2024 青少年编程挑战赛决赛

（提高级）

考试时间： 2024 年 4 月 21 日 8:30-12:00

题目名称	公共子串种类数	网络安全	包裹	数列排序
题目类型	传统型	传统型	传统型	传统型
目录	substring	security	package	series
可执行文件名	substring	security	package	series
输入文件名	substring.in	security.in	package.in	series.in
输出文件名	substring.out	security.out	package.out	series.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	2.0 秒
内存限制	512MB	512MB	512MB	512MB
子任务数目	10	10	10	10
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	substring.cpp	security.cpp	package.cpp	series.cpp
对于 C 语言	substring.c	security.c	package.c	series.c

编译选项

对于 C++ 语言	-O2 -lm
对于 C 语言	-O2 -lm

注意事项（请仔细阅读）

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/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。
3. 提交的程序代码文件的放置位置请参考文件提交格式的具体要求。
4. 因违反以上三点而出现的错误或问题，申述时一律不予受理。
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. 统一评测时采用的机器配置为：Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz 16G 内存。上述时限以此配置为准。
7. 只提供 Linux 格式附加样例文件。
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此为准。

2024 青少年编程挑战赛（提高级）

公共子串种类数（substring）

【题目描述】

在一个字符串中，子串是在字符串的基础上去掉 0 个或若干个字符后所形成的字符串，现有字符串 $"aabbcc"$,其中 $"abc"$、$"aa"$、$"abbc"$、$"bbcc"$等都是它的子串，现在给出3个字符串，找出 3 个字符串共同含有多少种子串（不算空串），在公共子串中出现位置不同，只算一种。

【输入格式】

从文件 **substring.in** 中读入数据。

每组数据只含 3 行，每行都是只包含小写字母的字符串。

【输出格式】

输出到文件 **substring.out** 中。

输出 3 个字符串共有的公共子串种类数。

【样例 1 输入】

```
apartment
apache
approach
```

【样例 1 输出】

```
6
```

【样例 1 解释】

3 个字符串共有的公共子串有：“a”、“p”、“ap”、“pa”、“aa”、“apa”。 其中子串 “a” 有多个，但由于统计的是公共子串种类，所以 “a” 只算 1 种子串。

【数据范围】

100%的数据中，字符串的长度不超过 100。字符串中只含有小写字母。

网络安全（security）

【题目描述】

小向正在设计一款网络安全应用程序，其中有许多功能模块需要连接。他已经确定了应用程序的模块结构，该安全应用程序由n个功能模块构成，这些功能模块通过m个程序接口连接（双向数据通信）。在某几个安全程序的接口处，有着敏感数据或者核心功能块，需要额外的安全功能块。小向希望你帮助他确定在哪些接口上添加额外的安全功能块。

安全功能块将被安装在接口s处开始，于接口t处结束。通过计算确定这些接口之后，小向将在s到t的每个接口中添加安全功能块，使得没有安全功能块的接口就无法从接口s到达接口t。小向希望尽可能多地保护关键接口，因此他请你帮助确定最多需要安装多少安全功能块，且任意的功能模块都可以选择作为s或者t。

【输入格式】

从文件 **security .in** 中读入数据

第一行包含两个正整数n和m，分别表示功能模块的数量和接口的数量。

接下来的m行，每行包含两个正整数x和y（$1 \le x, y \le n, x \ne y$），描述一个接口的两端功能模块编号。

保证不存在通过两个及以上连接相同功能模块的接口，且任意功能模块可从其他任何功能模块通过接口到达。

【输出格式】

输出到文件 **security .out** 中。

输出一个正整数，表示在所有可能的s和t的选择情况下，能够安装的最多安全模块数量。

【样例 1 输入】

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

【样例 1 输出】

```
2
```

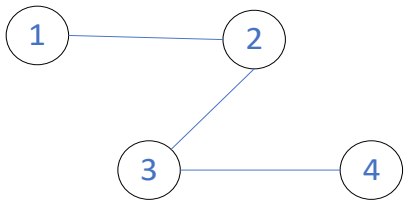
【样例 2 输入】

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

【样例 2 输出】

```
3
```

【样例 2 解释】



样例 2 的（如上图所示），s和t分别选 1、4 两个位置，这时路径经过的接口最多，故添加的安全安全功能块也是最多，为 3 个。

【数据规模】

对于 100%的数据：$2 \le n \le 3 \times 10^5, 1 \le m \le 3 \times 10^5$。

测试点	n≤	m≤
1, 2	50	100
3, 4	500	1000
5, 6	2000	4000
7, 8	10000	20000
9, 10	100000	200000

包裹（package）

【题目描述】

现在我国货物运输以及包裹运输的主要方式仍然是依靠公路运输，包裹运输主要由快递公司统一运送到某一特定的地方，在根据地址或区域分别派送到包裹的指定地点。

京东作为目前的电商巨头之一，在全国很多地方都建立了货运中心，京东每天的订单量都是非常多，同时包裹需要发送到全国各地，这些包裹均会按照省份先运送到各省的物流中心，在进行分拣发送到各个市、区/县……，这样的运输网络我们可以看成一棵树，货运中心就为根节点，各省物流中就为根节点的子结点，以此类推。

昆明就有一个京东的货运仓库，假设现在京东收到了一批订单，这批订单需要发送到n个地方，编号从1到n，到达每个站点的包裹有w个，站点和站点之间只存在一条路线；在运输过程中可能会出现，用户退货或者站点新收包裹需要寄送的情况，对于这两种情况京东会在某个运输站点将退货包裹挑拣出来重新返回仓库，将需要寄送包裹加入运输的包裹中，则在此站点的包裹数量将会减少或增加；现在为了减少各站点的工作量，有如下几个操作：

（1） $CHANGE\ u\ t$：站点u的包裹数量为t。

（2） $QMAX\ u\ v$：询问从站点u到站点v的包裹那个站点的包裹数量最多。

（3） $QSUM\ u\ v$：询问从站点u到站点v的包裹总和。

注意：从点u到点v的路径上的结点包括u和v本身。

【输入格式】

从文件 **package.in** 中读入数据。

第一行为1个正整数n，表示结点的个数。

接下来的$n-1$行，每行2个整数x和y，表示站点x和站点y之间有一条路线。

接下来的n个整数，第i个正整数w_i，表示站点i的包裹数量。

接下来的1行表示有q次操作。

接下来的q行，每行一个操作，以$CHANGE\ u\ t$或$QMAX\ u\ v$或$QSUM\ u\ v$的形式给出。

【输出格式】

输出到文件 **package.out** 中。

对于每个$QMAX$或$QSUM$的操作，每行一个整数表示要求输出的结果。

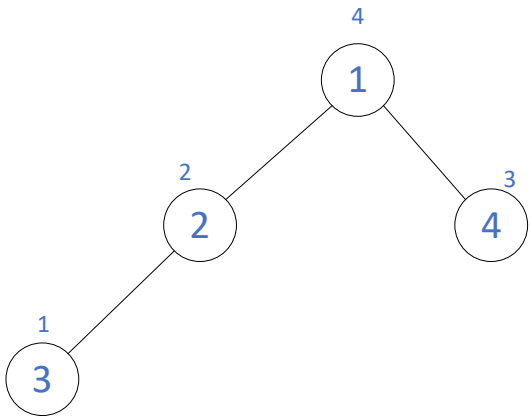
【样例 1 输入】

```
4
1 2
2 3
4 1
4 2 1 3
12
QMAX 3 4
QMAX 3 3
QMAX 3 2
QMAX 2 3
QSUM 3 4
QSUM 2 1
CHANGE 1 5
QMAX 3 4
CHANGE 3 6
QMAX 3 4
QMAX 2 4
QSUM 3 4
```

【样例 1 输出】

```
4
1
2
2
10
6
5
6
5
16
```

【样例 1 解释】



共12次操作：

第1次查询3到4路径上的最大值，具体路径为$3-2-1-4$，路径结点中权值最大为1号结点，权值为4，故输出4。

第2、3、4次查询与第一次类似。

第5次查询3到4路径权值之和，具体路径为$3-2-1-4$，结点权值之和为$1+2+4+3=10$，故输出10。

第6次查询与第5次类似。

第7次为改变1号结点的权值为5。

……

【数据范围】

对于 100%的数据，保证 $1 \le n \le 3 \times 10^4$，$0 \le q \le 2 \times 10^5$。

中途操作中保证每个节点的权值 w 在 -3×10^4 到 3×10^4 之间。

数据点	n≤	q≤
1，2	100	100
3，4	500	1000
5，6	5000	20000
7，8	20000	100000
9，10	30000	200000

数列排序（series）

【题目描述】

在今年，小明的姐姐喜欢上了数字序列。所以她经常研究关于序列的一些奇奇怪怪的问题，现在她在研究一个难题，需要你来帮助她。

这个难题是这样的：给出一个排列，这个序列有n个数，现在对这个排列序列进行m次局部排序的操作，排序分为两种：

$0\ l\ r$ 表示将区间$[l,r]$的数字升序排序;

$1\ l\ r$ 表示将区间$[l,r]$的数字降序排序;

注意，这里是对下标在区间$[l,r]$ 内的数排序。

m次操作后，最后询问第 q 位置上的数字。

【输入格式】

从文件 **series.in** 中读入数据。

输入数据的第一行为两个整数n和m，n表示序列的长度，m表示局部排序的次数。

第二行为n个整数，表示排列的具体数值。

接下来输入m行，每一行有三个整数op、l、r，op 为0代表升序排序，op 为 1 代表降序排序，l、r表示排序的区间。

最后输入一个整数q，表示排序完之后询问的位置。

【输出格式】

输出到文件 **series.out** 中。

输出数据仅有一行，一个整数，表示按照顺序将全部的部分排序结束后第 q 位置上的数字。

【样例 1 输入】

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

【样例 1 输出】

```
5
```

【样例 1 解释】

初始序列为{1,6,2,5,3,4}。

op = 0，对[1,4]区间进行升序后，序列为{1,2,5,6,3,4}；

op = 1，对[3,6]区间进行降序后，序列为{1,2,6,5,4,3}；

op = 0，对[2,4]区间进行升序后，序列为{1,2,5,6,4,3}；

最后查询第三个位置的数为：5，故输出5。

【数据规模】

对于 20% 的数据，$n, m \leq 1000$;

对于 100%的数据，$n, m \leq 10^5$，$1 \leq q \leq n$。

数据点	n≤	m≤	q≤
1，2	1000	1000	500
3，4	10000	50000	10000
5，6	20000	50000	20000
7，8	50000	100000	50000
9，10	100000	100000	100000