Sicily **Graph**
Contest ends in 1 days 13 hours

cupcup
You havn't any signature yet.
Logout

Home Problems My Status Standing Questions Clarifications Back <->

1009. Knight Moves

Total: 160 Accepted: 59

Time Limit: 1sec Memory Limit:32MB

Description

A friend of you is doing research on the Traveling Knight Problem (TKP) where you are to find the shortest closed tour of knight moves that visits each square of a given set of n squares on a chessboard exactly once. He thinks that the most difficult part of the problem is determining the smallest number of knight moves between two given squares and that, once you have accomplished this, finding the tour would be easy. Of course you know that it is vice versa. So you offer him to write a program that solves the "difficult" part. Your job is to write a program that takes two squares a and b as input and then determines the number of knight moves on a shortest route from a to b.

Input

There are multiple test cases. The first line contains an integer T, indicating the number of test cases. Each test case consists of one line containing two squares separated by one space. A square is a string consisting of a letter (a-h) representing the column and a digit (1-8) representing the row on the chessboard.

Output

For each test case, print one line saying "To get from xx to yy takes n knight moves.".

Sample Input

Copy

```
8
e2 e4
a1 b2
b2 c3
a1 h8
a1 h7
h8 a1
b1 c3
f6 f6
```

Sample Output

Copy

```
To get from e2 to e4 takes 2 knight moves.
To get from a1 to b2 takes 4 knight moves.
To get from b2 to c3 takes 2 knight moves.
To get from a1 to h8 takes 6 knight moves.
To get from a1 to h7 takes 5 knight moves.
To get from h8 to a1 takes 6 knight moves.
To get from b1 to c3 takes 1 knight moves.
To get from f6 to f6 takes 0 knight moves.
```

Hint

This is a typical BFS problem. The graph has 8*8 vertices: (1,1),(1,2), ...,(8,8). Two vertices (m1,n1) and (m2,n2) are connected by an edge if the knight can move from (m1,n1) to (m2,n2). Then for a given starting position, you simply do a BFS from the position. When you have arrived at the given destination, you are done.

[MathJax]/extensions/MathZoom.js加载中

1. Get the adjacency lists for the graph;

2. Implelment BFS, and in the algorithm when you push some vertex u into the queue, you will also remember the smallest number of steps from the starting vertex to u and push this number together with u into the queue.

The solution above is a nice exercise to get you familar with graph representations and BFS.

However, you don't have to constuct the adjacency lists explicitly. Because when you are doing BFS, you will be able to compute the adjacent vertices of a vertex (m,n) by just looking at the vertex itself.

Some other possible solution: computer a 64*64 lookup table where an (i,j) element is the smallest number of steps from vertex i to vertex j. And when you get the input data, you just look it up in the table. This is the all-pairs shortest path problem.

As the table is symmetric, you only need to remember half of the table, and thus save half of the memory.

Submit

[MathJax]/extensions/MathZoom.js加載中