School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 11

14–18 October 2019

Plan

The exam is not far away, so keep up with tutorials; as always, try tackling the problems *before* the tute.

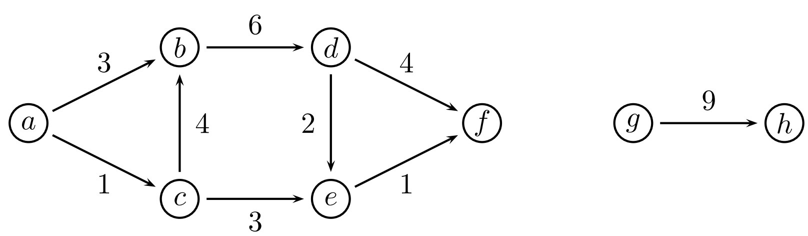
The exercises

74. Use the dynamic-programming algorithm developed in Lecture 18 to solve this instance of the coin-row problem: 20, 50, 20, 5, 10, 20, 5.

75. In Week 12 we will meet the concept of *problem reduction*. This question prepares you for that. First, when we talk about the length of a path in an un-weighted directed acyclic graph (DAG), we mean the number of edges in the path. (You could also consider the un-weighted graph weighted, with all edges having weight 1)

    Show how to reduce the coin-row problem to the problem of finding a longest path in a DAG. That is, give an algorithm that transforms any coin-row instance into a longest-path-in-DAG instance in such as way that a solution to the latter provides a solution to the former. Hint: If there are n coins, use $n + 1$ nodes; let an edge with weight i correspond to picking a coin with value i.

76. Consider the problem of finding the length of a "longest" path in a *weighted*, not necessarily connected, DAG. We assume that all weights are positive, and that a "longest" path is a path whose edge weights add up to the maximal possible value. For example, for the following graph, the longest path is of length 15:



    Use a dynamic programming approach to the problem of finding longest path in a weighted dag.

77. Design a dynamic programming algorithm for the version of the knapsack problem in which there are unlimited numbers of copies of each item. That is, we are given items $I_1, \ldots, I_n$ that have values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$ as usual, but each item I_i can be selected several times. Hint: This actually makes the knapsack problem a bit easier, as there is only one parameter (namely the remaining capacity w) in the recurrence relation.

78. Work through Warshall's algorithm to find the transitive closure of the binary relation given by this table (or directed graph):

	a	b	c	d
a	0	0	1	1
b	0	0	1	0
c	1	0	0	0
d	0	0	0	0