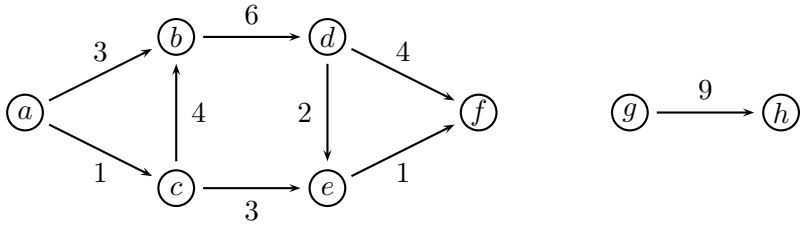
Department of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 12

Plan

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

The exercises

1. 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. (Hint: Start by doing topological sorting.)

2. 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 $I_1, \ldots, I_n$ 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.

3. 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

4. Floyd's algorithm sometimes works even if we allow negative weights in a dag.

   For example, for the left graph above, it will produce these successive distance matrices:

$$D^0 = D^1 = D^2 = \begin{bmatrix} 0 & 4 \\ -3 & 0 \end{bmatrix}$$

   What happens for the right graph above? What do D^0, D^1 and D_2 look like? Explain why D^2 ends up giving an incorrect result in this case (but not in the previous case).

5. We are given a sequence of "connection points" spaced out evenly along a straight line. There are n white, and n black points, in some (random) order.

The points are spaced out evenly, so that the distance between two adjacent points is 1.

The points need to be connected, so that each white point is connected to exactly one black point and vice versa. However, the total length of wire used must be kept as small as possible.

Consider the following (greedy) algorithm to solve the problem:

$k \leftarrow 1$
while there are still unconnected points **do**
    create all possible connections of length k
    $k \leftarrow k + 1$

Argue the correctness of this algorithm, or, alternatively, devise an example that proves that it may not produce an optimal wiring.