第22届全国青少年信息学奥林匹克联赛

CCF-NOIP-2016

提高组（复赛） 第一试

竞赛时间：2016年11月19日 8:30 ~ 12:00

题目名称	玩具谜题	天天爱跑步	换教室
题目类型	传统型	传统型	传统型
目录	toy	running	classroom
可执行文件名	toy	running	classroom
输入文件名	toy.in	running.in	classroom.in
输出文件名	toy.out	running.out	classroom.out
每个测试点时限	1.0秒	2.0秒	1.0秒
内存限制	512 MB	512 MB	512 MB
测试点数目	20	20	25
每个测试点分值	5	5	4

提交源程序文件名

对于C++ 语言	toy.cpp	running.cpp	classroom.cpp
对于C 语言	toy.c	running.c	classroom.c
对于Pascal 语言	toy.pas	running.pas	classroom.pas

编译选项

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

注意事项：

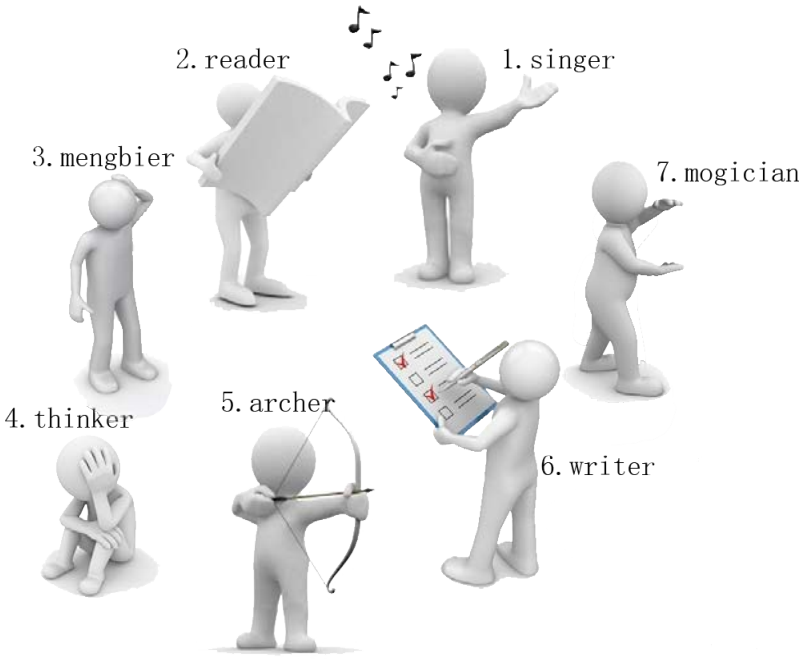
1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. 除非特殊说明，结果比较方式均为忽略行末空格及文末回车的全文比较。
3. C/C++中函数main()的返回值类型必须是int，程序正常结束时的返回值必须是0。
4. 全国统一评测时采用的机器配置为：CPU AMD Athlon(tm) II x2 240 processor，2.8GHz，内存4G，上述时限以此配置为准。
5. 只提供Linux格式附加样例文件。
6. 评测在NOI Linux下进行。
7. 编译时不打开任何优化选项。

玩具谜题（toy）

【问题描述】

小南有一套可爱的玩具小人，它们各有不同的职业。

有一天，这些玩具小人把小南的眼镜藏了起来。小南发现玩具小人们围成了一个圈，它们有的面朝圈内，有的面朝圈外。如下图：



这时singer 告诉小南一个谜题：“眼镜藏在我左数第 3 个玩具小人的右数第 1 个玩具小人的左数第 2 个玩具小人那里。”

小南发现，这个谜题中玩具小人的朝向非常关键，因为朝内和朝外的玩具小人的左右方向是相反的：面朝圈内的玩具小人，它的左边是顺时针方向，右边是逆时针方向；而面向圈外的玩具小人，它的左边是逆时针方向，右边是顺时针方向。

小南一边艰难地辨认着玩具小人，一边数着：

“singer 朝内，左数第 3 个是archer。

“archer 朝外，右数第 1 个是thinker。

“thinker 朝外，左数第 2 个是writer。

“所以眼镜藏在writer 这里！”

虽然成功找回了眼镜，但小南并没有放心。如果下次有更多的玩具小人藏他的眼镜，或是谜题的长度更长，他可能就无法找到眼镜了。所以小南希望你写程序帮他解决类似的谜题。这样的谜题具体可以描述为：

有 n 个玩具小人围成一圈，已知它们的职业和朝向。现在第 1 个玩具小人告诉小南一个包含 m 条指令的谜题，其中第 i 条指令形如“左数/右数第 s_i 个玩具小人”。你需要输出依次数完这些指令后，到达的玩具小人的职业。

【输入格式】

从文件***toy.in*** 中读入数据。

输入的第一行包含两个正整数 n, m ，表示玩具小人的个数和指令的条数。

接下来 n 行，每行包含一个整数和一个字符串，以逆时针为顺序给出每个玩具小人的朝向和职业。其中 0 表示朝向圈内， 1 表示朝向圈外。保证不会出现其他的数。字符串长度不超过 10 且仅由小写字母构成，字符串不为空，并且字符串两两不同。整数和字符串之间用一个空格隔开。

接下来 m 行，其中第 i 行包含两个整数 a_i, s_i ，表示第 i 条指令。若 $a_i = 0$ ，表示向左数 s_i 个人；若 $a_i = 1$ ，表示向右数 s_i 个人。保证 a_i 不会出现其他的数， $1 \le s_i < n$ 。

【输出格式】

输出到文件***toy.out*** 中。

输出一个字符串，表示从第一个读入的小人开始，依次数完 m 条指令后到达的小人的职业。

【样例1输入】

```
7 3
0 singer
0 reader
0 mengbier
1 thinker
1 archer
0 writer
1 mogician
0 3
1 1
0 2
```

【样例1输出】

```
writer
```

【样例1说明】

这组数据就是【题目描述】中提到的例子。

【样例2输入】

```
10 10
1 c
0 r
0 p
1 d
1 e
1 m
1 t
1 y
1 u
0 v
1 7
1 1
1 4
0 5
0 3
0 1
1 6
1 2
0 8
0 4
```

【样例2输出】

```
y
```

【子任务】

子任务会给出部分测试数据的特点。如果你在解决题目中遇到了困难，可以尝试只解决一部分测试数据。

每个测试点的数据规模及特点如下表：

测试点	n	m	全朝内	全左数	$s_i = 1$	职业长度为1
1	$= 20$	$= 10^3$	√	√	√	√
2			×			
3			√	×		
4			×			
5			√	√	×	
6			×			
7			√	×		
8			×			
9			√	√	√	×
10			×			
11			√	×		
12			×			
13			√	√	×	
14			×			
15			√	×		
16			×			
17	$= 10^5$	$= 10^5$	√	√		
18			×			
19			√	×		
20			×			

其中一些简写的列意义如下：

- 全朝内：若为“√”，表示该测试点保证所有的玩具小人都朝向圈内；
- 全左数：若为“√”，表示该测试点保证所有的指令都向左数，即对任意的 $1 \le i \le m$，$a_i = 0$；
- $s_i = 1$：若为“√”，表示该测试点保证所有的指令都只数 1 个，即对任意的 $1 \le i \le m$，$s_i = 1$；
- 职业长度为 1：若为“√”，表示该测试点保证所有玩具小人的职业一定是一个长度为 1 的字符串。

天天爱跑步（running）

【问题描述】

小C同学认为跑步非常有趣，于是决定制作一款叫做《天天爱跑步》的游戏。《天天爱跑步》是一个养成类游戏，需要玩家每天按时上线，完成打卡任务。

这个游戏的地图可以看作一棵包含 n 个结点和 $n-1$ 条边的树，每条边连接两个结点，且任意两个结点存在一条路径互相可达。树上结点编号为从 1 到 n 的连续正整数。

现在有 m 个玩家，第 i 个玩家的起点为 S_i，终点为 T_i。每天打卡任务开始时，所有玩家**在第 0 秒**同时从**自己的起点**出发，以**每秒跑一条边**的速度，不间断地沿着最短路径向着**自己的终点**跑去，跑到终点后该玩家就算完成了打卡任务。（由于地图是一棵树，所以每个人的路径是唯一的）

小C想知道游戏的活跃度，所以在每个结点上都放置了一个观察员。在结点 j 的观察员会选择在第 W_j 秒观察玩家，一个玩家能被这个观察员观察到当且仅当该玩家在第 W_j 秒也**正好**到达了结点 j。小C想知道每个观察员会观察到多少人？

注意：我们认为一个玩家到达自己的终点后该玩家就会结束游戏，他不能等待一段时间后再被观察员观察到。即对于把结点 j 作为终点的玩家：若他在第 W_j 秒**前**到达终点，则在结点 j 的观察员**不能观察到**该玩家；若他**正好**在第 W_j 秒到达终点，则在结点 j 的观察员**可以观察到**这个玩家。

【输入格式】

从文件***running.in*** 中读入数据。

第一行有两个整数 n 和 m。其中 n 代表树的结点数量，同时也是观察员的数量，m 代表玩家的数量。

接下来 $n-1$ 行每行两个整数 u 和 v，表示结点 u 到结点 v 有一条边。

接下来一行 n 个整数，其中第 j 个整数为 W_j，表示结点 j 出现观察员的时间。

接下来 m 行，每行两个整数 S_i 和 T_i，表示一个玩家的起点和终点。

对于所有的数据，保证 $1 \le S_i, T_i \le n$，$0 \le W_j \le n$。

【输出格式】

输出到文件***running.out*** 中。

输出 1 行 n 个整数，第 j 个整数表示结点 j 的观察员可以观察到多少人。

【样例1输入】

```
6 3
2 3
```

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

【样例1输出】

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

【样例1说明】

对于 1 号点，$W_1 = 0$，故只有起点为 1 号点的玩家才会被观察到，所以玩家 1 和玩家 2 被观察到，共 2 人被观察到。

对于 2 号点，没有玩家在第 2 秒时在此结点，共 0 人被观察到。

对于 3 号点，没有玩家在第 5 秒时在此结点，共 0 人被观察到。

对于 4 号点，玩家 1 被观察到，共 1 人被观察到。

对于 5 号点，玩家 1 被观察到，共 1 人被观察到。

对于 6 号点，玩家 3 被观察到，共 1 人被观察到。

【样例2输入】

```
5 3
1 2
2 3
2 4
1 5
0 1 0 3 0
3 1
1 4
5 5
```

【样例2输出】

```
1 2 1 0 1
```

【子任务】

每个测试点的数据规模及特点如下表所示。提示：数据范围的个位上的数字可以帮助判断是哪一种数据类型。

测试点编号	n	m	约定
1	$= 991$	$= 991$	所有人的起点等于自己的终点，即$S_i = T_i$
2			
3	$= 992$	$= 992$	$W_j = 0$
4			
5	$= 993$	$= 993$	无
6	$= 99994$	$= 99994$	树退化成一条链，其中1与2有边，2与3有边，...，$n-1$与n有边
7			
8			
9	$= 99995$	$= 99995$	所有的$S_i = 1$
10			
11			
12			
13	$= 99996$	$= 99996$	所有的$T_i = 1$
14			
15			
16			
17	$= 99997$	$= 99997$	无
18			
19			
20	$= 299998$	$= 299998$	

【提示】

如果你的程序需要用到较大的栈空间（这通常意味着需要较深层数的递归），请务必仔细阅读选手目录下的文档***running/stack.pdf***，以了解在最终评测时栈空间的限制与在当前工作环境下调整栈空间限制的方法。

在最终评测时，调用栈占用的空间大小不会有单独的限制，但在我们的工作环境中默认会有 8 MB 的限制。这可能会引起函数调用层数较多时，程序发生栈溢出崩溃。

我们可以使用一些方法修改调用栈的大小限制。例如，在终端中输入下列命令

```
ulimit -s 1048576
```

此命令的意义是，将调用栈的大小限制修改为 1 GB。

例如，在选手目录建立如下 ***sample.cpp*** 或 ***sample.pas***

sample.cpp

```
void dfs(int a){
    if(a == 0)
        return;
    int t = a;
    dfs(a - 1);
}
int main(){
    dfs(1000000);
    return 0;
}
```

sample.pas

```
procedure dfs(a: longint);
    var t: longint;
    begin
        if a = 0 then
            exit;
        t := a;
        dfs(a - 1);
    end;
begin
    dfs(1000000);
end.
```

将上述源代码编译为可执行文件 ***sample*** 后，可以在终端中运行如下命令运行该程序

```
./sample
```

如果在没有使用命令“`ulimit -s 1048576`”的情况下运行该程序，***sample*** 会因为栈溢出而崩溃；如果使用了上述命令后运行该程序，该程序则不会崩溃。

特别地，当你打开多个终端时，它们并不会共享该命令，你需要分别对它们运行该命令。

请注意，**调用栈占用的空间会计入总空间占用中，和程序其他部分占用的内存共同受到内存限制**。

换教室（classroom）

【问题描述】

对于刚上大学的牛牛来说，他面临的第一个问题是如何根据实际情况申请合适的课程。

在可以选择的课程中，有 $2n$ 节课程安排在 n 个时间段上。在第 i（$1 \le i \le n$）个时间段上，两节内容相同的课程同时在不同的地点进行，其中，牛牛预先被安排在教室 c_i 上课，而另一节课程在教室 d_i 进行。

在不提交任何申请的情况下，学生们需要按时间段的顺序依次完成所有的 n 节安排好的课程。如果学生想更换第 i 节课程的教室，则需要提出申请。若申请通过，学生就可以在第 i 个时间段去教室 d_i 上课，否则仍然在教室 c_i 上课。

由于更换教室的需求太多，申请不一定能获得通过。通过计算，牛牛发现申请更换第 i 节课程的教室时，申请被通过的概率是一个已知的实数 k_i，并且对于不同课程的申请，被通过的概率是互相独立的。

学校规定，所有的申请只能在学期开始前一次性提交，并且每个人只能选择**至多** m 节课程进行申请。这意味着牛牛必须一次性决定是否申请更换每节课的教室，而不能根据某些课程的申请结果来决定其他课程是否申请；牛牛可以申请自己最希望更换教室的 m 门课程，也可以**不用完**这 m 个申请的机会，甚至可以一门课程都不申请。

因为不同的课程可能会被安排在不同的教室进行，所以牛牛需要利用课间时间从一间教室赶到另一间教室。

牛牛所在的大学有 v 个教室，有 e 条道路。每条道路连接两间教室，并且是可以**双向通行**的。由于道路的长度和拥堵程度不同，通过不同的道路耗费的体力可能会有所不同。当第 i（$1 \le i \le n-1$）节课结束后，牛牛就会从这节课的教室出发，选择一条耗费体力最少的**路径**前往下一节课的教室。

现在牛牛想知道，申请哪几门课程可以使他因在教室间移动耗费的体力值的总和的**期望值**最小，请你帮他求出这个最小值。

【输入格式】

从文件***classroom.in*** 中读入数据。

第一行四个整数 n, m, v, e。 n 表示这个学期内的时间段的数量； m 表示牛牛最多可以申请更换多少节课程的教室； v 表示牛牛学校里教室的数量； e 表示牛牛的学校里道路的数量。

第二行 n 个正整数，第 i（$1 \le i \le n$）个正整数表示 c_i，即第 i 个时间段牛牛被安排上课的教室；保证 $1 \le c_i \le v$。

第三行 n 个正整数，第 i（$1 \le i \le n$）个正整数表示 d_i，即第 i 个时间段另一间上同样课程的教室；保证 $1 \le d_i \le v$。

第四行 n 个实数，第 i（$1 \le i \le n$）个实数表示 k_i，即牛牛申请在第 i 个时间段更换教室获得通过的概率。保证 $0 \le k_i \le 1$。

接下来 e 行，每行三个正整数 a_j， b_j， w_j，表示有一条双向道路连接教室 a_j，b_j，通过这条道路需要耗费的体力值是 w_j；保证 $1 \le a_j, b_j \le v$， $1 \le w_j \le 100$。

保证 $1 \le n \le 2000$， $0 \le m \le 2000$， $1 \le v \le 300$， $0 \le e \le 90000$。

保证通过学校里的道路，从任何一间教室出发，都能到达其他所有的教室。

保证输入的实数最多包含 3 位小数。

【输出格式】

输出到文件***classroom.out*** 中。

输出一行，包含一个实数，四舍五入精确到小数点后**恰好 2 位**，表示答案。你的输出必须和标准输出**完全一样**才算正确。

测试数据保证四舍五入后的答案和准确答案的差的绝对值不大于 4×10^{-3}。（如果你不知道什么是浮点误差，这段话可以理解为：对于大多数的算法，你可以正常地使用浮点数类型而不用对它进行特殊的处理）

【样例1输入】

```
3 2 3 3
2 1 2
1 2 1
0.8 0.2 0.5
1 2 5
1 3 3
2 3 1
```

【样例1输出】

```
2.80
```

【样例1说明】

所有可行的申请方案和期望收益如下表：

申请更换教室的时间段	申请通过的时间段	出现的概率	耗费的体力值	耗费的体力值的期望
无	无	1.0	8	8.0
1	1	0.8	4	4.8
	无	0.2	8	
2	2	0.2	0	6.4
	无	0.8	8	
3	3	0.5	4	6.0
	无	0.5	8	
1、2	1、2	0.16	4	4.48
	1	0.64	4	
	2	0.04	0	
	无	0.16	8	
1、3	1、3	0.4	0	2.8
	1	0.4	4	
	3	0.1	4	
	无	0.1	8	
2、3	2、3	0.1	4	5.2
	2	0.1	0	
	3	0.4	4	
	无	0.4	8	

【样例2】

见选手目录下的***classroom/classroom2.in*** 与***classroom/classroom2.ans***。

【提示】

1. 道路中可能会有**多条**双向道路连接相同的两间教室。也有可能有道路两端连接的是**同一间**教室。
2. 请注意区分 n, m, v, e 的意义， n **不是**教室的数量， m **不是**道路的数量。

【子任务】

测试点	n	m	v	特殊性质1	特殊性质2
1	≤ 1	≤ 1	≤ 300	×	×
2	≤ 2	≤ 0	≤ 20		
3		≤ 1	≤ 100		
4		≤ 2	≤ 300		
5	≤ 3	≤ 0	≤ 20	√	√
6		≤ 1	≤ 100		×
7		≤ 2	≤ 300	×	
8	≤ 10	≤ 0		√	√
9		≤ 1	≤ 20		×
10		≤ 2	≤ 100	×	
11		≤ 10	≤ 300		√
12	≤ 20	≤ 0	≤ 20	√	×
13		≤ 1	≤ 100	×	
14		≤ 2	≤ 300	√	
15		≤ 20		×	√
16	≤ 300	≤ 0	≤ 20		×
17		≤ 1	≤ 100		
18		≤ 2	≤ 300	√	√
19		≤ 300		×	
20	≤ 2000	≤ 0	≤ 20		×
21		≤ 1			
22		≤ 2	≤ 100		
23		≤ 2000			
24			≤ 300		
25					

特殊性质1：图上任意两点 a_i ， b_i ， $a_i \neq b_i$ 间，存在一条耗费体力最少的路径只包含一条道路。

特殊性质2：对于所有的 $1 \leq i \leq n$ ， $k_i = 1$ 。