Problem Set 4

This problem set is divided into two parts: Part A problems are theory questions, and Part B problems are programming tasks.

Part A questions are due Tuesday, November 3rd at 11:59PM.

Part B questions are due Thursday, November 5th at 11:59PM.

Solutions should be turned in through the course website in PDF form using \LaTeX or scanned handwritten solutions.

A template for writing up solutions in \LaTeX is available on the course website.

Remember, your goal is to communicate. Full credit will be given only to the correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Part A: Due Tuesday, November 3rd

1. (20 points) Reasoning about DFS

   For each of the following statements, prove the statement or give a small counter example to show that it is false. You may use \LaTeX to draw counter-example graphs if necessary (the solution template contains a drawing of the following graph to get you started).

   \begin{center}
   \begin{tikzpicture}
   \foreach \x in {0,1,2}
   \draw (\x,0) node (\x) {};
   \foreach \y in {0,1,2}
   \draw (0,\y) node (\y) {};
   \foreach \x in {0,1,2}
   \foreach \y in {0,1,2}
   \draw (\x,\y) -- (\x,\y + 1);
   \end{tikzpicture}
   \end{center}

   (a) (6 points) If an undirected graph $G$ has no cycles, then DFS always finds the shortest path (i.e. the path with the fewest edges).

   (b) (6 points) If a directed graph $G$ has no cycles, then DFS always finds the shortest path (i.e. the path with the fewest edges).

   (c) (8 points) Suppose that you use both DFS and BFS to find paths between two points in a graph. If DFS returns the shortest path in the graph, then it visited fewer nodes than BFS did.
2. (14 points) Bipartite graphs

An undirected graph is called bipartite if its nodes can each be assigned a color, either red or blue, such that no red node is adjacent to another red node, and no blue node is adjacent to another blue node. Give an efficient algorithm to determine if a graph is bipartite. What is its running time?

3. (16 points) Bus routes

You are traveling a strange country with $N$ cities. You start at city 1 at time 0 and need to get to city $N$ no later than time $T$. There are $M$ busses each of which travels from some city to some other city in exactly 1 hour. (It’s a small country.) Put another way, each bus $i$ leaves some city $a_i$ at a certain time $t_i$ and arrives at another city $b_i$ at time $t_i + 1$. All busses depart and arrive at integer hours. Assume that if you arrive in a city at time $t$ and there is a bus leaving at time $t$ then you can make the transfer.

Given the times of the $M$ busses, propose an algorithm based on BFS for determining whether there is a way to get from city 1 to city $N$ in no more than time $T$. Your algorithm should run in $O(N \cdot T + M)$ time.

Part B: Due Thursday, November 5th

(50 points) Eight Puzzle

The eight puzzle is a game in which 8 tiles numbered 1 through 8 are placed on a $3 \times 3$ board. One of the squares of the board always remains uncovered. A possible configuration is the following:

```
 2 6 4
1 7
8 3 5
```

In each step the player can slide one of the (at most four) tiles adjacent to the empty square into the empty square. The above configuration can for instance be turned into one of the following configurations:

```
 2 6 4  2 6  2 6 4
1 7    1 7 4  1 7 5
8 3 5  8 3 5  8 3 
```

The goal of the game is to turn the initial configuration into the following configuration:

```
1 2 3
4 5 6
7 8
```
Note that that there are $9! = 362,880$ possible different configurations of the board. Not every of them needs be reachable from every other one. Your task in this problem is to write a program that reads a description of the problem from the standard input, and writes a solution to the standard output.

Input Format

**Lines 1...3:** The lines describe initial configuration. The number 0 denotes the blank square.

Sample Input

```
4 1 3
2 6 0
7 5 8
```

Output Format

**Line 1:** The number of connected components in the graph of all configurations. Note that this is independent of the input data.

**Line 2:** The minimum number of steps to solve the puzzle in the input. If the puzzle is not solvable, just output “unsolvable”

Sample Output

```
???
7
```

In the above example, we have hidden the number of components so that you can discover it yourself.