South African Computer Olympiad Training DAY 1

Overview

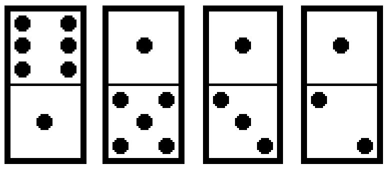
Author	Richard	Nick	Linsen	Marco
Problem	**Dominoes**	**Evacuate**	**Tracks**	**Delivery**
Executable	`dom.exe`	`evac.exe`	`tracks.exe`	`delivery.exe`
Source	`dom.pas` `dom.cpp`	`evac.pas` `evac.cpp`	`tracks.pas` `tracks.cpp`	`delivery.pas` `delivery.cpp`
Input file	`dom.in`	`evac.in`	`tracks.in`	`delivery.in`
Output files(10)	`dom.out`	`evac.out`	`tracks.out`	`delivery.out`
Time limit	1 second	1 second	1.5 seconds	1 second
Num. of tests	10	10	10	10
Points per test	10	10	10	10
Total points	**100**	**100**	**100**	**100**

The maximum total score for Day1 is 300 points.

Dominoes

Author
Richard Starfield (from CEOI)

Introduction
A domino is a flat, thumb sized tile, the face of which is divided into two squares, each left blank or bearing from one to six dots. There is a row of dominoes laid out on a table:



The number of dots in the top line is 6+1+1+1=9 and the number of dots in the bottom line is 1+5+3+2=11. The gap between the top line and the bottom line is 2. The gap is the absolute value of difference between two sums.

Each domino can be turned by 180 degrees keeping its face always upwards.

Task
Compute the smallest number of turns necessary to minimise the gap between the top and bottom lines, and output this number.

Example
For the figure above it is sufficient to turn the last domino in the row in order to decrease the gap to 0. In this case the answer is 1.

Input (dom.in)
The first line contains an integer n. This is the number of dominoes on the table.
The next n lines each contain two integers a and b separated by a single space. The integers a and b written in the line i + 1 of the input file, $1 \le i \le n$, are the numbers of dots on the i-th domino in the row, respectively, in the top line and in the bottom one.

Sample input:
```
4
6 1
1 5
1 3
1 2
```

Output (dom.out)
Your program should output exactly one integer: the smallest number of turns needed to minimise the gap between the top line and the bottom line.

Sample output:
```
1
```

Constraints
$1 \le n \le 1000$
$0 \le a, b \le 6$

Time limit
1 second.

Scoring
You will score 10 if your answer is correct, 0 otherwise.

Evacuate

Author

Nick Pilkington

Introduction

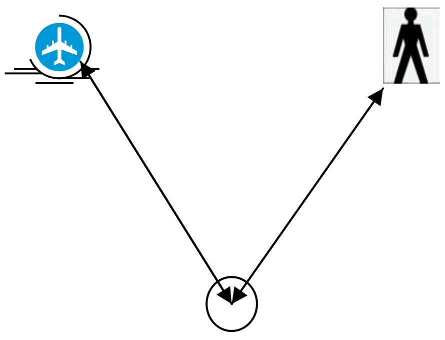
A number of delegates from different countries have met in the capital of a hostile country to negotiate peace. The delegates are all staying at different hotels dotted around the city. The situation has turned hostile and now the delegates are in danger and need to be evacuated from the country as soon as possible. The only way of exiting the country is via one of the airports. There are numerous road blocks on the surface so the delegates must use the routable underground to evacuate. It is assumed that each subway train can depart as soon as necessary. You are to find the shortest amount of time that it will take each delegate to reach an airport. There is a subway station at each hotel. There is a subway station at each airport. All delegates leave there hotels to evacuate concurrently and their transport arrangements don't interfere with each others.

Task

You will be told at which station each delegate is initially situated and which stations contain airports. You will also be told the time in minutes to travel between the stations, 0 minutes indicating there is no direct route between the two stations. It is then your task to find the shortest amount of time required to evacuate each delegate to safety. It is always possible for each delegate to reach an airport.

Example

Consider the following city plan.



The locations of the airport and the delegate are represented by their corresponding symbols.

Input (evac.in)

Line 1: N The number of subway terminals.
Line 2.. N+1 each line contains N+1 Integers. The first number is a value either 0,1 or 2 which indicates a normal station, a station with an airport or a station with a delegate at it respectively. The next N integers T represent the amount of time required to travel from the current station to this one. 0 indicates that the station is not reachable directly from here. Note that the time taken to travel from one station to another may not necessarily be the same at traveling the other way.

Sample Input:

```
3
1 0 2 0
0 2 0 7
2 0 3 0
```

Output (evac.out)

The output should contain D lines, where D is the number of delegates detected each with one number on it. The number should indicate the minimum number of minutes required to evacuate this delegate to safety. The distances should appear in the same order that their respective delegates appear in the input file. eg. Line 1 should represent the shortest distance for the first delegate encountered in the input file. Etc.

Sample output:

```
5
```

Constraints

$2 \leq N \leq 250$
$1 \leq D < N$
$0 \leq T \leq 100{,}000$

Time limit

1 second.

Scoring

If the output is in any way invalid you will score 0 and have the guilt of D civilian lives on your conscience. If the output reflects the optimal evacuation time for each delegate you will be awarded 100% and be heralded as a national hero!

Tracks

Author

Linsen Loots (Inspired by USACO, 1999)

Introduction

The friendly ants have recently started a new colony, and have been collecting food by scouring the floor of their "patron's" kitchen. Unfortunately, this is not appreciated, and he has begun observing their efforts. Luckily, so long as they do not use the same path twice, they will be able to keep getting food safely. Because the ants are very organised, they would like to know how long they will safely be able to use their current location (ie. Before they need to move or use a path twice).

Task

You will be given the size of the rectangular kitchen, and the location of any obstacles, like tables or chairs, (these will also be rectangular), and must compute the number of possible different paths from a given starting point to a given end point. Each open square must be covered exactly once in a path.

Example

If the kitchen is a 5 by 4 room as follows:

	1	2	3	4	5
A	.	.	.	.	.
B	.	*	*	.	.
C	.	.	.	.	e
D	s	.	.	.	.

with 's' representing the starting square and 'e' representing the ending square, '.' Being an open square and the *'s representing obstacles.

There are 3 possible paths for this arrangement:

D1, C1, B1, A1, A2, A3, A4, A5, B5, B4, C4, C3, C2, D2, D3, D4, D5, C5
D1, D2, C2, C1, B1, A1, A2, A3, A4, A5, B5, B4, C4, C3, D3, D4, D5, C5
D1, D2, D3, C3, C2, C1, B1, A1, A2, A3, A4, A5, B5, B4, C4, D4, D5, C5

Input (tracks.in)

The first line will contain 2 single-space separated integers, R and C, the number of rows and columns of the kitchen.

The second line will contain 2 integers representing the co-ordinates of the starting point. Co-ordinates are numbered starting with (1,1) in the top left corner, giving the row then the column.

The third line will contain 2 integers for the co-ordinates of the ending point.

The fourth line will contain an integer N, the number of obstacles in the kitchen

The next N lines will each contain 4 space-separated integers, the co-ordinates of the top left and bottom right corners of the each obstacles. Obstacles will never overlap.

Sample Input:

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

Output (tracks.out)

The output should be a single integer, P, the number of valid paths from the starting point to the end point.

Sample output:

```
3
```

Constraints

$R \leq 25$
$C \leq 25$
$N \leq 10$
The total number of open squares will be smaller than 75, and the total number of paths will fit in a signed 32-bit integer.

Time limit

1.5 seconds.

Scoring

10 marks for the correct answer, 0 for an incorrect answer.

Delivery

Author

Marco Gallotta

Introduction

The Guji tribe has been delivering packages since the beginning of their existence. All the time they have been delivering very inefficiently. Now that we are in the 21st century, the members of the Guji tribe are getting very angry, since other delivery systems are a lot more efficient. They want their delivery system to be as efficient as possible by delivering as many packages as fast as possible.

Task

There are a large number of packages that need to be delivered, within a certain time limit if possible. There are two vehicles available for making the deliveries, a small van and a large lorry. Each package is either *small* or *large*. A small package can be delivered either with the van or with the lorry, but a big one requires the lorry. For each package we know the amount of time it would take to deliver it; we call this the *time requirement* of the package and it includes the time it takes to get the delivery vehicle to deliver the package and return. One vehicle can deliver only one package at a time. We are also given a *deadline*, which is the total amount of time within which we should deliver as many packages as possible. We do not care about what happens after the deadline.

Thus, from our point of view, the deliveries take place as follows.

- A certain number of small packages will be delivered by the van, and the total time requirement of these packages must not exceed the deadline, and
- A certain number of small and large packages will be delivered by the lorry, and the total time requirement of these packages must also not exceed the deadline.

The task in this problem is to determine the largest total number of packages that can be delivered in this manner.

Example

The deadline is 10 time units. There are eight small packages: five with time requirement 2 and three with time requirement 4. There are four large packages, two with time requirement 3 and two with time requirement 6.

Eight of the twelve packages can be delivered in 10 time units by delivering the small packages with time requirement 2 with the van, and using the lorry to deliver one small package with time requirement 4 and two large packages with time requirement 3.

Input (delivery.in)

The first line contains a single integer *T*, the deadline. The second line contains a single integer *N*, the total number of small packages. The next *N* lines each contain a single integer, the time requirement of a small package. The next line contains a single integer *M*, the total number of large packages. The next *M* lines each contain a single integer, the time requirement of a large package.

Sample Input:

```
10
8
2
4
4
2
2
4
2
2
4
6
3
3
6
```

Output (delivery.out)

The only line of the output file should contain a single integer, the largest number of packages that can be delivered within the deadline.

Sample output:

```
8
```

Constraints

- $1 \leq T \leq 10000$
- $1 \leq N, M \leq 500$
- $1 \leq$ Time requirement of a package ≤ 1000

Time limit

1 second.

Scoring

- You will score 0% for an incorrect answer and 100% for a correct answer.