FIT2004: Tutorial 5 (held in Week 11)
Covering graph concepts from Weeks 9 & 10

Objectives: The tutorials, in general, give practice in problem solving, in analysis of algorithms and data-structures, and in mathematics and logic useful in the above.

Instructions to the class: Prepare your answers to the questions **before** the tutorial! It will probably not be possible to cover all questions unless the class has prepared them all in advance.

Instructions to Tutors:

1. The purpose of the tutorials is not to solve the practical exercises!
2. The purpose is to check answers, and to discuss particular sticking points, not to simply make answers available.

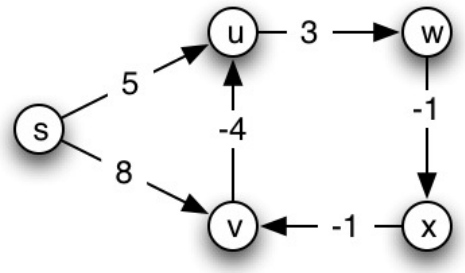
∗ **1.** In week 9 lectures we discussed about the *transitive closure* of a graph. Revise what this term means. We said, if A is the adjacency matrix of some graph $G(V, E)$, then exponentiating this matrix by an integer k, i.e., A^k, $1 \leq k \leq |V|$ yields its transitive closure.

Can you confirm this proposition by considering a small graph (say containing 4 to 5 vertices, both directed and undirected), and by working out by hand the matrices: A, A^2, A^3, $\cdots$ of this graph? (On the other hand, you can write a quick python program to do this as well.)

2. **a)** Revise the Bellman-Ford's algorithm from Week 9 lecture.

b) On lines 16-22 on slide 11, you will notice that Bellman-Ford's algorithm has a postprocessing step to ensure that no negative-weight-cycle was encountered along a path to some vertex in the graph from the source. The criteria used to detect such cycles is given on line 19. Reason why this criteria is correct.

c) Work out the algorithm on the following graph by filling the table with your calculations as you proceed through the algorithm.



	i=0	i=1	i=2	i=3	i=4
s	0				
u	∞				
v	∞				
w	∞				
x	∞				

3. Work out the time complexity of Kruskal's algorithm for Minimum Spanning Tree problem using the Union-Find data structure introduced in Week 10 lecture. Stick to the linked-list based union-find data structure when doing this analysis.

***4.** The unit progression in your course map can be depicted as a Directed Acyclic Graph (DAG), where each directed edge gives "prerequisite of" relationship. Looking into your course map, draw a DAG involving your core units and appreciate the unit relationships.

-=END=-