IOI2004 中国国家队选拔赛

第一试

题目名称	最优切割	数字搜索	石器时代
目录	day1/cut	day1/regular	day1/stoneage
可执行文件名	cut	regular	stoneage
输入文件名	cut.in	regular.in	---
输出文件名	cut.out	regular.out	---
测试点个数	10	10	10
满分	100	100	100
运行限制	1s	2s	5s
	64M	64M	64M+24M

竞赛时间：2004 年 5 月 9 日上午 8:00-13:00

最优切割

(cut)

【背景描述】

HURRICANE小组的成员最近去工厂实习，在实习的过程中遇到了这样的一个问题。即要在一个模板内，切割出一个零件。现已知模板和零件都是给定的凸多边形，且零件在模板中的位置已经固定。我们知道，对于零件来说，除相邻的两边外，任何两条边的延长线的交点都在模板之外。

由于工厂的加工条件所限，切割时，每一刀必须沿零件的某一条边所在的直线切下，把模板分成两部分，然后保留含有零件的一部分，再继续切割。现定义每一刀的费用为模板上切痕的长度。问如何选择切割顺序，才能使花费最少？

【任务描述】

你的程序需要根据给定的输入，给出符合题意的输出：

- 输入包括模板及零件的形状和坐标；
- 你需要根据给出的输入，计算出把模板切割成为零件的最少花费；
- 输出中只包括一个数，即最少的花费。

【输入格式】: (cut.in)

输入文件包括两个部分，分别描述模板和零件的形状及坐标：

- 第一行为模板的顶点个数n（$3 \le n \le 2000$）。下面的n行每行两个实数x，y（$-1{,}000{,}000 < x, y < 1{,}000{,}000$），为按逆时针方向给出模板顶点的坐标。
- 第$n+2$ 行为零件的顶点个数m（$3 \le m \le 2000$）。下面的m行每行两个实数x，y（$-1{,}000{,}000 < x, y < 1{,}000{,}000$）为按逆时针方向给出的零件顶点的坐标。

【输入样例 1】

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

```
2 1
```

【输入样例 2】

```
4
0 0
3 0
3 3
0 3
4
0 0
2 0
2 3
0 3
```

【输出格式】: (cut.out)

输出文件中只需要包括一个整数，为最少花费四舍五入到整数后的值。

【输出样例 1】

```
8
```

【输出样例 2】

```
3
```

【运行限制】

运行时间	1 秒
内存使用	64M

【评分方法】

本题目一共有十个测试点，每个测试点的分数为总分数的 10%。对于每个测试点来说，如果你给出的答案正确，那么你将得到该测试点全部的分数，否则得 0 分。

数字搜索

(regular)

【背景描述】

HURRICANE小组最近接到了一个搜索文本的任务，即从一个由数字构成的长文本中，匹配满足指定条件的子串。搜索的条件采用形如‘(0+10*1)*10*’这样的正则表达式来描述。其中正则表达式的归纳定义如下：

1. 0, 1, …, 9，0*, 1*, …, 9*是正则表达式；
2. 如果A和B是正则表达式，则(A)，A+B，AB，(A)*都是正则表达式；
3. 只有按以上方法构造出来的表达式才是正则表达式。

其中，A+B表示“或者”关系，AB表示“连接”关系，(A)*表示A的内容“重复”零次或者多次。比如正则表达式(12+3)(4+5)6*，就可以匹配以 124，125，34，35 之一开头，之后接零 0 个或任意多个 6 的字符串（例如字符串 12566）。正则表达式(1+0)*可以匹配所有由 0 和 1 构成的字符串，或者是空串。如果一个正则表达式不能匹配空串，则称它是非空的。**本题考虑的都是非空正则表达式**。

如果在给定文本的某一个位置，存在一个以该位置结束的子串，能够被给定的非空正则表达式匹配，则称该位置是可匹配的。现在HURRICANE小组接到的任务就是找出所有可匹配的位置。你能帮助他们完成这个任务么？

【任务描述】

你的程序需要根据给定的输入，给出符合题意的输出：

- 输入包括一个满足如上定义的正则表达式，以及一长串文本；
- 你需要根据输入的正则表达式及文本，找出文本中所有可匹配的位置；
- 你给出的输出需要包括所有可匹配的位置。

【输入格式】: (regular.in)

输入文件regular.in的第一行描述一个正则表达式，第二行为需要处理的文本：

- 第一行的正则表达式包括由一个空格分开的两个部分：
  - 一个非负整数n（$1 \le n \le 10$），表示我们所要考虑的数字集（即在正则表达式和文本中所出现的数字）是 0, 1, …, n–1。
  - 接下来是一个正则表达式，它由{‘(’, ’)’, ’+’, ’*’}中的 4 个符号和{0, …, n–1}中的数字构成，表达式的长度不超过 500 个字符。
- 第二行为一个由 0 到 n-1 之间数字构成的字符串，为需要处理的文本。

该文本长度不超过 10,000,000 个字符。

【输入样例】

4 12333+33 312331	**说明：**对于输入示例，需要处理的文本是'312331'，其中只有第 5 个字符所在的位置（下划线所在处）是可匹配的。这时正则表达式'12333+33'中的'33'可以与之匹配。

【输出格式】: (regular.out)

输出文件只有一行，包括一些由空格分开的整数，按从小到大的顺序依次输出所需处理的文本中每一个可匹配的位置。

【输出样例】

5

【运行限制】

运行时间	2 秒
内存使用	64M

【有关语言方面的提示】

对于使用 C++选手来说，用 ifstream 读取数据是一个很慢的过程，可能会消耗大量的时间。所以我们建议使用 fread/fgetc/fscanf 读取数据，并使用 setbuf 设置缓存。另，fscanf(file, "%c", &c)的速度经过测试表现也比较慢，可以使用 fgetc(file)代替。

对于使用 Pascal 的选手来说，可以用 SetTextBuf 设置缓存来加快读取文件的速度。也可以考虑使用 BlockRead 来读取文件。

【评分方法】

本题目一共有十个测试点，每个测试点的分数为总分数的 10%。对于每个测试点来说，如果你给出的答案正确，那么你将得到该测试点全部的分数，否则得 0 分。

提示：在本次的测试数据中，有 6 个测试点中的正则表达式不出现'*'，其中有 3 个测试点，正则表达式只由数字和'+'构成。有一个测试点的待处理文本不超过 1,000,000 个字符。该题有严格的时间限制，请尽量优化你的算法。

石器时代 (交互式)

(stoneage)

【背景描述】

借助月光宝盒，HURRICANE 小组的成员们来到了石器时代。在热闹的广场上，不同部落的原始人来来往往进进出出。但是，石器时代是非常容易爆发冲突的。冲突爆发时，处于绝对优势地位的部落总是能够获胜。某一时刻一个部落在广场上有“绝对优势”，指的是该时刻在广场上此部落的人数比其他部落的人数之和还要多。出于安全考虑，HURRICANE 小组的成员都非常迫切的希望知道目前在广场上哪个部落有绝对优势。但不幸的是，HURRICANE 小组成员们只能判断两个原始人是不是属于一个部落。你的任务就是协助 HURRICANE 小组！

【任务描述】

这是一道交互式的题目，你的程序需要和交互库交互并按照约定完成所有库函数的调用：

- 你需要在执行任何其他的操作之前初始化交互库；
- 然后通过调用库函数来完成你对数据的输入及处理，其中包括：
  - 获得原始人进入广场的信息；
  - 获得原始人离开广场的信息；
  - 获得 HURRICANE 小组对于占“绝对优势”的部落的查询，并在此之后返回该信息给交互库。
- 在接到程序结束的任务后，将由程序库终止你程序的执行。你的程序不得自行退出。

【交互方式】（库函数）

我们提供的交互库中包括了如下的函数，现将它们的功能说明如下：

- ***init***：初始化函数。你需要在你程序的一开始调用它，且仅能调用一次。调用后交互库会完成相应的初始化工作，此时，广场上没有原始人（即广场是空的）。

- ***getjob***：你应当调用该函数来获取你的程序下一步要做的工作。我们对本函数返回值作如下约定：
  - 0：表示有一个原始人进入了广场——我们约定第 *i* 个进入广场的原始人编号为 *i*；
  - 一个正整数 *i*：表示编号为 *i* 的原始人离开了广场；

- -1：表示 HURRICANE 小组希望知道当前广场上哪个部落有绝对优势。在下一次调用本函数前你必须调用且仅调用一次 *answer* 函数作为回答；
- -2：表示任务完成。当程序库接到这个任务的时候，它会立即退出程序。

注意：我们约定进入广场的原始人总数不超过 1,000,000；

- ***query(i, j)***：调用 *init* 以后，你随时都可以调用本函数。它的作用是测试编号为 *i* 的和编号为 *j* 的原始人是不是属于同一个部落，返回 1 表示属于同一部落，否则返回 0。
  注意：你必须保证编号为 *i* 和 *j* 的原始人都进入过广场，否则视为对库函数的非法调用。

- ***answer(i)***：*i* 是你的回答。如果目前有部落处于绝对优势地位，则第 *i* 个入场的原始人应该属于该部落并且仍然在场上（*i* 可以是他们中任何一个的入场序号）；如果目前没有部落处于绝对优势，*i* 应该为 0。

【交互样例】

调用	返回	解释
init	N/A	程序初始化，石器时代广场上一个人也没有。
getjob	0	原始人 A 第一个进入广场（编号 1）。
getjob	0	原始人 B 第二个进入广场（编号 2）。
query(1,2)	0	查询原始人 A 和 B，发现他们不属于同一个部落。
getjob	-1	询问：当前存在占绝对优势的部落吗？
answer(0)	N/A	显然不存在。以 0 作为回答。
getjob	0	原始人 C 第三个进入广场（编号 3）。
getjob	-1	再次询问是否存在占绝对优势的部落。
query(3,2)	1	查询 C 和 B，发现他们属于同一个部落。
answer(3)	N/A	C 和 B 都是占绝对优势的部落的成员，回答其中任何一个即可。
getjob	2	原始人 B 离开了广场。
getjob	-1	再次询问。
answer(0)	N/A	当前不存在占绝对优势的部落。
getjob	1	原始人 A 离开了广场。
getjob	-2	得到退出的标志，程序退出。由于是由库方面终止程序，该信息可能不会被你的程序所获得。

【测试说明】

你应该写一个输入文件 stoneage.in 来测试你的程序，文件格式说明如下：

- 第一行为一个正整数 *N*，表示广场上总共将进入的原始人的数目；
- 第 2 行到第*N*+1 行，每一行有一个正整数，第*i*+1 行的数 J_i表示第*i*个进入广场的原始人所属部落的编号（$1 \le J_i \le 1{,}000{,}000$）；
- 从第 *N*+2 行起，每一行含有一个整数，依次为每次调用 *getjob* 的返回值；
- 你需要保证输入文件的最后一行是一个整数-2，表示最后一次调用 *getjob* 的返回值是-2，同时输入结束。

运行程序时测试库会生成 stoneage.log，记录你的程序的每一次对库函数的调用。stoneage.log 会忠实记录你的程序的调用，即使它们是非法的。你可以根据 stoneage.log 来判断你的程序的正确性。

【语言相关的说明】

对于使用 PASCAL 的选手来说，我们提供的 stonelib.ppu 单元包括如下的函数定义：

- `procedure init;`
- `function getjob:longint;`
- `function query(i,j:longint): integer;`
- `procedure answer(ans:longint);`

你可以通过 uses stonelib;语句来使用这个单元并调用其中的函数。

对于使用 C/C++的选手来说，我们提供的 stonelib.h/.o 库包括如下的函数定义：

- `void init();`
- `long getjob();`
- `int query(long i,long j);`
- `void answer(long ans);`

你需要建立一个工程，然后将 stonelib.o 及你的源程序加入到该工程中。然后就可以通过#include “stonelib.h”预编译指令来包含这个头文件并使用并调用其中的函数。

【运行限制】

运行时间	5 秒
内存使用	64M+24M

【评分方法】

本题目一共有十个测试点，每个测试点的分数为总分数的 10%。在测试过程中，如果有如下的情况发生：

- 以任何方式进行直接文件操作；
- 非法调用测试库；
- 使测试库非法退出；
- 自行退出程序；
- 运行时间超过最大允许时间；

那么得 0 分；否则你的交互结果将被用来做进一步的评审：

- 如果你的程序给出的回答不正确，那么 0 分；
- 如果你的程序回答正确，那么：
  - 如果每次 *getjob* 调用后调用 *query* 的次数都不超过 5 次，那么你将得到这个测试点全部的分数；
  - 如果每次 *getjob* 调用后调用 *query* 的次数平均不超过 5 次，那么你将得到这个测试点 80%的分数；
  - 如果每次 *getjob* 调用后调用 *query* 的次数都不超过 10 次，那么你将得到这个测试点 70%的分数；
  - 如果每次 *getjob* 调用后调用 *query* 的次数平均不超过 10 次，那么你将得到这个测试点 60%的分数；
  - 如果每次 *getjob* 调用后调用 *query* 的次数平均超过了 10 次，那么你所得到的分数将按照如下的公式计算：

$$score = 30\% \times \frac{10 \times query\_number}{your\_query\_count}$$

其中 *query_number* 为测试数据中包含绝对优势查询的个数，*your_query_count* 为你的程序调用 *query* 的总次数。

说明：我们的交互库将会使用 24M 的内存，不过这个内存不会记入你的程序中。