Day 2

|  |  |  |  |
| --- | --- | --- | --- |
| 题目名称 | 收益 | 最短路 | 未来程序 |
| 可执行文件名 | value | distance | crack |
| 输入文件名 | value.in | distance.in | crack.in |
| 输出文件名 | value.out | distance.out | crack.out |
| 每个测试点时限 | 2秒 | 1秒 | 1秒 |
| 内存限制 | 256MB | 256MB | 256MB |
| 是否有部分分 | 否 | 否 | 否 |
| 题目类型 | 传统型 | 传统型 | 提交答案型 |
| 是否有样例文件 | 否 | 否 | 否 |
| 是否有附加文件 | 否 | 否 | 否 |

提交源程序须加后缀

|  |  |  |  |
| --- | --- | --- | --- |
| 对于C++ 语言 | value.cpp | distance.cpp | crack.cpp |

编译开关

|  |  |  |  |
| --- | --- | --- | --- |
| 对于C++ 语言 | -O2 -lm | -O2 -lm | -O2 -lm |

nodesnodesnodes1.in~nodes10.innodes1.out~nodes10.outnodes1.ans~nodes10.ans

收益(Value)

【问题描述】

给两个维向量,。

你需要设定两个参数，对于每个，计算收益：

• 若，获得的收益

• 否则，若，获得的收益

• 否则，获得的收益

最大化总收益。

【输入格式】

输入文件为***value.in***。

第一行包含个正整数。

接下来行每行包含个整数。

【输出格式与部分分】

输出文件为***value .out***。

共一行包含1个整数，表示总收益的最大值。

【样例1输入】

5

80 20

60 50

40 40

15 10

70 30

【样例1输出】

220

【子任务】

对于的数据，

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

最短路(Distance)

【问题描述】

给定一个 的网格图和包括 在内的个标记点，你只能通过标记点进行转移。

任意两个标记点间的距离定义为的切比雪夫距离次方，即和间的距离为。

输出一条从到的最短路。

【输入格式】

输入文件为***distance.in***。

第一行包含个正整数。

接下来行每行包含个正整数，表示一个标记点的坐标，其中号点一定是，号点一定是。

【输出格式与部分分】

输出文件为***distance.out***。

第一行包含一个正整数。

第二行包含个正整数，表示一条最短路。表示最短路中到达的第个标记点的编号，其中一定是，一定是。

【样例1输入】

5 6 9

1 1

4 3

4 6

2 5

3 1

3 3

3 6

5 4

5 6

【样例1输出】

5

1 6 2 8 9

【子任务】

所有测试数据的范围和特点如下:

对于的数据，

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

未来程序(Crack)

【问题背景】

本题共分为10 个子任务，下文将有针对每个子任务的具体说明。**每个子任务各对应一种不同的基础算法，并且保证采用的任意单个算法对所有合法输入数据具有普适性，即不会针对有限组输入数据采用特例**。你的目标便是还原这些算法。

为了发掘这些程序所完成的任务，你可以利用下发的可执行文件crack\_force 运行任意自行设计的合法输入数据，具体使用方法将在下文提及。然而由于某些原因，下发的crack\_force 只能处理**规模较小**的输入数据，并且将会花费较多的时间。但你还原的程序必须在规定时间内解决**规模较大**的问题。

crack\_force从crack\_force.in 读取输入数据，并将输出数据写入到crack\_force.ans。

本题将采用传统题方式进行评测，针对每个子任务均有若干组赛前已经生成的测试点，以检验你还原出的算法的正确性，只有全部通过才能获得该子任务的全部分数。

【输入格式】

输入文件为crack.in。

对于所有数据，输入第一行均为个整数，，表示该组数据所属的子任务编号。

接下来对于每个子任务分别进行描述（描述中出现的变量名均按照小写字母的顺序依次标识）：

* 子任务1\2
  + 共一行包含个整数。
* 子任务3
  + 第一行包含个整数。
  + 第二行包含个整数，。
  + 接下来行每行包含个整数，。
* 子任务4
  + 第一行包含个整数。
  + 第二行包含个整数，。
  + 接下来行每行包含个整数，。
* 子任务5/6
  + 共一行包含个整数。
  + 接下来 行每行包含个整数，; ，数据满足：
    - 存在序列满足：
      * 存在满足或
* 子任务7/8
  + 共一行包含个整数。
  + 接下来行每行包含个整数，; ，数据满足：
    - 存在序列满足：
      * 存在满足或
* 子任务9/10
  + 共一行包含一个由小写字母组成的字符串 。

【输出格式】

输出文件为crack***.out***。

* 子任务 1/2/5/6/7/8/9/10
  + 共一行包含一个整数, 。
* 子任务 3/4
  + 共行每行包含一个整数, 。

【子任务】

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |
|  |  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |
|  |  |  |  |
|  |  |
|  |  |  |  |
|  |  |