全国青少年信息学奥林匹克竞赛

CCF NOI2021 统一省选

Day2（A 卷）

时间：2021 年 4 月 11 日 08:30 ～ 13:00

题目名称	宝石	滚榜	支配
题目类型	传统型	传统型	传统型
目录	gem	ranklist	dominator
可执行文件名	gem	ranklist	dominator
输入文件名	gem.in	ranklist.in	dominator.in
输出文件名	gem.out	ranklist.out	dominator.out
每个测试点时限	2.0 秒	1.0 秒	1.0 秒
内存限制	512 MB	512 MB	512 MB
子任务数目	20	20	20
测试点是否等分	是	是	是

提交源程序文件名

对于 C++　语言	gem.cpp	ranklist.cpp	dominator.cpp

编译选项

对于 C++　语言	-O2

注意事项与提醒（请选手务必仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C++ 中主函数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。
3. 提交的程序代码文件的放置位置请参照各省的具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 全国统一评测时采用的机器配置为：Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz，内存 32GB。上述时限以此配置为准。
8. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。
9. 最终评测时所用的编译命令中不含编译选项之外的任何优化开关。

宝石（gem）

【题目描述】

欧艾大陆上有 n 座城市，城市从 $1 \sim n$ 编号，所有城市经由 $n-1$ 条无向道路互相连通，即 n 座城市与 $n-1$ 条道路构成了一棵树。

每座城市的集市上都会出售宝石，总共有 m 种不同的宝石，用 $1 \sim m$ 编号。i 号城市的集市出售的是第 w_i 种宝石，一种宝石可能会在多座城市的集市出售。

K 神有一个宝石收集器。这个宝石收集器能按照顺序收集至多 c 颗宝石，其收集宝石的顺序为：$P_1, P_2, \cdots, P_c$。更具体地，收集器需要先放入第 P_1 种宝石，然后才能再放入第 P_2 种宝石，之后再能放入第 P_3 种宝石，以此类推。其中 $P_1, P_2, \cdots, P_c$ 互不相等。

K 神到达一个城市后，如果该城市的集市上出售的宝石种类和当前收集器中需要放入的种类相同，则他可以在该城市的集市上购买一颗宝石并放入宝石收集器中；否则他只会路过该城市什么都不做。

现在 K 神给了你 q 次询问，每次给出起点 s_i 与终点 t_i，他想知道如果从 s_i 号城市出发，沿最短路线走到 t_i 号城市后，他的收集器中最多能收集到几个宝石？（在每次询问中，收集器内初始时没有任何宝石。起点与终点城市集市上的宝石可以尝试被收集）

【输入格式】

从文件 ***gem.in*** 中读入数据。

第一行包含三个正整数 n, m, c，分别表示城市数，宝石种类数，收集器的容量。

第二行包含 c 个正整数 P_i。数据保证 $1 \leq P_i \leq m$ 且这些数互不相等。

第三行包含 n 个正整数 w_i，表示每个城市集市上出售的宝石种类。

接下来 $n-1$ 行，每行两个正整数 u_i, v_i，表示一条连接 u_i 和 v_i 号城市的道路。

第 $n+3$ 行包含一个正整数 q 表示询问次数。

接下来 q 行，每行两个正整数 s_i，t_i 表示该次询问的起点与终点。

【输出格式】

输出到文件 ***gem.out*** 中。

按输入顺序输出 q 行，每行一个整数表示询问的答案。

【样例 1 输入】

```
7 3 3
2 3 1
2 1 3 3 2 1 3
1 2
```

```
2 3
1 4
4 5
4 6
6 7
5
3 5
1 3
7 3
5 7
7 5
```

【样例 1 输出】

```
2
2
2
3
1
```

【样例 2】

见选手目录下的 ***gem/gem2.in*** 与 ***gem/gem2.ans***。

【样例 3】

见选手目录下的 ***gem/gem3.in*** 与 ***gem/gem3.ans***。

【数据范围】

对于所有测试数据：$1 \le n, q \le 2 \times 10^5$，$1 \le c \le m \le 5 \times 10^4$，$1 \le w_i \le m$。
每个测试点的具体限制见下表：

测试点编号	$n, q \le$	特殊限制
$1 \sim 2$	10	无
$3 \sim 5$	1000	
$6 \sim 10$	2×10^5	$m \le 300$
$11 \sim 14$		$u_i = i$，$v_i = i + 1$
$15 \sim 20$		无

滚榜（ranklist）

【题目描述】

封榜是 ICPC 系列竞赛中的一个特色机制。ICPC 竞赛是实时反馈提交结果的程序设计竞赛，参赛选手与场外观众可以通过排行榜实时查看每个参赛队伍的过题数与排名。竞赛的最后一小时会进行“封榜”，即排行榜上将隐藏最后一小时内的提交的结果。赛后通过滚榜环节将最后一小时的结果（即每只队伍最后一小时的过题数）公布。

Alice 围观了一场 ICPC 竞赛的滚榜环节。本次竞赛共有 n 支队伍参赛，队伍从 $1 \sim n$ 编号，i 号队伍在封榜前通过的题数为 a_i。排行榜上队伍按照过题数从大到小进行排名，若两支队伍过题数相同，则编号小的队伍排名靠前。

滚榜时主办方以 b_i 不降的顺序依次公布了每支队伍在封榜后的过题数 b_i（最终该队伍总过题数为 $a_i + b_i$），并且每公布一支队伍的结果，排行榜上就会实时更新排名。Alice 并不记得队伍被公布的顺序，也不记得最终排行榜上的排名情况，只记得每次公布后，本次被公布结果的队伍都成为了新排行榜上的第一名，以及所有队伍在封榜后一共通过了 m 道题（即 $\sum_{i=1}^{n} b_i = m$）。

现在 Alice 想请你帮她算算，最终排行榜上队伍的排名情况可能有多少种。

【输入格式】

从文件 ***ranklist.in*** 中读入数据。

第一行包含两个正整数 n, m，表示队伍数量与它们封榜后的总过题数。

第二行包含 n 个正整数表示 a_i。

【输出格式】

输出到文件 ***ranklist.out*** 中。

仅一行一个整数表示答案。

【样例 1 输入】

```
3 6
1 2 1
```

【样例 1 输出】

```
5
```

【样例 1 解释】

1. 最终排名：$1,3,2$，滚榜情况（按公布顺序，下同）：$b_2=0$，$b_3=2$，$b_1=4$。
2. 最终排名：$2,1,3$，滚榜情况：$b_3=2$，$b_1=2$，$b_2=2$。
3. 最终排名：$2,3,1$，滚榜情况：$b_1=1$，$b_3=2$，$b_2=3$。
4. 最终排名：$3,1,2$，滚榜情况：$b_2=0$，$b_1=2$，$b_3=4$。
5. 最终排名：$3,2,1$，滚榜情况：$b_1=1$，$b_2=1$，$b_3=4$。

【样例 2 输入】

```
6 50
4 7 9 3 0 3
```

【样例 2 输出】

```
96
```

【样例 3 输入】

```
11 300
6 8 8 12 0 11 6 1 0 15 5
```

【样例 3 输出】

```
30140983
```

【数据范围】

对于所有测试数据：$1\le n\le 13$，$1\le m\le 500$，$0\le a_i\le 10^4$。

每个测试点的具体限制见下表：

测试点编号	$n\le$	$m\le$
$1\sim 2$	2	10
$3\sim 5$	3	
$6\sim 8$	8	100
$9\sim 12$	10	200
$13\sim 16$	12	300
$17\sim 20$	13	500

支配（dominator）

【题目描述】

给定一张 n 个点 m 条边的有向图 G，其顶点从 $1 \sim n$ 编号。

对于任意两个点 u, v，若从顶点 1 出发到达顶点 v 的所有路径都需要经过顶点 u，则称顶点 u 支配顶点 v。特别地，每个顶点支配其自身。

对于任意一个点 v，我们将图中支配顶点 v 的顶点集合称为 v 的受支配集 D_v。

现在有 q 次互相独立的询问，每次询问给出一条有向边，请你回答在图 G 中加入该条边后，有多少个顶点的受支配集发生了变化。

【输入格式】

从文件 ***dominator.in*** 中读入数据。

第一行三个整数 n, m, q，分别表示图中的顶点数、边数，以及询问数。

接下来 m 行每行两个整数 x_i，y_i，表示一条有向边 $x_i \to y_i$。

接下来 q 行每行两个整数 s_i，t_i，表示每次询问加入的边 $s_i \to t_i$。

数据保证给出的图 G 中，从 1 号点出发能到达所有其他点，并且图中不包含重边与自环。

【输出格式】

输出到文件 ***dominator.out*** 中。

对于每个询问输出一行一个整数表示答案。

【样例 1 输入】

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

【样例 1 输出】

```
1
0
2
```

【样例 1 解释】

对于原图，六个点的受支配集分别为：$D_1 = \{1\}$，$D_2 = \{1, 2\}$，$D_3 = \{1, 3\}$，$D_4 = \{1, 3, 4\}$，$D_5 = \{1, 3, 4, 5\}$，$D_6 = \{1, 2, 6\}$。

加入 $5 \to 6$ 后，$D_6 = \{1, 6\}$，其他点受支配集不变。

加入 $3 \to 2$ 后没有点受支配集改变。

加入 $2 \to 4$ 后 $D_4 = \{1, 4\}$，$D_5 = \{1, 4, 5\}$，其他点受支配集不变。

【样例 2】

见选手目录下的 ***dominator/dominator2.in*** 与 ***dominator/dominator2.ans***。

【样例 3】

见选手目录下的 ***dominator/dominator3.in*** 与 ***dominator/dominator3.ans***。

【数据范围】

对于所有测试数据：$1 \le n \le 3000$，$1 \le m \le 2 \times n$，$1 \le q \le 2 \times 10^4$。

每个测试点的具体限制见下表：

测试点编号	$n \le$	特殊性质
$1 \sim 2$	10	无
$3 \sim 6$	100	$q \le 100$
$7 \sim 9$	1000	$m = n - 1$
$10 \sim 15$		$q \le 2000$
$16 \sim 20$	3000	无