CS 341: Algorithms @ Waterloo **Last Updated:** May 31, 2021
Homework 2 **Due Date: June 14, 2021, 11:00pm**

You are allowed to discuss with others but are not allowed to use any references other than the course notes and the three reference books. Please list your collaborators for each question. You must write your own solutions. See the course outline for the homework policy.

There are totally 52 marks. The full mark is 50. This homework is counted 10% of the course grade.

---

1. **Programming Problem: A Game with Two Tokens**[1] (20 marks)

   Let's consider the following game on an $n \times n$ square grid. Some squares of the grid are marked as *obstacles*, and one grid square is marked as the *target*. There are two tokens – token A and token B. In the beginning, each of them is placed in a distinct square that is neither an obstacle nor the target.

   In each turn, the player must move one of the tokens from its current position *as far as possible* upward, downward, left, or right. That is, the player chooses a token and a direction, and moves the token along that direction, stopping just before it hits (1) the edge of the board, (2) an obstacle square, or (3) the other token. The goal of the game is to move *either* of the tokens to the target square. When the goal is attained, the game finishes. Note that moving past the target square doesn't count.

   Your task is to return a shortest sequence of moves to finish the game, or report that it is impossible.

   Input: The first line of input has a positive integer n, the size of the grid. Each of the next n lines contains n characters, representing the grid. Here's a table of all possible characters and their meaning:

character	meaning
`A`	token A
`B`	token B
`T`	the target
`#`	an obstacle
`.`	an empty square

   It is guaranteed that $2 \leq n \leq 20$, and that the characters `A`, `B`, `T` appear exactly once in the input.

   Output: If there is no solution, output -1 on a single line. Otherwise, on the first line output k, the minimum number of moves to finish the game, followed by k lines representing the moves in order. Each line must be in the form {`A|B`} {`up|down|left|right`}. The sample test case should give you a better idea of the output format.

   Sample Input 1:

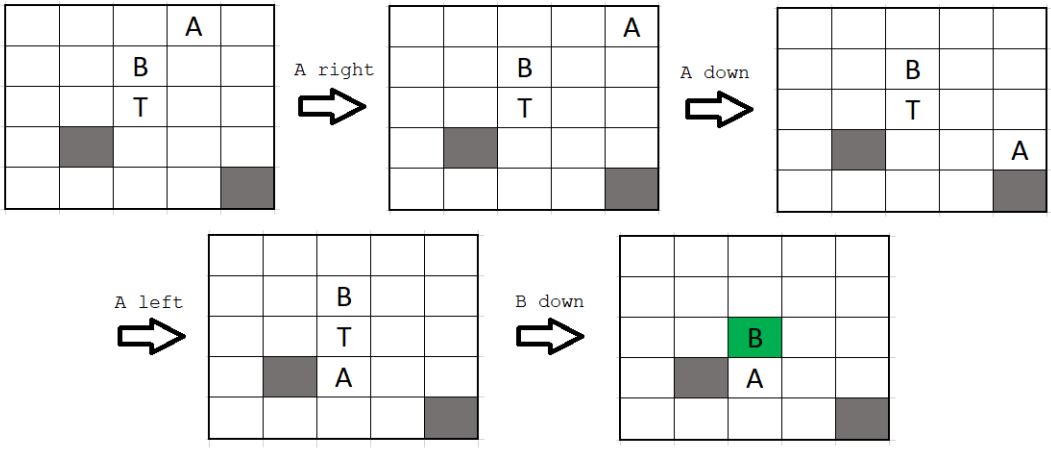
```
5
...A.
..B..
..T..
.#...
....#
```

[1] This problem is from Chapter 5 of the textbook "Algorithms" by Jeff Erickson, available online and under the CC BY 4.0 license. Wording is slightly modified.

Sample Output 1:

```
4
A right
A down
A left
B down
```

Explanation:



Sample Input 2:

```
5
..#A.
..B##
..T..
.#...
....#
```

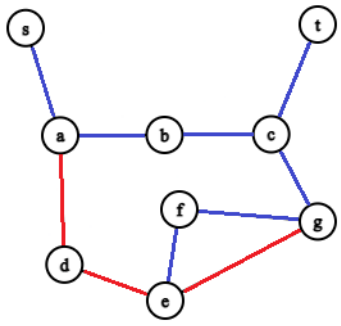
Sample Output 2:

```
-1
```

2. **Written Problem: Red-Blue Paths**[2] (10 marks)

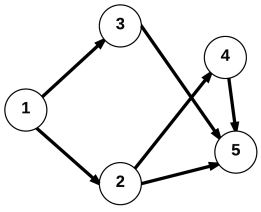
Suppose we are given an undirected graph $G = (V, E)$ where some edges are red and the remaining edges are blue. Describe an algorithm to find a shortest walk in G from one vertex s to another vertex t in which no three consecutive edges have the same color, or report that no such walks exist. That is, if the walk contains two red edges in a row, the next edge must be blue, and if the walk contains two blue edges in a row, the next edge must be red. You will get full marks if the time complexity of the algorithm is $O(|V| + |E|)$ and the proofs are correct.

For example, given the following graph as input, your algorithm should return $(s, a, d, e, f, e, g, c, t)$, which is the shortest legal walk from s to t.

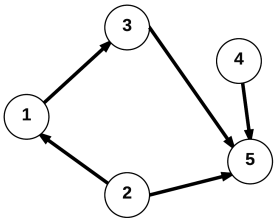


3. **Written Problem: Reachability** (10 marks)

Given a directed graph $G = (V, E)$, design an algorithm to find a vertex $s \in V$ from which all other vertices are reachable (i.e. there is a directed path from s to v for all $v \in V$) or report that no such vertices exist. You will get full marks if the time complexity is $O(|V| + |E|)$ and the proofs are correct.



(a) Vertex 1 can reach all other vertices.



(b) No vertex can reach all other vertices.

[2]This problem is also from Chapter 5 of the textbook "Algorithms" by Jeff Erickson.

4. **Written Problem: One-Way Streets** (12 marks)

Imagine in a city where all the streets are two-way streets. Perhaps because the streets are too narrow, the government would like to see if it is possible to make all streets one way, so that there is still a way to drive from any street to any other street in the city.

Show how to model this problem as a graph problem. Then design an algorithm to determine if it is possible, and when it is possible to output one solution to assign the directions.

You will get full marks if the time complexity of the algorithm is $O(n+m)$ and the proofs are correct, where m is the number of streets in the city and n is the number of intersections in the city.

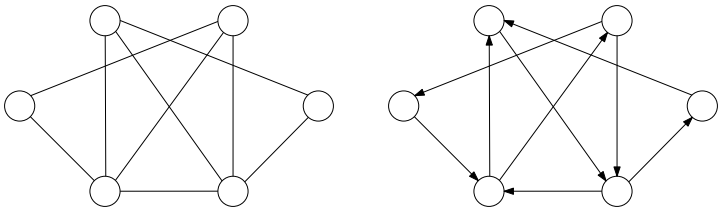


Figure 2: There is a solution in this example.

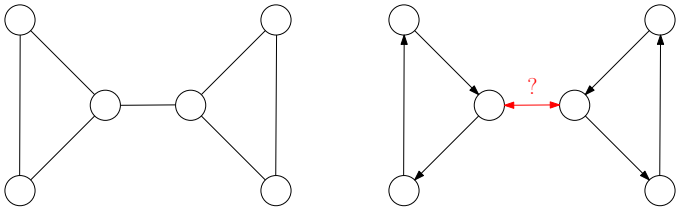


Figure 3: There is no solution in this example.