 到 `E` 之间的字符，分别表示小A 到 小E。保证 `x` 和 `y` 不同。

## 输出格式

输出到标准输出。

输出一个非负整数，表示有多少种满足条件的拍照站位方案

## 样例1输入

```
1
1 A B
```

## 样例1输出

```
48
```

# 样例1解释

只有一条限制：小 A 和 小 B 站一起拍照，那么一共有 $48$ 种方案。

# 数据范围

对于所有测试数据，$m \leq 3$。

通过每个测试点可以获得 $10$ 分：

- 对于测试点 $1$，保证 $m = 0$。
- 对于测试点 $2 \sim 3$，保证 $m = 1$。
- 对于测试点 $4 \sim 5$，保证 $id = 1$。
- 对于测试点 $6 \sim 7$，保证 $id = 2$。
- 对于测试点 $8 \sim 10$，无特殊限制。

语言和编译选项

| # | 名称 | 编译器 | 额外参数 | 代码长度限制 |
|---|---|---|---|---|
| 0 | g++ | `g++` | `-O2 -DONLINE_JUDGE` | 65536 B |
| 1 | gcc | `gcc` | `-O2 -DONLINE_JUDGE` | 65536 B |
| 2 | java | `javac` | | 65536 B |
| 3 | python2 | `python` | | 65536 B |
| 4 | python3 | `python3` | | 65536 B |

递交历史

| # | 状态 | 时间 |
|---|---|---|
| 69491 | **Wrong Answer** | 2023-06-30 12:36:17 **有效递交** |
| 67602 | **Wrong Answer** | 2023-06-30 09:57:31 |
| 67514 | **Wrong Answer** | 2023-06-30 09:50:59 |
| 67432 | **Wrong Answer** | 2023-06-30 09:44:48 |
| 67090 | **Wrong Answer** | 2023-06-30 09:04:31 |

1

# 卡卡颂

**时间限制：** 3.0 秒

**空间限制：** 512 MiB

刷新

**相关文件：** 下发文件 (/staticdata/224.cFrLK3hRHn95MSL9.pub/izXuW8PKUjuPYzsv.down.zip/down.zip)

## 题目描述

卡卡颂是一个桌游，玩家需要取出 $n$ 张卡片放置下来，形成版图。请注意，本题中为了简化起见，部分设定和原版游戏不同，请仔细阅读。如果你在理解过程中产生了障碍，建议参考样例进行直观理解。

在平面直角坐标系中考虑，一张卡片是一个 $1 \times 1$ 的正方形，必须以边与坐标轴平行的方式被放置。设第 $i$ 张被放置的卡片的中心坐标为 $(x_i, y_i)$。

放置卡片有如下要求：

- 卡片中心位于整点，即 $x_i, y_i$ 为整数。
- 不能和之前的卡片重叠，即不存在 $j < i$ 使得 $(x_j, y_j) = (x_i, y_i)$。
- 除第一张卡片外，放置每张卡片时都存在一个相邻的位置之前放置过卡片，即存在 $j < i$ 使得 $(x_j, y_j)$ 为 $(x_i - 1, y_i)$，$(x_i + 1, y_i)$，$(x_i, y_i - 1)$，$(x_i, y_i + 1)$ 之一。

卡片上可能有若干段道路或若干块城堡。我们用 `U,D,L,R` 四个字母分别表示一张卡片的上、下、左、右边界。第 $i$ 张卡片上包含的道路段数和城堡块数之和为 $c_i$，规则如下：

- 一块城堡可以用 `U,D,L,R` 的一个非空子集 $S$ 表示，表示它占据了这张卡片上子集 $S$ 所描述的这些边界。
- 一段道路可以用 `U,D,L,R` 的一个非空子集 $S$ 表示，满足 $|S| \leq 2$。如果 $|S| = 1$ 则表示这段道路从相应边界连到卡片内部中断；如果 $|S| = 2$ 则表示这段道路的两端分别是 $S$ 中的两个边界。
- 如果有两张分别位于 $(x, y), (x + 1, y)$ 的卡片（它们左右相邻），记前者的右边界为 $C$，后者的左边界为 $D$。那么 $C$ 和 $D$ 上的建筑类型必须相同（同时没有东西、同时是道路或同时是城堡）；且若 $C$ 和 $D$ 各属于一块城堡/一段道路，那么它们将视为同一块城堡/同一段道路；
- 如果有两张分别位于 $(x, y), (x, y + 1)$ 的卡片（它们上下相邻），记前者的上边界为 $C$，后者的下边界为 $D$。那么 $C$ 和 $D$ 上的建筑类型必须相同（同时没有东西、同时是道路或同时是城堡）；且若 $C$ 和 $D$ 各属于一块城堡/一段道路，那么它们将视为同一块城堡/同一段道路；
- 如果一块城堡/一段道路中不存在任意一个没有被连接的边界，那么就称它是闭合的：因为此时该城堡/道路不可能再被扩大。

现在，游戏有 $m$ 位玩家，编号为 $1, \ldots, m$，每名玩家有 $k$ 个战士。玩家可以将战士派遣到某一块城堡上，规则如下：

- 玩家只能在刚刚放置的卡片的城堡/道路上派遣战士，且所有玩家在一张卡片上总共只能派遣至多一个战士。
- 如果一个玩家的 $k$ 个战士全都被派遣出去且未收回，则不能再派遣战士。

- 玩家在一块城堡/一段道路（参考上文的描述，并不局限于这一张卡片上）上派遣战士时，需要保证当前这块城堡/这段道路上没有任何玩家（包括自己）曾经派遣过战士。注意，虽然这里要求一块城堡/一段道路上只有一个战士，但如果之后加入新的卡片使得原先不连通的两个城堡/道路连通了，那么这上面就可能存在多个战士。

一块城堡的分数等于包含这块城堡的卡片张数乘以 $2$，一段道路的分数等于包含这段道路的卡片张数。当一张卡片被放置，且上面的战士被派遣后，若此时某一块城堡/一段道路闭合，进行如下结算：

- 统计每名玩家在这块城堡/这段道路上派遣的战士个数，若无人派遣战士，则所有玩家不得分。
- 否则，派遣了最多战士的玩家可以获得其分数，若有多个玩家派遣了同样多战士且都是最多的，那么这些玩家都能获得其分数。
- 所有玩家收回派遣在这块城堡/这段道路上的战士。

现在依次给出每张卡片的描述和放置位置，你需要判定每张卡片的放置和战士的派遣是否符合上面的规则，如果一张卡片放置不符合规则，则忽略这张卡片和这张卡片上派遣的战士；如果卡片放置符合规则但战士派遣不符合规则，则只忽略派遣的战士。

你需要统计每张卡片被放下后每名玩家的当前分数。

# 输入格式

从标准输入读入数据。

输入的第一行包含三个正整数 $n, m, k$，分别表示方块数、玩家数和每个玩家所拥有的战士数量。

接下来 $n$ 段，依次描述每张卡片，第 $i$ 张卡片的描述包含 $c_i + 1$ 行：

第一行包含三个整数 $x_i, y_i, c_i$，表示这张卡片放置的位置以及这张卡片上城堡的块数与道路的段数总和。

接下来 $c_i$ 行：每行首先是一个字母，`C` 表示城堡而 `R` 表示道路，然后一个仅包含 `U,D,L,R` 四个字母的字符串描述一块城堡或一段道路，最后一个整数 $x$。若 $x = 0$ 则无人在这块城堡/这段道路上派遣战士，若 $x > 0$ 则表示编号为 $x$ 的玩家在这块城堡/这段道路上派遣了一个战士。

# 输出格式

输出到标准输出。

输出 $n$ 行：每行 $m + 2$ 个数 $f_1, f_2, s_1, \ldots, s_m$，其中 $f_1$ 表示第 $i$ 张卡片的放置是否符合规则（$1$ 表示符合，$0$ 表示不符合）；$f_2$ 表示第 $i$ 张卡片上派遣的战士是否符合规则（如果 $f_1 = 0$ 则约定 $f_2$ 一定为 $0$；否则若没派遣任何战士则约定 $f_2 = 1$）；$s_i$ 表示放置完（或由于不合规则而忽略）第 $i$ 张卡片后编号为 $i$ 的玩家的分数。

# 样例1输入

```
7 3 10
0 0 2
C L 1
C U 0
-1 0 1
R UR 3
-1 0 1
C UR 3
0 1 1
C LD 2
0 1 1
C LD 2
-1 1 1
C DR 0
1000 1000 0
```

# 样例1输出

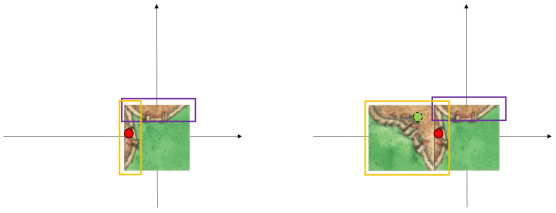
```
1 1 0 0 0
0 0 0 0 0
1 0 0 0 0
1 1 0 0 0
0 0 0 0 0
1 1 8 8 0
0 0 8 8 0
```

# 样例1解释

第一张卡片成功放置，并且玩家 1 在左边界的城堡上派遣了一个战士，如左下图。

第二张卡片不能成功放置，因为它右边界没有城堡，但是与它右侧相邻的第一张卡片的左边界有城堡。

第三张卡片成功放置，但玩家 3 尝试在城堡上派遣战士并没有成功，因为此时这张卡片上的城堡与第一张卡片左边界的城堡已经连接，而这块城堡上已经有玩家 1 派遣的战士了，如右下图（虚线表示没有成功派遣的战士）。

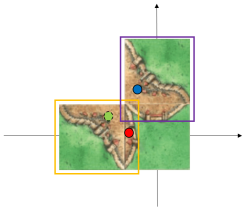
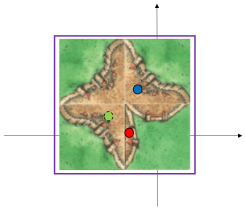


第四张卡片成功放置，并且玩家 2 在第四张卡片的城堡上派遣了一个战士，如左下图。

第五张卡片不能成功放置，因为 $(0, 1)$ 已经放置过了第四张卡片。

第六张卡片成功放置，此时一块城堡闭合，这块城堡上有玩家 1 和玩家 2 派遣的各一个战士，因此玩家 1 和玩家 2 各获得这块城堡的 $8$ 分，如右下图。

第七张卡片不能成功放置，因为没有与它相邻的卡片在此前已经放置。

上面的图片中，用不同颜色的矩形框起来的区域表示当前不同的城堡块。

# 样例2输入

```
6 3 1
0 0 2
R DR 1
C U 0
0 1 2
C D 3
C R 0
1 1 2
C L 0
R DR 2
2 1 4
R L 0
R R 0
R U 0
R D 3
0 -1 1
R U 2
1 0 2
R LU 0
C R 1
```

# 样例2输出

```
1 1 0 0 0
1 1 0 0 4
1 1 0 0 4
1 1 0 0 4
1 0 0 0 4
1 0 5 5 4
```

# 样例2解释

下图展示了最终版图。注意第六张卡片上 1 号玩家未能成功派遣战士是因为：派遣发生在结算之前，此时他的唯一一个战士仍然在道路上（尽管这张卡片的结算后他可以收回这个战士），故不能派遣到城堡上。

# 子任务

对于所有测试数据，$1 \leq n \leq 10^5$，$1 \leq m \leq 10$，$1 \leq k \leq 10^5$，$|x_i|, |y_i| \leq 10^5$，并有如下保证：

- 第一张卡片会被放置在 $(0, 0)$ 的位置。

- 每张卡片上（所有玩家总计）至多派遣了一个战士。
- 每张卡片上城堡/道路的布置都是合理的，具体地：
- 不会有不同的城堡/道路占据同一个边界；
- 不会有两个城堡/道路分别占据左右边界和上下边界。

通过每个测试点可以获得 $4$ 分：

- 对于测试点 $1 \sim 5$，保证 $n \leq 20$, $m = 1$, $k = 100$。
- 对于测试点 $6 \sim 10$，保证 $n \leq 100$, $m = 1$。
- 对于测试点 $11 \sim 15$，保证 $n \leq 1000$, $m = 2$, $k = 10^5$。
- 对于测试点 $16 \sim 20$，保证 $n \leq 1000$。
- 对于测试点 $21 \sim 25$，无特殊限制。

语言和编译选项

| # | 名称 | 编译器 | 额外参数 | 代码长度限制 |
|---|---|---|---|---|
| 0 | g++ | `g++` | `-DONLINE_JUDGE` | 65536 B |
| 1 | gcc | `gcc` | `-DONLINE_JUDGE` | 65536 B |
| 2 | java | `javac` | | 65536 B |
| 3 | python2 | `python` | | 65536 B |
| 4 | python3 | `python3` | | 65536 B |

递交历史

| # | 状态 | 时间 |
|---|---|---|
| 表中没有数据 | | |

# 花店

**时间限制：** 1.0 秒

**空间限制：** 512 MiB

刷新

**相关文件：** 下发文件 (/staticdata/223.7ExgyKNTElPNMwgO.pub/SFfIcQWUA8dJcpmq.down.zip/down.zip)

## 题目描述

小 I 经营着一个花店，其中 $n$ 盆最漂亮的花是非卖品，用于花店的装饰。方便起见，将它们从 $1$ 至 $n$ 编号。第 $i$ 盆花的魅力为 $b_i$。

为了迎客，每天小 I 都需要在 $n$ 盆花中选择一盆花摆在店门口。受花期、天气、花的种类的多样性等影响，小 I 已经决定了接下来 $m$ 天里每天摆哪一盆花，其中第 $i$ 天摆第 $a_i$ 盆花。

根据历年来的数据，小 I 预测接下来 $m-k+1$ 天里，第 $j(1 \leq j \leq m-k+1)$ 天将会有 $c_j$ 位顾客开始关注小 I 的花店。小 I 的花店可以打动某一位顾客，当且仅当：

- 若这个顾客在第 $x$ 天开始关注小 I 的花店，则在 $x,(x+1),\cdots,(x+k-1)$ 这连续的 $k$ 天里，小 I 花店每天摆出的花的魅力都大于等于 $V$，其中 $V$ 是一个对于所有顾客均相同的给定值。

每有一位顾客被打动，小 I 就会获得 $1$ 的收益。

为了获得更多的收益，小 I 决定今天培养一下这 $n$ 盆非卖品。具体地，小 I 可以花费 $x$ 的代价，将任意一盆花的魅力增加 $x$，其中 $x$ 为非负整数。小 I 可以使用这一方法对任意多的花增加魅力。

小 I 想知道，在如上情境下，$m$ 天后他的收益总和减去代价总和最大可以是多少。可是小 I 是花店老板不懂算法，所以需要你来帮忙。

## 输入格式

从标准输入读入数据。

输入的第一行包含四个非负整数 $n,m,k,V$，分别表示花的盆数、天数、打动顾客需要的连续天数、魅力值限制。

输入的第二行包含 $n$ 个非负整数 $b_1,b_2,\cdots,b_n$，表示每盆花的魅力值。

输入的第三行包含 $m$ 个正整数 $a_1,a_2,\cdots,a_m$，表示每天摆花的方案。

输入的第四行包含 $m-k+1$ 个非负整数 $c_1,c_2,\cdots,c_{m-k+1}$，表示每天开始关注花店的顾客个数。

## 输出格式

输出到标准输出。

输出一行一个整数，表示收益总和减去代价总和的最大值。

# 样例1输入

```
2 3 2 10
9 10
1 1 2
2 3
```

# 样例1输出

```
4
```

# 样例1解释

将第一盆花的魅力值增加 $1$，这样所有的 $5$ 个顾客都会被打动，收益为 $5$，代价为 $1$。容易证明没有更优的方案，因此输出为 $5-1=4$。

# 样例2输入

```
2 3 2 10
10 0
1 1 2
2 3
```

# 样例2输出

```
2
```

# 样例2解释

不做任何操作，这样第一天开始关注花店的两个顾客会被打动，而第二天开始关注花店的三个顾客不会。收益为 $2$，代价为 $0$。容易证明没有更优的方案，因此输出为 $2-0=2$。

# 子任务

只有通过一个子任务内所有的数据点才能获得该子任务的分数。

对于所有测试数据，$1 \leq n \leq 5000$，$1 \leq k \leq m \leq 5000$，$0 \leq b_i \leq V \leq 10^6$，$1 \leq a_i \leq n$，$0 \leq c_i \leq 10^6$，$\sum_{i=1}^{n}(V-b_i) \leq 10^6$，$\sum_{i=1}^{m-k+1} c_i \leq 10^6$。

| 子任务编号 | $n \leq$ | $m \leq$ | 特殊性质 | 分值 |
|---|---|---|---|---|
| 1 | 5,000 | 5,000 | A | 10 |
| 2 | | | B | |
| 3 | | | C | 15 |
| 4 | 12 | | 无 | |
| 5 | 300 | 300 | | 25 |
| 6 | 5,000 | 5,000 | | |

特殊性质 A：$k = m$。

特殊性质 B：$k = 1$。

特殊性质 C：$n \geq m,\ \forall 1 \leq i \leq m, a_i = i$。

语言和编译选项

| # | 名称 | 编译器 | 额外参数 | 代码长度限制 |
|---|---|---|---|---|
| 0 | g++ | `g++` | `-DONLINE_JUDGE` | 65536 B |
| 1 | gcc | `gcc` | `-DONLINE_JUDGE` | 65536 B |
| 2 | java | `javac` | | 65536 B |
| 3 | python2 | `python` | | 65536 B |
| 4 | python3 | `python3` | | 65536 B |

递交历史

| # | 状态 | 时间 |
|---|---|---|
| 68811 | Wrong Answer | 2023-06-30 12:07:01 有效递交 |
| 68776 | Wrong Answer | 2023-06-30 12:05:05 |
| 68535 | Wrong Answer | 2023-06-30 11:42:32 |
| 68531 | Wrong Answer | 2023-06-30 11:42:05 |
| 68520 | Wrong Answer | 2023-06-30 11:40:58 |
| 68489 | Wrong Answer | 2023-06-30 11:38:35 |
| 68433 | Wrong Answer | 2023-06-30 11:32:08 |

| # | 状态 | 时间 |
| --- | --- | --- |
| 68380 | Wrong Answer | 2023-06-30 11:26:54 |
| 68362 | Wrong Answer | 2023-06-30 11:25:16 |
| 68282 | Wrong Answer | 2023-06-30 11:16:42 |