CS29003 ALGORITHMS LABORATORY
Assignment 3 : Graphs
Date: 24 – Sept – 2020

James is a frog who is currently sitting on a rock. The rocks around him are arranged in a grid of size $A \times B$, with co-ordinates ranging from $(0, 0)$ to $(A - 1, B - 1)$. James is sitting at (Jx, Jy). James can jump on adjacent rocks which are to the North, East, West or South of it's current rock. Some rocks are unstable and hence he cannot land over those rocks. A fly is sitting on a rock which is at the coordinate (Fx, Fy). James has to reach the rock on which the fly is sitting (*it is assured that the rock on which James / fly is sitting is stable*), by jumping on only the stable rocks. Is it possible? You are supposed to use graph traversal strategies to solve this problem.

Strategy 1

Use of Queues: You can use queues to travel the rocks in a BFS manner. BFS first explores all the immediate neighbouring rocks of a rock and then continues to explore the immediate neighbours of previously explored rocks. Initially enqueue (Jx, Jy) to an empty queue. As long as queue is not empty, dequeue a rock (x, y) from the queue. Look at all the neighbours of (x, y) and for all the neighbouring rocks which are stable and unvisited, enqueue the rocks to the queue. If the search stops (that is, the queue becomes empty) without ever having (Fx, Fy) at the front of the queue, report failure.

Strategy 2

Use of Stacks: This strategy employs stacks to explore the rocks by going as far as possible jumping on stable rocks using DFS. You are required to use a stack for the corresponding implementation. For each of the stable unvisited neighbour (u, v) of rock (i, j), you have to push the neighbouring rock on the top of the stack. When there are no unvisited and stable neighbouring rocks left, pop (i, j) off the stack. If at any point of the algorithm, you find the fly, i.e., you reach (Fx, Fy), then there exists a path from James to the fly. Print this path by popping the nodes from the stack one by one. If the search completes (that is, the stack becomes empty) without ever having (Fx, Fy) in the stack, then there is no path.

Input

First line of input contains integers A, B denoting the size of the grid and N denoting number of unstable rocks.
Second line contains N integers x_i, $i \in [0, A - 1]$ denoting the X-coordinates of unstable rocks.
Third line contains N integers y_i, $i \in [0, B - 1]$ denoting the Y-coordinates of unstable rocks.
Fourth line contains two pairs of integers (Jx, Jy) and (Fx, Fy), representing positions of James and fly, respectively.

$$0 < A, B, N < 200$$

$$0 \leqslant x_i, Jx, Fx < A, \qquad 0 \leqslant y_i, Jy, Fy < B$$

Part 1

Implementing the data structure to represent the rocks and printing the initial state of the system.

You can use a 2-d static array to store the information about rocks. Then print the structure as shown in the sample output.

void printgrid(...)

Inputs: Grid information

Output: Printing the grid in the format as seen in **sample input and output**.

Note: You don't need to construct an explicit graph to represent this data.

Part 2

Implementing queue

The queue must be implemented as array of fixed size which is provided as input while creating the array (If you are using an older c compiler, use a fixed size array of hardcoded size QUEUE_SIZE_MAX). Keep two indices denoting the start and end of the queue. Implement the following functions:

```
typedef struct {
   int x, y;
} POINT ;
typedef struct {
    POINT *Arr ;
    int queue_size, start_id, end_id;
} QUEUE ;
void init(QUEUE *qP, int size);//allocates space for queue
int isempty(QUEUE qP);//returns 1 if the queue is empty, else 0
void enqueue(QUEUE *qP, POINT p);
POINT dequeue(QUEUE *qP);
```

Part 3

Implement search using Strategy 1

int strategy1(...)

Inputs: State of rocks and position of fly

Output: If any path exists, it prints "Path Exists" and returns 1. If path does not exist, it prints "No Path Exists" and returns 0.

Part 4

Implementing stack

Stack must be implemented using a linked list. Head of linked list points to the top of the stack. For implementing the push operation, you need to insert a new head and to implement the pop operation, you have to delete a head. The following functions must be implemented.

```
typedef struct {
   int x, y;
} POINT ;
typedef struct {
    struct POINT head;
```

```
    struct STACK *next;
} STACK ;
void init(STACK *s); //initializes the head and next pointer
int isempty(STACK s);//returns 1 if the stack is empty, 0 otherwise
void push(STACK *s, POINT p);
POINT pop(STACK *s);
```

Part 5

Implement search using Strategy 2

Use Stack to travel the grid in a DFS fashion

void strategy2(...)

Inputs: State of rocks and position of fly

Output: Print a path of points from James to fly as shown in the sample output

Main Function

The main() function first creates and initializes the data structure required to store the input. It then reads the four lines of input and stores it appropriately. After reading input, it calls the printgrid() function to print the initial grid. This function may also help you while debugging. strategy1() is called with appropriate parameters to find if a path exists from James to the fly. If a path exists, strategy2() is invoked and it prints any path from James to fly consisting only of stable rocks. Note that you cannot go outside of the grid of rocks.

Sample Input and Output

Test Case 1

```
Input
  4 5 6
  0 0 1 2 2 3
  0 4 4 1 2 4
  0 1 3 3

Output
  Grid of Stones is:
  -***-
  ****-
  *--**
  ****-
  Path Exists
  (0, 1), (0, 2), (0, 3), (1, 3), (2, 3), (3, 3)

Explanation
  The printgrid() function first prints 'Grid of Stones is:', then it prints the actual grid.
  '-' indicates an unstable rock and '*' denotes a stable rock. As there exists a path between
  James and the fly, strategy1() prints 'Path Exists'. Then strategy2() prints a possible
  path from (0, 1) to (3, 3).
```

Test Case 2

```
Input
  2 5 3
  0 1 1
  2 0 4
  0 0 0 4

Output
  Grid of Stones is:
  **-**
  -***-
  Path Exists
  (0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (0, 3), (0, 4)
```

Test Case 3

```
Input
  2 2 2
  0 1
  1 0
  0 0 1 1

Output
  Grid of Stones is:
  *-
  -*
  No Path Exists

Explanation
  There is no path from James to fly, hence the strategy1() prints 'No Path Exists' and
  strategy2() is not called from the main() function.
```

Submission

Please note that your submissions will not be evaluated unless you follow the below specified file naming convention for the program file. <ROLLNO(IN CAPS)>_G<Group_No>_Assign<Assign_No>.c/cpp

Eg: 18CS30004_G03_Assign3.c / 18CS30004_G03_Assign3.cpp