NOIP友谊赛

2012年6月

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 题目名称 | 塔 | 长 | 跳 | 恶魔城 |
| 目录 | tower | chan | jump | maze |
| 可执行文件名 | tower | chan | jump | maze |
| 输入文件名 | tower.in | chan.in | jump.in | maze.in |
| 输出文件名 | tower.out | chan.out | jump.out | maze.out |
| 每个测试点时限 | 1秒 | 1秒 | 1秒 | 1秒 |
| 测试点数目 | 10 | 10 | 10 | 10 |
| 每个测试点分值 | 10 | 10 | 10 | 10 |
| 是否有部分分 | 无 | 无 | 无 | 无 |
| 题目类型 | 传统 | 传统 | 传统 | 传统 |

提交源程序须加后缀

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对于Pascal语言 | tower.pas | chan.pas | jump.pas | maze.pas |
| 对于C 语言 | tower.c | chan.c | jump.c | maze.c |
| 对于C++ 语言 | tower.cpp | chan.cpp | jump.cpp | maze.cpp |

**提示：最终测试时，所有编译命令均不打开任何优化开关**

**塔**

## 【问题描述】

敌方的*N*个哨塔整齐地排成一行，并且间距相等均为1，每个哨塔都有一定的生命值*D*。而我方的大炮的攻击范围为*M*，攻击力为*K*，即我方发射一枚炮弹就能够打到连续的*M*个哨塔。假如哨塔被打到时的生命值为*Di*，那么被打后它的生命值就会降为***Di* / *K***(取下整)，生命值降到0哨塔即被炸毁。如果发射一枚炮弹的代价为*P*，那么炸毁所有哨塔的最小代价是多少呢？

## 【输入文件】tower.in

第一行四个正整数*N*，*M*，*K*，*P*，分别表示哨塔个数、炮弹攻击范围、炮弹攻击力以及一枚炮弹的代价。

第二行*N*个正整数，表示每个哨塔的初始生命值。

## 【输出文件】tower.out

只有一个整数，炸毁所有哨塔的最小代价。

## 【样例输入】

8 5 3 10

3 3 3 5 5 1 1 2

## 【样例输出】

30

## 【数据规模和约定】

30%的数据满足 *N* ≤ 10；

60%的数据满足 *N* ≤ 1000；

100%的数据满足0 < *M* ≤ *N* ≤ 100000，1 ≤ *Di*，*P* ≤ 32767，2 ≤ *K* ≤ 32。

**长**

## 【问题描述】

给出序列*A*1，*A*2，…，*A*N，统计元素*Ai*的数量，要求删除元素*Ai*后，序列的最长上升子序列长度减小。

## 【输入文件】chan.in

第一行为一个整数*N*。

第二行*N*个整数*A*1，*A*2，…，*AN*。

## 【输出文件】chan.out

输出一个整数满足条件的元素个数。

## 【输入样例】

4

1 2 2 3

## 【输出样例】

2

## 【样例解释】

原序列的最长上升子序列长度为3，删除*A*1或*A*4能使序列的最长上升子序列长度减小为2。

## 【数据规模和约定】

40%的数据满足 *N* ≤ 100；

70%的数据满足 *N* ≤ 1000；

100%的数据满足0 < *N* ≤ 100000，1 ≤ *Ai* ≤ 1000000000。

**跳**

## 【问题描述】

有一个很大湖面上有*N*片荷叶，给出他们的坐标(两两不同),一只青蛙在这*N*片荷叶之间跳。从坐标为(*x*, *y*)的荷叶他可以跳到:

1. 方向A: (*x* + *P*, *y* + *P*),其中*P*属于正整数集。
2. 方向B: (*x* + *P*, *y* - *P*),其中*P*属于正整数集。
3. 方向C: (*x* - *P*, *y* + *P*),其中*P*属于正整数集。
4. 方向D: (*x* - *P*, *y* - *P*),其中*P*属于正整数集。

每一次，青蛙会选择其中一个方向跳到离他最近的荷叶上，**如果这个方向上没有荷叶，此次跳跃就原地不动。**在一次跳跃后，若青蛙没有原地不动，则原来青蛙所在的荷叶**立即消失**。初始时青蛙在1号荷叶上。

现在我们知道所有荷叶的坐标以及青蛙*K*次跳跃的计划。求*K*次之后青蛙所在的荷叶的坐标。

## 【输入文件】jump.in

输入文件jump.in第一行为两个正整数*N*,*K*。

第二行*K*个字符表示*K*次跳越的方向。

接下来*N*行*N*对整数表示荷叶坐标。

## 【输出文件】jump.out

输出文件名为jump.out。输出两个整数表示青蛙最终所在位置的坐标。。

## 【输入输出样例】

|  |  |
| --- | --- |
| 样例输入1 | 样例输出1 |
| 7 5  ACDBB  5 6  8 9  4 13  1 10  7 4  10 9  3 7 | 7 4 |
| 样例输入2 | 样例输出2 |
| 6 12  AAAAAABCCCDD  1 1  2 2  3 3  4 4  5 3  6 2 | 5 3 |

## 【数据规模和约定】

对于40%的数据，1 ≤ *N* ，*K* ≤ 100，所有坐标在[0, 1000]；

对于100%的数据，1 ≤ *N* ，*K* ≤ 100000，所有坐标在[0, 1000000000]。

**恶魔城**

## 【问题描述】

在最近版恶魔城游戏通关后会随机进入一个无穷大的迷宫，这个迷宫和平时我们常见到的迷宫一样，由一行行同样大的小格子构成。迷宫中的很多小格子中有一定数量的金币，没有金币的格子就是不可穿越的墙。Barty每次玩到这时，总是拼命的捡金币，可每次都没有别人捡的多。

现在请你来帮帮Barty，看一看他是真的很笨还是本来就没有很多金币能捡。当然，Barty已经很细心的研究过了，每次进到的迷宫都是由*N*\**M*的原始地图拼起来的，比如坐标(*x*, *y*)和(*x* + *aN*, *y* + *bM*)的格子相同。坐标可以为负数。原始地图在平面中的位置是(0, 0)到(*N* - 1, *M* - 1)。一个格子的金币捡完了就不会再有了，Barty的初始位置是(*Sx*, *Sy*)，每次可以选择上下左右四个方向进行移动。Barty想尽量多的获得金币，问最多能够拿到多少枚金币。注意，因为地图是无限拼接的，因此有可能会拿到无穷多枚金币哦。

## 【输入文件】maze.in

第一行两个整数*N*，*M*表原始地图为*N*行*M*列。

接下来*N*行每行*M*个字符表示初始地图，对于每个字符，'#'表示障碍物，'1'..'9'表示该位置有1..9个金币。

第*N* + 2行一个整数*Q*表示询问的次数。接下来*Q*行每行两个整数*Sx*， *Sy*，表示一组起点。

## 【输出文件】maze.out

输出*Q*行，对于每组询问分别输出解。如果解为正无穷则输出inf。

## 【输入样例1】

5 4

##1#

##1#

#11#

#1##

#11#

1

2 2

## 【输出样例1】

inf

## 【输入样例2】

5 4

##1#

##2#

#43#

65#7

#8##

1

2 2

## 【输出样例2】

36

## 【数据规模和约定】

30%的数据满足 *N*，*M* ≤ 10；

100%的数据满足 1 ≤ *N*，*M* ≤ 500，-231 ≤ *Sx*，*Sy* ≤ 231-1，1 ≤ *Q* ≤ 3；

保证至少有10%的数据不会出现inf的情况

保证每个位置只可能是’#’,’1’...’9’这些字符

保证结果在32位有符号整型(int)范围内

Hint:地图在平面的拼接方式请参照windows桌面的”平铺”方式