C++编程提高组模拟卷

题目名称	靶形数独	最大的矩形	骑士	数据传输
题目类型	传统型	传统型	传统型	传统型
目录	sudoku	rectangle	knight	transmit
可执行文件名	sudoku	rectangle	knight	transmit
输入文件名	sudoku.in	rectangle.in	knight.in	transmit.in
输出文件名	sudoku.out	rectangle.out	knight.out	transmit.out
每个测试点时限	2.0 秒	1.0 秒	2.0 秒	1.0 秒
内存限制	128MB	128MB	128MB	512MB
子任务数目	20	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	sudoku.cpp	rectangle.cpp	knight.cpp	transmit.cpp
对于C 语言	sudoku.c	rectangle.c	knight.c	transmit.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）

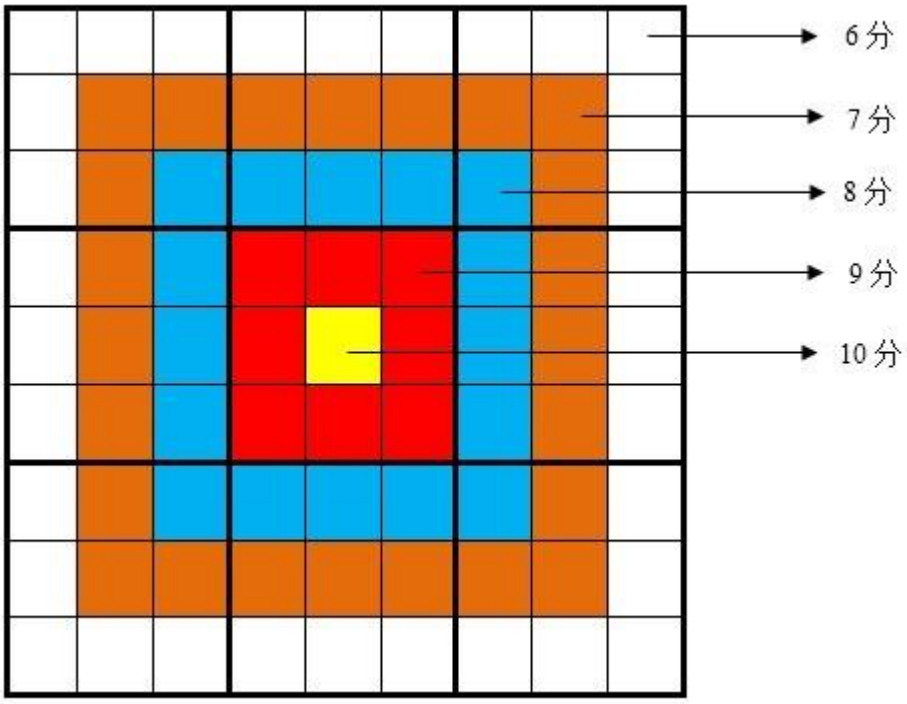
1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参考文件提交格式的具体要求。
4. 因违反以上三点而出现的错误或问题，申述时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 全省统一评测时采用的机器配置为：Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz 16G 内存。上述时限以此配置为准。
7. 只提供 Linux 格式附加样例文件。
8. 评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以此为准。

靶形数独（sudoku）

【题目描述】

小城和小华都是热爱数学的好学生，最近，他们不约而同地迷上了数独游戏，好胜的他们想用数独来一比高低。但普通的数独对他们来说都过于简单了，于是他们向 Z 博士请教，Z 博士拿出了他最近发明的“靶形数独”，作为这两个孩子比试的题目。

靶形数独的方格同普通数独一样，在 9 格宽×9 格高的大九宫格中有 9 个 3 格宽×3 格高的小九宫格（用粗黑色线隔开的）。在这个大九宫格中，有一些数字是已知的，根据这些数字，利用逻辑推理，在其他的空格上填入 1 到 9 的数字。每个数字在每个小九宫格内不能重复出现，每个数字在每行、每列也不能重复出现。但靶形数独有一点和普通数独不同，即每一个方格都有一个分值，而且如同一个靶子一样，离中心越近则分值越高。（如图）



上图具体的分值分布是：最里面一格（黄色区域）为 10 分，黄色区域外面的一圈（红色区域）每个格子为 9 分，再外面一圈（蓝色区域）每个格子为 8 分，蓝色区域外面一圈（棕色区域）每个格子为 7 分，最外面一圈（白色区域）每个格子为 6 分，如上图所示。比赛的要求是：每个人必须完成一个给定的数独（每

个给定数独有可能有不同的填法），而且要争取更高的总分数。而这个总分数即每个方格上的分值和完成这个数独时填在相应格上的数字的乘积的总和。如图，在以下这个已经填完数字的靶形数独游戏中，总分为 2829。游戏规定，将以总分数的高低决出胜负。

7	5	4	9	3	8	2	6	1
1	2	8	6	4	5	9	3	7
6	3	9	2	1	7	4	8	5
8	6	5	4	2	9	1	7	3
9	7	2	3	5	1	6	4	8
4	1	3	8	7	6	5	2	9
5	4	7	1	8	2	3	9	6
2	9	1	7	6	3	8	5	4
3	8	6	5	9	4	7	1	2

由于求胜心切，小城找到了善于编程的你，让你帮他求出，对于给定的靶形数独，能够得到的最高分数。

【输入格式】

从文件 **sudoku.in** 中读入数据。每组输入数据一共 9 行，每行 9 个整数（每个数都在 0—9 的范围内），表示一个尚未填满的数独方格，未填满的空格用“0”表示。每两个数字之间用一个空格隔开。

数据规模：

40%的数据，数独中非 0 数的个数不少于 30；

80%的数据，数独中非 0 数的个数不少于 26；

100%的数据，数独中非 0 数的个数不少于 24。

【输出格式】

输出到文件 **sudoku.out** 中。每组输出共 1 行。输出可以得到的靶形数独的最高分数。如果这个数独无解，则输出整数-1。

【样例 1 输入】

```
700900001
100005900
000200080
005020003
000000648
413000000
007002090
201060804
080504012
```

【样例 1 输出】

```
2829
```

【样例 2 输入】

```
000702453
900008000
740005010
195080000
070000025
030579108
000601000
060900001
000000006
```

【样例 2 输出】

```
2852
```

最大的矩形(rectangle)

【题目背景】

给定 n 个非负整数，用来表示柱状图中各个柱子的高度。每个柱子彼此相邻，且宽度为 1 。

求在该柱状图中，能够勾勒出来的矩形的最大面积。

【输入格式】

从文件 **rectangle.in** 中读入数据。第一行数字n，表示柱状图的柱子个数。

第二行 n 个数字，a_i表示柱状图第 i 个柱子的高度

【输出格式】

输出到文件 **rectangle.out** 中。

一个正整数。表示能够勾勒出来的矩形的最大面积。

【样例 1 输入】

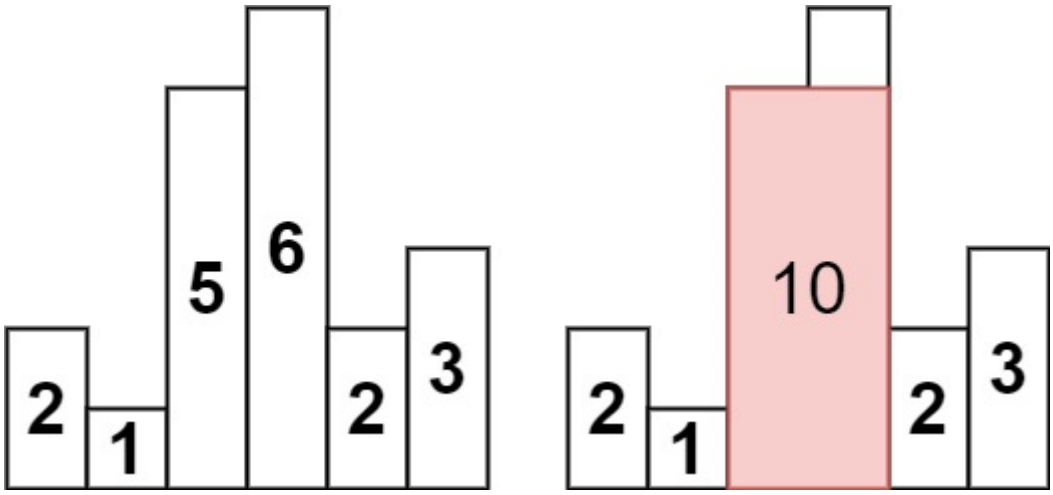
```
6
2 1 5 6 2 3
```

【样例 1 输出】

```
10
```

【样例 1 解释】

柱状图如下图所示，最大的矩形面积如右图所示。



【样例 2 输入】

```
2
```

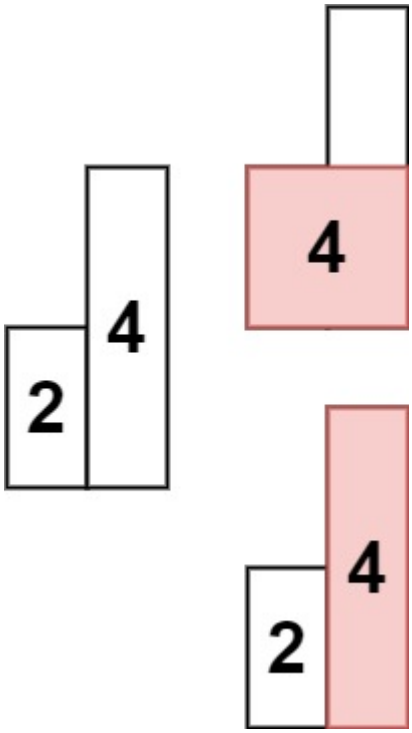
```
2 4
```

【样例 2 输出】

```
4
```

【样例 2 解释】

柱状图如下图所示。



【数据范围】

数据保证，对于所有数据，$1 \le n \le 10^5, 1 \le a_i \le 10^4$。

测试点	$n \le$
1～2	10^1
3～4	10^2
5～6	10^3
7～8	10^4
9～10	10^5

骑士（knight）

【题目描述】

Z 国的骑士团是一个很有势力的组织，帮会中汇聚了来自各地的精英。他们劫富济贫，惩恶扬善，受到社会各界的赞扬。

最近发生了一件可怕的事情，邪恶的 Y 国发动了一场针对 Z 国的侵略战争。战火绵延五百里，在和平环境中安逸了数百年的 Z 国又怎能抵挡的住 Y 国的军队。于是人们把所有的希望都寄托在了骑士团的身上，就像期待有一个真龙天子的降生，带领正义打败邪恶。

骑士团是肯定具有打败邪恶势力的能力的，但是骑士们互相之间往往有一些矛盾。每个骑士都有且仅有一个自己最厌恶的骑士（当然不是他自己），他是绝对不会与自己最厌恶的人一同出征的。

战火绵延，人民生灵涂炭，组织起一个骑士军团加入战斗刻不容缓！国王交给了你一个艰巨的任务，从所有的骑士中选出一个骑士军团，使得军团内没有矛盾的两人（不存在一个骑士与他最痛恨的人一同被选入骑士军团的情况），并且，使得这支骑士军团最具有战斗力。

为了描述战斗力，我们将骑士按照 1 至 n 编号，给每名骑士一个战斗力的估计，一个军团的战斗力为所有骑士的战斗力总和。

【输入格式】

从文件 **knight.in** 中读入数据。第一行包含一个整数 n，描述骑士团的人数。

接下来 n 行，每行两个整数，按顺序描述每一名骑士的战斗力和他最痛恨的骑士。

【输出格式】

输出到文件 **knight.out** 中。应输出一行，包含一个整数，表示你所选出的骑士军团的战斗力。

【样例 1 输入】

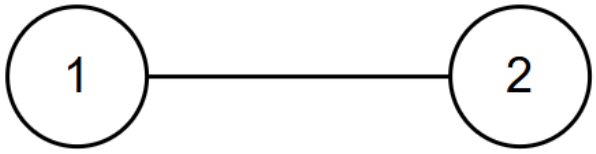
```
3
10 2
20 3
30 1
```

【样例 1 输出】

```
30
```

【样例 1 解释】

只有 2 个节点，1 和 2 直接连接，1 可以染 1 号颜色，2 号可以染 1、2 两种颜色，不过方案只有一种，即 1 号染 1 号颜色，2 号染 2 号颜色。



【数据范围】

对于 30% 的测试数据，满足 $n \leq 10$；
对于 60% 的测试数据，满足 $n \leq 100$；
对于 80% 的测试数据，满足 $n \leq 10^4$ 。
对于 100% 的测试数据，满足 $1 \leq n \leq 10^6$ ，每名骑士的战斗力都是不大于 10^6的正整数。

数据传输（transmit）

【题目描述】

小 C 正在设计计算机网络中的路由系统。

测试用的网络总共有 n 台主机，依次编号为 1~n。这 n 台主机之间由 n−1 根网线连接，第 i 条网线连接个主机a_i和b_i。保证任意两台主机可以通过有限根网线直接或者间接地相连。受制于信息发送的功率，主机 a 能够直接将信息传输给主机 b 当且仅当两个主机在可以通过不超过 k 根网线直接或者间接的相连。

在计算机网络中，数据的传输往往需要通过若干次转发。假定小 C 需要将数据从主机 a 传输到主机 b(a≠b），则其会选择出若干台用于传输的主机$c_1 = a, c_2, \ldots, c_{m-1}, c_m = b$，并按照如下规则转发：对于所有的$1 \le i < m$，主机$c_i$将信息直接发送给$c_{i+1}$。

每台主机处理信息都需要一定的时间，第 i 台主机处理信息需要v_i单位的时间。数据在网络中的传输非常迅速，因此传输的时间可以忽略不计。据此，上述传输过程花费的时间为 $\sum_{i=1}^{m} v_{c_i}$。

现在总共有 q 次数据发送请求，第 i 次请求会从主机s_i发送数据到主机t_i。小 C 想要知道，对于每一次请求至少需要花费多少单位时间才能完成传输。

【输入格式】

从文件 **transmit.in** 中读入数据。

输入的第一行包含三个正整数 n,Q,k，分别表示网络主机个数，请求个数，传输参数。数据保证 $1 \le n \le 2 \times 10^5$ ，$1 \le Q \le 2 \times 10^5$，$1 \le k \le 3$。

输入的第二行包含 n 个正整数，第 i 个正整数表示v_i，保证$1 \le v_i \le 10^9$。

接下来 n−1 行，第 i 行包含两个正整数a_i，b_i，表示一条连接主机a_i，b_i的网线。保证 $1 \le a_i$，$b_i \le n$。

接下来 Q 行，第 i 行包含两个正整数s_i，t_i,表示一次从主机s_i发送数据到主机t_i的请求。保证$1 \le s_i$，$t_i \le n$，$s_i \ne t_i$。

【输出格式】

输出到文件 **transmit.out** 中。

输出一共 Q 行，每行一个正整数，表示第 i 次请求在传输的时候至少需要花费多少单位的时间。

【样例 1 输入】

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

【样例 1 输出】

```
12
12
3
```

【样例 1 解释】

对于第一组请求，由于主机 4,7 之间需要至少 4 根网线才能连接，因此数据无法在两台主机之间直接传输，其至少需要一次转发；我们让其在主机 1 进行一次转发，不难发现主机 1 和主机 4,7 之间都只需要两根网线即可连接，且主机 1 的数据处理时间仅为 1，为所有主机中最小，因此最少传输的时间为 4+1+7=12。

对于第三组请求，由于主机 1,2 之间只需要 1 根网线就能连接，因此数据直接传输就是最优解，最少传输的时间为 1+2=3。

【样例 2】

见附件中的 transmit/transmit2.in 与 transmit/transmit2.ans。该样例满足测试点 2 的限制。

【样例 3】

见附件中的 transmit/transmit3.in 与 transmit/transmit3.ans。该样例满足测试点 3 的限制。

【样例 4】

见附件中的 transmit/transmit4.in 与 transmit/transmit4.ans。该样例满足测试点 20 的限制。

【数据范围】

对于所有的测试数据，满足$1 \le n \le 2 \times 10^5$，$1 \le Q \le 2 \times 10^5$，$1 \le k \le 3$，$1 \le a_i$，$b_i \le n$，$1 \le s_i$，$t_i \le n$，$s_i \ne t_i$。

测试点	$n\leqslant$	$Q\leqslant$	$k=$	特殊性质
1	10	10	2	是
2	10	10	3	是
3	200	200	2	是
4~5	200	200	3	是
6~7	2000	2000	1	否
8~9	2000	2000	2	否
10~11	2000	2000	3	否
12~13	2×10^5	2×10^5	1	否
14	5×10^4	5×10^4	2	是
15~16	10^5	10^5	2	是
17~19	2×10^5	2×10^5	2	否
20	5×10^4	5×10^4	3	是

测试点	$n\leqslant$	$Q\leqslant$	$k=$	特殊性质
21~22	10^5	10^5	3	是
23~25	2×10^5	2×10^5	3	否

特殊性质：保证 $a_i=i+1$，而b_i则从 $1,2,\cdots,$ i 中等概率选取。