<u>CSE422 Lab 01: A* Search</u>

<u>Part 1</u>

Description:
Task: You are to write the program of a Maze Solver Agent to find a path through a maze from a entry point to an exit point using the **A* search** algorithm.

Maze: For this task, the maze is of size **N x M**, where N is the height and M is the width of the maze. The maze is described using 2 symbols, **“0” and “#”**, where “0”s denote the paths and “#”s denote the walls. In addition, the mazes will have only one entry point and only one exit point. An example maze is as follows:

```
# # # 0 # # # #
# 0 0 0 0 0 0 #
# # # 0 # 0 0 #
# 0 0 0 # 0 0 #
# 0 # # # # # #
# 0 0 0 0 0 0 #
# # # # # 0 # #
```

Fig: A maze of size 7 x 8, where (0, 3) and (6, 5) indices are the entry and exit points, respectively.

Heuristic function: You need to use the **Manhattan distance** as the heuristic function. For two points (x_1, y_1) and (x_2, y_2), the Manhattan distance is calculated as $|x_1 - x_2| + |y_1 - y_2|$.

Moves: The agent can move only in **horizontal and vertical** directions. It cannot move in diagonal directions.

Cost: The cost of each move in the maze is **1**.

Input:
The first line contains two integers n, m – height and width of the maze.
The second line contains two integers a, b – index of the entry point of the maze.
The third line contains two integers c, d – index of the exit point of the maze.
The following n lines contain m characters – description of the maze.

Output:
The first output will be the total path cost (number of steps required to exit the maze).
The second output will be the sequence of actions to go from initial to goal state [it will contain a sequence of actions represented using the characters U (Up), D (Down), R (Right), and L (Left) Print -1 if no path is found].

Sample:

Input	Output
7 8 0 3 6 5 ###0#### #000000# ###0#00# #000#00# #0###### #000000# #####0##	12 DDDLLDDRRRRD
10 12 0 1 9 8 #0########## #000000###0# ###0###0##0# #0#0#000##0# #0#0###0##0# #0000#00##0# ###0###0##0# #000#0000#0# #0#####0000# #######0####	-1
15 15 1 0 13 14 ############### 0000#########0# ###0#####00##0# #00000000#0##0# #0######0####0# #0######0#0##0# #0000000000000# ##########0#### #0######000####	28 RRRDDRRRRRDDDRRDDDRRDDLDDRRR

#0######000000# #0##########00# #0000000000000# ###0#######0### #00000000000000 ###############	

Part 2

Write a program to check whether heuristic values are admissible or not.

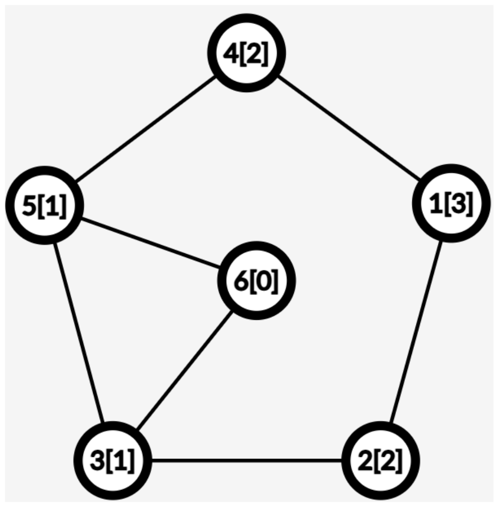
Admissibility checker: You need to write a function that takes a graph and a set of heuristic values as input and outputs whether the heuristic values are admissible or not.

Graph: The graph is an **undirected and unweighted graph** with **N** vertices numbered from **1 to N**.

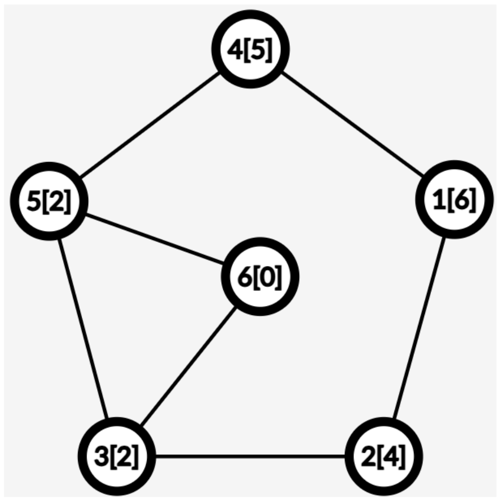
Input:
The first line contains two integers n, m – number of vertices and edges of the graph.
The second line contains two integers a, b – initial vertex and goal vertex.
Each of the following n lines contains two integers x, y – denoting that vertex x has a heuristic value of y. Each of the next m lines contains two integers u, v – denoting that an edge connects vertices u and v.
Output: Print 1 if all the heuristic values are admissible. **Otherwise print 0 and mention which nodes are not admissible.**

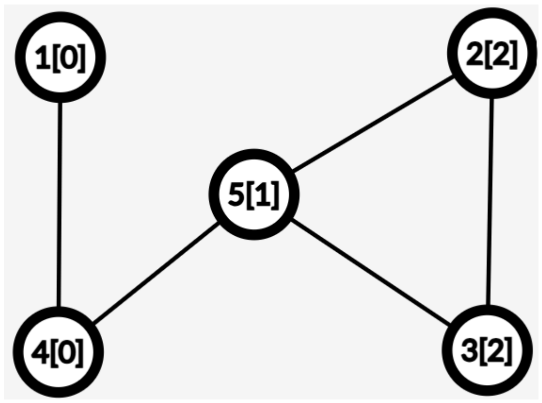
Sample:

Input	Graph	Output

6 7
 	
1 6
 	
1 3
2 2
3 1
4 2
5 1
6 0
 	
1 2
2 3
3 6
1 4
4 5
		1

5 6 3 5		

6 7
 	
1 6
 	
1 6
2 4
3 2
4 5
5 2
6 0
 	
1 2
2 3
3 6
1 4
4 5
5 6
3 5
		0
		
		Here nodes 1, 2, 3, 4 and 5 are inadmissible.

5 5
 	
2 4
 	
1 0
2 2
3 2
4 0
5 1
 	
5 2
2 3
1 4
4 5
5 3
		1