CS 374 Lab 16: Graph Modelling

Date: March 14, 2018.

For each of the problems below, transform the input into a graph and apply a standard graph algorithm that youve seen in class. Whenever you use a standard graph algorithm, you **must** provide the following information. (I recommend actually using a bulleted list.)

- What are the vertices?
- What are the edges? Are they directed or undirected?
- If the vertices and/or edges have associated values, what are they?
- What problem do you need to solve on this graph?
- What standard algorithm are you using to solve that problem?
- What is the running time of your entire algorithm, *including* the time to build the graph, *as a function of the original input parameters*?

1. Let G be a connected undirected graph. Suppose we start with two coins on two arbitrarily chosen vertices of G. At every step, each coin *must* move to an adjacent vertex. Describe and analyze an algorithm that given G and starting vertices s, t (which may or may not be distinct) correctly decides whether the game can reach a configuration where both coins are on the same vertex. Can you think of an instance where the desired configuration is not reachable?

   Hint: Form a graph, but NOT the one you are given.

2. Let $G = (V, E)$ be an *undirected* graph. Describe a linear-time ($O(m + n)$ time) algorithm that given G finds a cycle in G or reports that there is none. Describe an algorithm that finds 2 distinct cycles in G or reports that G does not have them.

3. **Snakes and Ladders** is a classic board game, originating in India no later than the 16th century. The board consists of an $n \times n$ grid of squares, numbered consecutively from 1 to n^2, starting in the bottom left corner and proceeding row by row from bottom to top, with rows alternating to the left and right. Certain pairs of squares, always in different rows, are connected by either snakes (leading down) or ladders (leading up). Each square can be an endpoint of at most one snake or ladder.

   You start with a token in cell 1, in the bottom left corner. In each move, you advance your token up to k positions, for some fixed constant k (typically 6). If the token ends the move at the *top* end of a snake, you **must** slide the token down to the bottom of that snake. If the token ends the move at the *bottom* end of a ladder, you **may** move the token up to the top of that ladder.

   Describe and analyze an algorithm to compute the smallest number of moves required for the token to reach the last square of the grid.

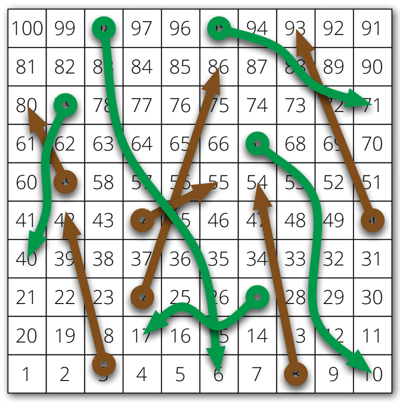

Figure 1: A typical Snakes and Ladders board. Upward straight arrows are ladders; downward wavy arrows are snakes.