Weekplan: Directed Graphs

The 02105+02326 DTU Algorithms Team

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Introduction to Part VI + Chapter 22.1-22.4 + Appendix B.4-B.5.

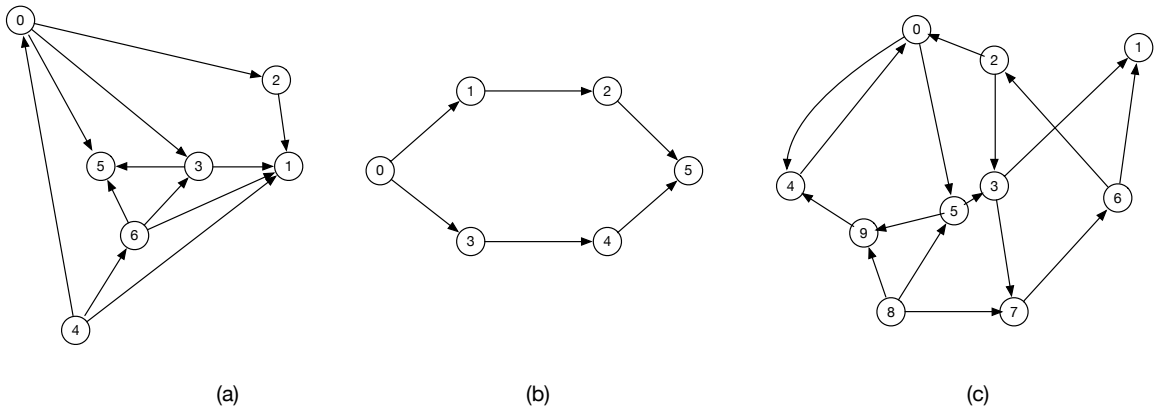


Figure 1: Graphs for the exercises.

Exercises

1 Representation, Properties and Algorithms Consider the graphs in Figure 1. Solve the following exercises.

1.1 [w] Show adjacency lists and adjacency matrices for (a) and (b).

1.2 [w] Run DFS or BFS starting in node 4 in (a) and node 5 in (c) by hand.

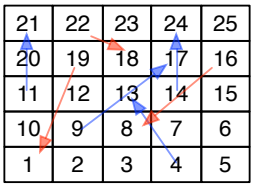
1.3 Which of (a) and (c) are DAGs? If the graph is a DAG find a topological ordering using the recursive algorithm for topological sorting. If the graph is not a DAG find a cycle.

1.4 Specify strongly connected components of (a) and (c).

1.5 How many topological orderings does (b) have?

1.6 How many strongly connected components does a DAG have?

2 Snakes and Ladders Snakes and ladders is a classic board game. We will look at the following variant. The game is played on a $n \times n$ grid with cells numbered from 1 to n^2 in order (see the figure). Special pairs of cells are *snakes* that leads downwards and *ladders* that leads upwards. A cell can be endpoint for at most one ladder or snake.



The goal of the game is to move from cell 1 to cell n^2 in the fewest possible rounds. First, place a piece on cell 1. In each round, you can move the piece *at most* 5 fields forward. If the piece ends in the top of a snake the piece is moved to the bottom of the snake, and, similarly, if the piece ends in the bottom of a ladder it is moved to the top of the ladder.

2.1 Give an algorithm to compute the fewest number of rounds needed to move a piece from cell 1 to cell n^2.

3 DAGs and Topological Sorting

3.1 Professor Gørtz suggests the following new and simple algorithm to construct a topological ordering: run BFS from a node s with in-degree 0 and sort the nodes by increasing distance to s. Does the algorithm work?

3.2 Give an algorithm that given a graph G and an ordering S of the nodes in G determines if S is a topological ordering.

3.3 Given a DAG G, does there exists a topological ordering of G that cannot be produced by the recursive algorithm for topological sorting?

3.4 [$*$] A *Hamiltonian path* is a path that visits all nodes exactly once. Give an algorithm that determines if a DAG has a Hamiltonian path.

4 [†] **BFS Implementation** Let G be a graph given by sorted adjacency lists. Implement BFS. Given a start node s and end end node t determine the *length* of the shortest path between s and t. When there are multiple neighbors to choose from, pick the one with the lowest number.

5 [$*$] **Ethnographers**[1] You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who've lived there over the past two hundred years.

From these interviews, they've learned about a set of n people (all of them now deceased), whom we'll denote $P_1, P_2, \ldots, P_n$. They've also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

(a) For some i and j, person P_i died before P_j was born.

(b) For some i and j, the life spans of P_i and P_j overlapped at least partially.

Naturally, they're not sure that all these facts are correct; memories are not so good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.

Give an efficient algorithm to do this: either it should produce proposed dates of birth and death for each of the n people so that all the facts hold true, or it should report (correctly) that no such dates can exist – that is, the facts collected by the ethnographers are not internally consistent.

6 Three Bottles You are given three bottles with capacity respectively 8, 5 and 3 liters. Initially, the 8 liter bottle is filled with water and the two other are empty. Your target is to have precisely 4 liters of water in one of the bottles. You can pour water from one bottle to another but you must continue until either the bottle you are pouring from is empty, or the one you are pouring to is full.

6.1 [$*$] Show it is possible to do this, and give the shortest sequence of fillings/empties you can find.

6.2 [$*$] Now assume you have n bottles with capacity $d_1, \ldots, d_n$ liters and a target of x liters water in a bottle in the end. Give an algorithm to compute the shortest sequence of fillings/empties. *Hint:* model the problem as an implcit graph.

[1] from *Algorithm Design*, Jon Kleinberg og Eva Tardos