2017 年全国青少年信息学奥林匹克竞赛

山西省代表队选拔　第二试

比赛时间：2017 年 4 月 23 日 08:00 ~ 12:30

题目名称	摧毁“树状图”	分手是祝愿	寿司餐厅
题目类型	传统型	传统型	传统型
目录	treediagram	trennen	sushi
可执行文件名	treediagram	trennen	sushi
输入文件名	treediagram.in	trennen.in	sushi.in
输出文件名	treediagram.out	trennen.out	sushi.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒
内存限制	512 MB	512 MB	512 MB
测试点数目	25	20	20
每个测试点分值	4	5	5

提交源程序文件名

对于 C++ 语言	treediagram.cpp	trennen.cpp	sushi.cpp
对于 C 语言	treediagram.c	trennen.c	sushi.c
对于 Pascal 语言	treediagram.pas	trennen.pas	sushi.pas

编译选项

对于 C++ 语言	-O2 -lm	-O2 -lm	-lm
对于 C 语言	-O2 -lm	-O2 -lm	-lm
对于 Pascal 语言	-O2	-O2	

注意事项：

1. 文件名（文件夹名、程序名和输入输出文件名）必须使用英文小写，不得带有空白符。
2. 源代码应当放置在选手目录下各题目的目录下，必须为每道题目单独建立目录。
3. 除非特殊说明，结果比较方式均为忽略行末空格及文末回车的全文比较。
4. C/C++ 函数 main() 的返回值类型必须是 int，程序正常结束时的返回值是 0。
5. 文件读写结束后必须关闭文件。
6. 评测在 NOI Linux 系统下进行，测试数据使用 Linux 换行符 `\n`。

摧毁“树状图”（treediagram）

【题目描述】

自从上次神刀手帮助蚯蚓国增添了上千万人口（蚯口？），蚯蚓国发展得越来越繁荣了！最近，他们在地下发现了一些神奇的纸张，经过仔细研究，居然是 D 国 X 市的超级计算机设计图纸！

这台计算机叫做“树状图”，由 n 个计算节点与 $n-1$ 条可以双向通信的网线连接而成，所有计算节点用不超过 n 的正整数编号。顾名思义，这形成了一棵树的结构。

蚯蚓国王已在图纸上掌握了这棵树的完整信息，包括 n 的值与 $n-1$ 条网线的连接信息。于是蚯蚓国王决定，派出蚯蚓国最强大的两个黑客，小 P 和小 H，入侵“树状图”，尽可能地摧毁它。

小 P 和小 H 精通世界上最好的编程语言，经过一番商量后，他们决定依次采取如下的步骤：

- 小 P 选择某个计算节点，作为他入侵的起始点，并在该节点上添加一个 P 标记。
- 重复以下操作若干次（可以是 0 次）：
  - 小 P 从他当前所在的计算节点出发，选择一条**没有被标记过的**网线，入侵到该网线的另一端的计算节点，并在路过的网线与目的计算节点上均添加一个 P 标记。
- 小 H 选择某个计算节点，作为她入侵的起始点，并在该节点上添加一个 H 标记。
- 重复以下操作若干次（可以是 0 次）：
  - 小 H 从她当前所在的计算节点出发，选择一条**没有被标记过的**网线，入侵到该网线的另一端的计算节点，并在路过的网线与目的计算节点上均添加一个 H 标记。（注意，小 H 不能经过带有 P 标记的网线，但是可以经过带有 P 标记的计算节点）
- 删除所有被标记过的计算节点和网线。
- 对于剩下的每条网线，如果其一端或两端的计算节点在上一步被删除了，则也删除这条网线。

经过以上操作后，“树状图”会被断开，剩下若干个（可能是 0 个）连通块。为了达到摧毁的目的，蚯蚓国王希望，连通块的个数越多越好。于是他找到了你，希望你能帮他计算这个最多的个数。

小 P 和小 H 非常心急，在你计算方案之前，他们可能就已经算好了最优方案或最优方案的一部分。你能得到一个值 x：

- 若 $x=0$，则说明小 P 和小 H 没有算好最优方案，你需要确定他们两个的入侵路线。

- 若 $x=1$，则说明小 P 已经算好了某种两人合作的最优方案中，他的入侵路线。他将选择初始点 p_0，并沿着网线一路入侵到了目标点 p_1，并且他不会再沿着网线入侵；你只需要确定小 H 的入侵路线。
- 若 $x=2$，则说明小 P 和小 H 算好了一种两人合作的最优方案，小 P 从点 p_0 入侵到了 p_1 并停下，小 H 从点 h_0 入侵到了 h_1 并停下。此时你不需要指挥他们入侵了，只需要计算最后两步删除计算节点与网线后，剩下的连通块个数即可。

【输入格式】

从文件 ***treediagram.in*** 中读入数据。

每个输入文件包含多个输入数据。输入文件的第一行为两个整数 T 和 x，T 表示该文件包含的输入数据个数，x 的含义见上述。（同一个输入文件的所有数据的 x 都是相同的）

接下来依次输入每个数据。

每个数据的第一行有若干个整数：

- 若 $x=0$，则该行只有一个整数 n。
- 若 $x=1$，则该行依次有三个整数 n,p_0,p_1。
- 若 $x=2$，则该行依次有五个整数 n,p_0,p_1,h_0,h_1。

保证 p_0,p_1,h_0,h_1 均为不超过 n 的正整数。

每个数据接下来有 $n-1$ 行，每行有两个不超过 n 的正整数，表示这两个编号的计算节点之间有一条网线将其相连。保证输入的是一棵树。

同一行相邻的整数之间用恰好一个空格隔开。

数据文件可能较大，请避免使用过慢的输入输出方法。

【输出格式】

输出到文件 ***treediagram.out*** 中。

对于每个数据，输出一行，表示在给定条件下，剩下连通块的最大个数。

【样例 1 输入】

```
1 0
13
1 2
2 3
2 4
4 5
4 6
```

```
4 7
7 8
7 9
9 10
10 11
10 12
12 13
```

【样例 1 输出】

```
8
```

【样例 1 说明】

这个输入文件只有一个输入数据。一种最优的方案如下：

- 小 P 从节点 2 开始入侵，节点 2 被小 P 标记。
- 小 P 从节点 2 入侵到节点 4，节点 4 和经过的网线被小 P 标记。
- 小 P 从节点 4 入侵到节点 7，节点 7 和经过的网线被小 P 标记。
- 小 H 从节点 10 开始入侵，节点 10 被小 H 标记。
- 删除被标记的节点 2,4,7,10 和被标记的网线 (2,4) 和 (4,7)。
- 删除任意一端在上一步被删除的网线。

此时还剩下 8 个连通块。其中节点 1,3,5,6,8,9,11 各自形成一个连通块，节点 12,13 形成了一个连通块。

【样例 2】

见选手目录下的 *treediagram/treediagram2.in* 与 *treediagram/treediagram2.ans*。

【样例 2 说明】

数据 1：只有 1 个计算节点，唯一可行的方案是小 P 从节点 1 开始入侵（并马上停止），小 H 也从节点 1 入侵到节点 1。所有的节点都被删去，剩下 0 个连通块。

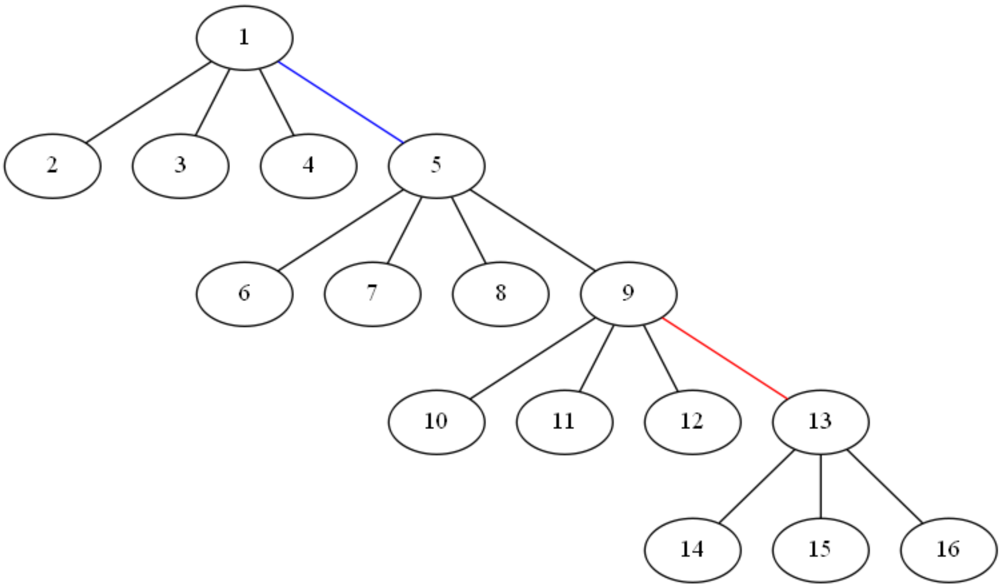
数据 2：一种最优方案是，小 P 从节点 1 入侵到节点 1，小 H 也从节点 1 入侵到节点 1。在删除操作后，剩下 1 个连通块（只有节点 2）。

数据 3：唯一的最优方案是，小 P 从节点 2 入侵到节点 2，小 H 也从节点 2 入侵到节点 2，剩下 2 个连通块。

数据 4：一种最优方案是，小 P 从节点 2 入侵到节点 2，小 H 也从节点 2 入侵到节点 2，剩下 2 个连通块。

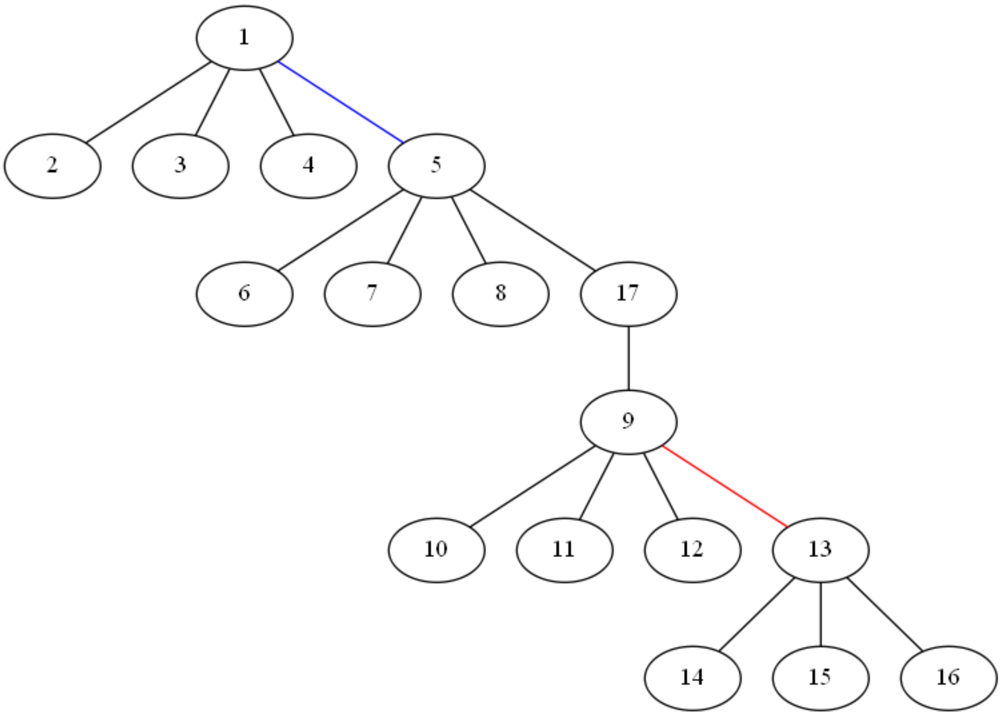
数据 5：唯一的最优方案是，小 P 从节点 5 入侵到节点 5，小 H 也从节点 5 入侵到节点 5，剩下 4 个连通块。

数据 6：



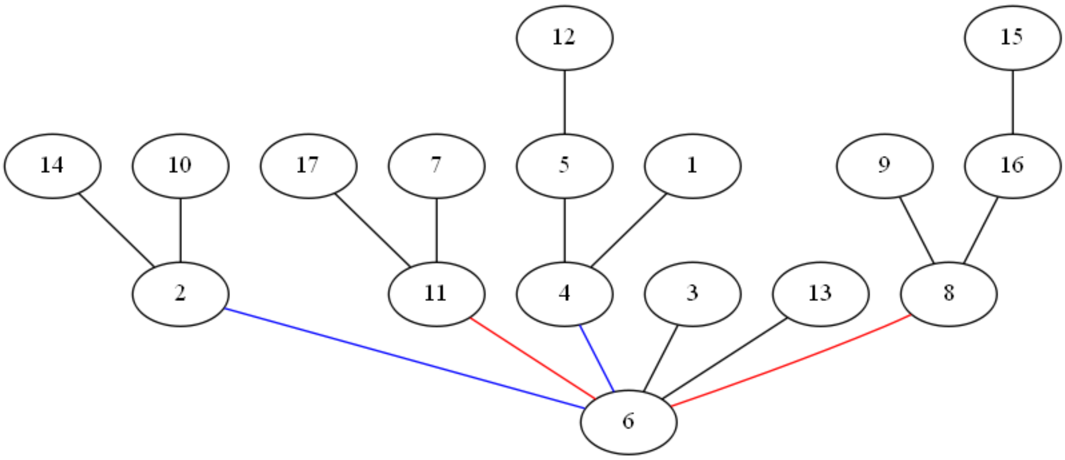
一种最优方案是，小 P 从节点 1 入侵到节点 5，小 H 从节点 9 入侵到节点 13，剩下 12 个连通块。

数据 7：



一种最优方案是，小 P 从节点 1 入侵到节点 5，小 H 从节点 9 入侵到节点 13，剩下 13 个连通块。

数据 8：



一种最优方案是，小 P 从节点 2 入侵到节点 4，小 H 从节点 8 入侵到节点 11，剩下 10 个连通块。

注意，这里节点 6 被小 P 和小 H 同时标记了，但是由于没有网线被同时标记，所以是合法的。

【样例 3】

见选手目录下的 *treediagram/treediagram3.in* 与 *treediagram/treediagram3.ans*。

【样例 3 说明】

这个样例与上个样例的唯一区别是这里 $x = 1$。输出结果应当是相同的。

【样例 4】

见选手目录下的 *treediagram/treediagram4.in* 与 *treediagram/treediagram4.ans*。

【样例 4 说明】

这个样例与上个样例的唯一区别是这里 $x = 2$。输出结果应当是相同的。

【样例 5~6】

见选手目录下的 *treediagram/treediagram5~6.in* 与 *treediagram/treediagram5~6.ans*。

【子任务】

对于整数 k，设 $\sum n^k$ 为某个输入文件中，其 T 个输入数据的 n^k 之和。

所有输入文件满足 $T \le 10^5, \sum n^1 \le 5 \times 10^5$。**请注意初始化的时间复杂度，避免输入大量小数据时超时。**

每个测试点的详细数据范围见下表。

如果表中“完全二叉”为 Yes，则该输入文件的每个数据满足：网线信息的第 j 行（$1 \le j < n$）输入的两个数依次是 $\lfloor \frac{j+1}{2} \rfloor$ 和 $j+1$。

<table>
<tr><th>测试点</th><th>x</th><th>n</th><th>$\sum n^k$</th><th>完全二叉</th><th>T</th></tr>
<tr><td>1</td><td rowspan="7">$=0$</td><td>$n \le 1$</td><td rowspan="4">$\sum n^0 \le 10^2$</td><td rowspan="7">No</td><td rowspan="4">$\le 10^2$</td></tr>
<tr><td>2</td><td>$n \le 2$</td></tr>
<tr><td>3</td><td>$n \le 3$</td></tr>
<tr><td>4</td><td>$n \le 4$</td></tr>
<tr><td>5</td><td>$n \le 5$</td><td rowspan="2">$\sum n^0 \le 10^3$</td><td rowspan="2">$\le 10^3$</td></tr>
<tr><td>6</td><td>$n \le 6$</td></tr>
<tr><td>7</td><td>$n \le 7$</td><td>$\sum n^0 \le 10^4$</td><td>$\le 10^4$</td></tr>
<tr><td>8</td><td>$=2$</td><td rowspan="6">$n \le 10^2$</td><td rowspan="6">$\sum n^3 \le 10^7$</td><td rowspan="3">Yes</td><td rowspan="18">$\le 10^5$</td></tr>
<tr><td>9</td><td>$=1$</td></tr>
<tr><td>10</td><td>$=0$</td></tr>
<tr><td>11</td><td>$=2$</td><td rowspan="3">No</td></tr>
<tr><td>12</td><td>$=1$</td></tr>
<tr><td>13</td><td>$=0$</td></tr>
<tr><td>14</td><td>$=2$</td><td rowspan="6">$n \le 10^3$</td><td rowspan="6">$\sum n^2 \le 10^7$</td><td rowspan="3">Yes</td></tr>
<tr><td>15</td><td>$=1$</td></tr>
<tr><td>16</td><td>$=0$</td></tr>
<tr><td>17</td><td>$=2$</td><td rowspan="3">No</td></tr>
<tr><td>18</td><td>$=1$</td></tr>
<tr><td>19</td><td>$=0$</td></tr>
<tr><td>20</td><td>$=2$</td><td rowspan="6">$n \le 10^5$</td><td rowspan="6">$\sum n^1 \le 5 \times 10^5$</td><td rowspan="3">Yes</td></tr>
<tr><td>21</td><td>$=1$</td></tr>
<tr><td>22</td><td>$=0$</td></tr>
<tr><td>23</td><td>$=2$</td><td rowspan="3">No</td></tr>
<tr><td>24</td><td>$=1$</td></tr>
<tr><td>25</td><td>$=0$</td></tr>
</table>

分手是祝愿（trennen）

【问题描述】

Zeit und Raum trennen dich und mich.

时空将你我分开。

B 君在玩一个游戏，这个游戏由 n 个灯和 n 个开关组成，给定这 n 个灯的初始状态，下标为从 1 到 n 的正整数。

每个灯有两个状态亮和灭，我们用 1 来表示这个灯是亮的，用 0 表示这个灯是灭的，游戏的目标是使所有灯都灭掉。

但是当操作第 i 个开关时，所有编号为 i 的约数（包括 1 和 i）的灯的状态都会被改变，即从亮变成灭，或者是从灭变成亮。

B 君发现这个游戏很难，于是想到了这样的一个策略，每次等概率随机操作一个开关，直到所有灯都灭掉。

这个策略需要的操作次数很多，B 君想到这样的一个优化。如果当前局面，可以通过操作小于等于 k 个开关使所有灯都灭掉，那么他将不再随机，直接选择操作次数最小的操作方法（这个策略显然小于等于 k 步）操作这些开关。

B 君想知道按照这个策略（也就是先随机操作，最后小于等于 k 步，使用操作次数最小的操作方法）的操作次数的期望。

这个期望可能很大，但是 B 君发现这个期望乘以 n 的阶乘一定是整数，所以他只需要知道这个整数对 100003 取模之后的结果。

【输入格式】

从文件 ***trennen.in*** 中读入数据。

第一行两个整数 n, k。

接下来一行 n 个整数，每个整数是 0 或者 1，其中第 i 个整数表示第 i 个灯的初始情况。

【输出格式】

输出到文件 ***trennen.out*** 中。

输出一行，为操作次数的期望乘以 n 的阶乘对 100003 取模之后的结果。

【样例 1 输入】

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

【样例 1 输出】

```
512
```

【样例 2 输入】

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

【样例 2 输出】

```
5120
```

【样例 3~8】

见选手目录下的 ***trennen/trennen3~8.in*** 与 ***trennen/trennen3~8.ans***。

【子任务】

- 对于 0% 的测试点，和样例一模一样；
- 对于另外 30% 的测试点，$n \leq 10$；
- 对于另外 20% 的测试点，$n \leq 100$；
- 对于另外 30% 的测试点，$n \leq 1000$；
- 对于 100% 的测试点，$1 \leq n \leq 100000, 0 \leq k \leq n$；
- 对于以上每部分测试点，均有一半的数据满足 $k = n$。

寿司餐厅（sushi）

【问题描述】

Kiana 最近喜欢到一家非常美味的寿司餐厅用餐。

每天晚上，这家餐厅都会按顺序提供 n 种寿司，第 i 种寿司有一个代号 a_i 和美味度 $d_{i,i}$，不同种类的寿司有可能使用相同的代号。每种寿司的份数都是无限的，Kiana 也可以无限次取寿司来吃，但每种寿司每次只能取一份，且每次取走的寿司必须是按餐厅提供寿司的顺序连续的一段，即 Kiana 可以一次取走第 1,2 种寿司各一份，也可以一次取走第 2,3 种寿司各一份，但不可以一次取走第 1,3 种寿司。

由于餐厅提供的寿司种类繁多，而不同种类的寿司之间相互会有影响：三文鱼寿司和鱿鱼寿司一起吃或许会很棒，但和水果寿司一起吃就可能会肚子痛。因此，Kiana 定义了一个综合美味度 $d_{i,j}(i < j)$，表示在一次取的寿司中，如果包含了餐厅提供的从第 i 份到第 j 份的所有寿司，吃掉这次取的所有寿司后将获得的额外美味度。由于取寿司需要花费一些时间，所以我们认为分两次取来的寿司之间相互不会影响。注意在吃一次取的寿司时，不止一个综合美味度会被累加，比如若 Kiana 一次取走了第 1,2,3 种寿司各一份，除了 $d_{1,3}$ 以外，$d_{1,2}, d_{2,3}$ 也会被累加进总美味度中。

神奇的是，Kiana 的美食评判标准是有记忆性的，无论是单种寿司的美味度，还是多种寿司组合起来的综合美味度，在计入 Kiana 的总美味度时都只会被累加一次。比如，若 Kiana 某一次取走了第 1,2 种寿司各一份，另一次取走了第 2,3 种寿司各一份，那么这两次取寿司的总美味度为 $d_{1,1} + d_{2,2} + d_{3,3} + d_{1,2} + d_{2,3}$，其中 $d_{2,2}$ 只会计算一次。

奇怪的是，这家寿司餐厅的收费标准很不同寻常。具体来说，如果 Kiana 一共吃过了 c 种代号为 x 的寿司，则她需要为这些寿司付出 $mx^2 + cx$ 元钱，其中 m 是餐厅给出的一个常数。

现在 Kiana 想知道，在这家餐厅吃寿司，自己能获得的总美味度（包括所有吃掉的单种寿司的美味度和所有被累加的综合美味度）减去花费的总钱数的最大值是多少。由于她不会算，所以希望由你告诉她。

【输入格式】

从文件 ***sushi.in*** 中读入数据。

第一行包含两个正整数 n, m，分别表示这家餐厅提供的寿司总数和计算寿司价格中使用的常数。

第二行包含 n 个正整数，其中第 k 个数 a_k 表示第 k 份寿司的代号。

接下来 n 行，第 i 行包含 $n - i + 1$ 个整数，其中第 j 个数 $d_{i,i+j-1}$ 表示吃掉寿司能获得的相应的美味度，具体含义见问题描述。

【输出格式】

输出到文件 ***sushi.out*** 中。

输出共一行包含一个正整数，表示 Kiana 能获得的总美味度减去花费的总钱数的最大值。

【样例 1 输入】

```
3 1
2 3 2
5 -10 15
-10 15
15
```

【样例 1 输出】

```
12
```

【样例 1 说明】

在这组样例中，餐厅一共提供了 3 份寿司，它们的代号依次为 $a_1 = 2$，$a_2 = 3$，$a_3 = 2$，计算价格时的常数 $m = 1$。在保证每次取寿司都能获得新的美味度的前提下，Kiana 一共有 14 种不同的吃寿司方案：

1.Kiana 一个寿司也不吃，这样她获得的总美味度和花费的总钱数都是 0，两者相减也是 0；

2.Kiana 只取 1 次寿司，且只取第 1 个寿司，即她取寿司的情况为 $\{[1,1]\}$，这样获得的总美味度为 5，花费的总钱数为 $1*2^2+1*2=6$，两者相减为 -1；

3.Kiana 只取 1 次寿司，且只取第 2 个寿司，即她取寿司的情况为 $\{[2,2]\}$，这样获得的总美味度为 -10，花费的总钱数为 $1*3^2+1*3=12$，两者相减为 -22；

4.Kiana 只取 1 次寿司，且只取第 3 个寿司，即她取寿司的情况为 $\{[3,3]\}$，这样获得的总美味度为 15，花费的总钱数为 $1*2^2+1*2=6$，两者相减为 9；

5.Kiana 只取 1 次寿司，且取第 1,2 个寿司，即她取寿司的情况为 $\{[1,2]\}$，这样获得的总美味度为 $5+(-10)+(-10)=-15$，花费的总钱数为 $(1*2^2+1*2)+(1*3^2+1*3)=18$，两者相减为 -33；

6.Kiana 只取 1 次寿司，且取第 2,3 个寿司，即她取寿司的情况为 $\{[2,3]\}$，这样获得的总美味度为 $(-10)+15+15=20$，花费的总钱数为 $(1*2^2+1*2)+(1*3^2+1*3)=18$，两者相减为 2；

7.Kiana 只取 1 次寿司，且取第 1,2,3 个寿司，即她取寿司的情况为 $\{[1,3]\}$，这样获得的总美味度为 $5+(-10)+15+(-10)+15+15=30$，花费的总钱数为

$(1*2^2+2*2)+(1*3^2+1*3)=20$，两者相减为 10。

8.Kiana 取 2 次寿司，第一次取第 1 个寿司，第二次取第 2 个寿司，即她取寿司的情况为 $\{[1,1],[2,2]\}$，这样获得的总美味度为 $5+(-10)=-5$，花费的总钱数为 $(1*2^2+1*2)+(1*3^2+1*3)=18$，两者相减为 −23；

9.Kiana 取 2 次寿司，第一次取第 1 个寿司，第二次取第 3 个寿司，即她取寿司的情况为 $\{[1,1],[3,3]\}$，这样获得的总美味度为 $5+15=20$，花费的总钱数为 $1*2^2+2*2=8$，两者相减为 12；

10.Kiana 取 2 次寿司，第一次取第 2 个寿司，第二次取第 3 个寿司，即她取寿司的情况为 $\{[2,2],[3,3]\}$，这样获得的总美味度为 $(-10)+15=5$，花费的总钱数为 $(1*2^2+1*2)+(1*3^2+1*3)=18$，两者相减为 −13；

11.Kiana 取 2 次寿司，第一次取第 1,2 个寿司，第二次取第 2 个寿司，即她取寿司的情况为 $\{[1,2],[3,3]\}$，这样获得的总美味度为 $5+(-10)+(-10)+15=0$，花费的总钱数为 $(1*2^2+2*2)+(1*3^2+1*3)=20$，两者相减为 −20；

12.Kiana 取 2 次寿司，第一次取第 1 个寿司，第二次取第 2,3 个寿司，即她取寿司的情况为 $\{[1,1],[2,3]\}$，这样获得的总美味度为 $5+(-10)+15+15=25$，花费的总钱数为 $(1*2^2+2*2)+(1*3^2+1*3)=20$，两者相减为 5；

13.Kiana 取 2 次寿司，第一次取第 1,2 个寿司，第二次取第 2,3 个寿司，即她取寿司的情况为 $\{[1,2],[2,3]\}$，这样获得的总美味度为 $5+(-10)+15+(-10)+15=15$，花费的总钱数为 $(1*2^2+2*2)+(1*3^2+1*3)=20$，两者相减为 −5；

14.Kiana 取 3 次寿司，第一次取第 1 个寿司，第二次取第 2 个寿司，第三次取第 3 个寿司，即她取寿司的情况为 $\{[1,1],[2,2],[3,3]\}$，这样获得的总美味度为 $5+(-10)+15=10$，花费的总钱数为 $(1*2^2+2*2)+(1*3^2+1*3)=20$，两者相减为 −10。

所以 Kiana 会选择方案 9，这时她获得的总美味度减去花费的总钱数的值最大为 12。

【样例 2 输入】

```
5 0
1 4 1 3 4
50 99 8 -39 30
68 27 -75 -32
70 24 72
-10 81
-95
```

【样例 2 输出】

```
381
```

【样例 3 输入】

```
10 1
5 5 4 4 1 2 5 1 5 3
83 91 72 29 22 -5 57 -14 -36 -3
-11 34 45 96 32 73 -1 0 29
-48 68 44 -5 96 66 17 74
88 47 69 -9 2 25 -49
86 -9 -77 62 -10 -30
2 40 95 -74 46
49 -52 2 -51
-55 50 -44
72 22
-68
```

【样例 3 输出】

```
1223
```

【子任务】

对于所有数据，保证 $-500 \le d_{i,j} \le 500$。

数据的一些特殊约定如下表：

<table>
<tr><th>测试点编号</th><th>n</th><th>a_i</th><th>m</th><th>备注</th></tr>
<tr><td>1</td><td rowspan="2">≤ 2</td><td rowspan="10">≤ 30</td><td>$= 0$</td><td rowspan="6">无</td></tr>
<tr><td>2</td><td>$= 1$</td></tr>
<tr><td>3</td><td rowspan="2">≤ 3</td><td>$= 0$</td></tr>
<tr><td>4</td><td>$= 1$</td></tr>
<tr><td>5</td><td rowspan="2">≤ 5</td><td>$= 0$</td></tr>
<tr><td>6</td><td>$= 1$</td></tr>
<tr><td>7</td><td rowspan="2">≤ 10</td><td>$= 0$</td><td>所有的 a_i 相同</td></tr>
<tr><td>8</td><td>$= 1$</td><td>无</td></tr>
<tr><td>9</td><td rowspan="2">≤ 15</td><td>$= 0$</td><td>所有的 a_i 相同</td></tr>
<tr><td>10</td><td>$= 1$</td><td>无</td></tr>
<tr><td>11</td><td rowspan="4">≤ 30</td><td>≤ 1000</td><td>$= 0$</td><td rowspan="2">所有的 a_i 相同</td></tr>
<tr><td>12</td><td>≤ 30</td><td>$= 0$</td></tr>
<tr><td>13</td><td rowspan="3">≤ 1000</td><td>$= 0$</td><td rowspan="2">无</td></tr>
<tr><td>14</td><td>$= 1$</td></tr>
<tr><td>15</td><td rowspan="4">≤ 50</td><td>$= 0$</td><td rowspan="2">所有的 a_i 相同</td></tr>
<tr><td>16</td><td>≤ 30</td><td>$= 0$</td></tr>
<tr><td>17</td><td rowspan="4">≤ 1000</td><td>$= 0$</td><td rowspan="2">无</td></tr>
<tr><td>18</td><td>$= 1$</td></tr>
<tr><td>19</td><td rowspan="2">≤ 100</td><td>$= 0$</td><td rowspan="2">无</td></tr>
<tr><td>20</td><td>$= 1$</td></tr>
</table>