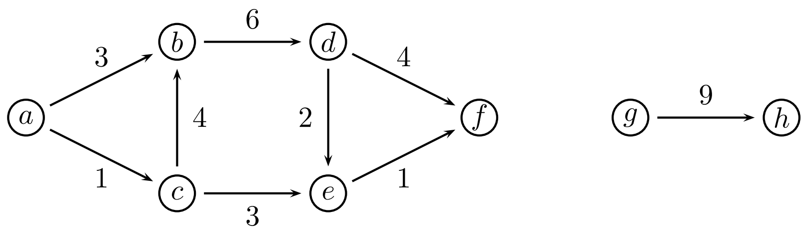
School of Computing and Information Systems
COMP90038 Algorithms and Complexity Tutorial Week 11

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

2. 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:

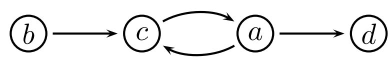


Use a dynamic programming approach to the problem of finding longest path in a weighted dag.

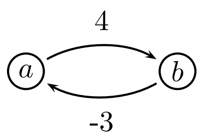
3. Design a dynamic programming algorithm for the version of the knapsack problem in which there are unlimited numbers of copies of each item. That is, we are given items $I_1, \ldots, I_n$ have values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$ as usual, but each item I_i can be selected several 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.

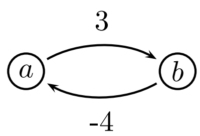
4. Work through Warshall's algorithm to find the transitive closure of the binary relation given by this table (or directed graph):

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



5. 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).