**Snakes and Ladders Game**

Vivek takes out his Snakes and Ladders game and wonders: If he had absolute control over the die (singular), and could get it to generate any number (in the range 1-6) he desired, what would be the least number of rolls of the die in which he'd be able to reach the destination square (Square Number 100) after having started at the base square (Square Number 1)?

**Rules**

1. Vivek has total control over the die and the face which shows up every time he tosses it.
2. A die roll that causes the player to land up at a square greater than 100 goes wasted, and the player remains at his original square. For example, when the player is at Square Number 99, rolls the die, and ends up with a 5.
3. If a person reaches a square which is the base of a ladder, they have to climb up that ladder. If a person reaches a square which has the mouth of the snake, they have to go down the snake and come out through the tail.

**Input Format**

The first line contains the number of tests, T. T test cases follow. For each test case:

* The first line contains N (Number of ladders) and after that N lines follow. Each of the N lines contains 2 integers representing the starting point and the ending point of a ladder respectively.
* The next line contains integer M (Number of snakes) and after that M lines follow. Each of the M lines contains 2 integers representing the starting point and the ending point of a snake respectively.

**Constraints**

* The board is always of the size 10 x 10 and squares are always numbered 1 to 100.
* 1 <= T <= 10
* 1 <= N <= 15
* 1 <= M <= 15
* Square numbers 1 and 100 will not be the starting point of a ladder or a snake.
* No square will have more than one starting or ending point of a snake or ladder.

**Output Format**

For each of the T test cases, output one integer in a new line, which is the least number of moves (or rolls of the die) in which the player can reach the target square (Square Number 100) after starting at the base (Square Number 1). If there is no solution, print -1.

**Sample Input**

2

3

32 62

42 68

12 98

7

95 13

97 25

93 37

79 27

75 19

49 47

67 17

4

8 52

6 80

26 42

2 72

9

51 19

39 11

37 29

81 3

59 5

79 23

53 7

43 33

77 21

**Sample Output**

3

5

**Explanation**

For the first test: To traverse the board via the shortest route, the player first rolls the die to get a 5 and ends up at square 6. He then rolls the die to get 6 and ends up at square 12, from where he climbs the ladder to square 98. He then rolls the die to get '2' and ends up at square 100, which is the target square. So, the player required 3 rolls of the die for this shortest and best case scenario. So the answer for the first test is 3.

**Solution Code**

python

Copy code

from collections import deque

def min\_dice\_throws(n, ladders, snakes):

board = list(range(101))

# Place ladders and snakes on the board

for start, end in ladders:

board[start] = end

for start, end in snakes:

board[start] = end

visited = [False] \* 101

queue = deque([(1, 0)]) # (current square, number of rolls)

visited[1] = True

while queue:

current, rolls = queue.popleft()

if current == 100:

return rolls

for dice in range(1, 7):

next\_square = current + dice

if next\_square <= 100 and not visited[next\_square]:

visited[next\_square] = True

queue.append((board[next\_square], rolls + 1))

return -1

# Input reading

t = int(input())

for \_ in range(t):

n = int(input())

ladders = [tuple(map(int, input().split())) for \_ in range(n)]

m = int(input())

snakes = [tuple(map(int, input().split())) for \_ in range(m)]

# Calculate and print the minimum number of rolls

print(min\_dice\_throws(n, ladders, snakes))