Student Information

Name: _____ Student ID: _____
Due Date: 29 Nov 23:59.
Submit answers on eDimension in pdf format. Submission without student information will **NOT** be marked! Any questions regarding the homework can be directed to the TA through email (contact information on eDimension).

Exercise 1

Given the following set of adjacency list:

$$adj(A) = [B, F]$$
$$adj(B) = [C, F]$$
$$adj(C) = [A, D]$$
$$adj(D) = []$$
$$adj(E) = [A]$$
$$adj(F) = [E]$$

1. Construct its corresponding **directed** graph.
   Solution:

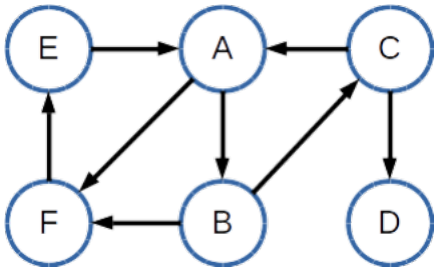


Figure 1: Solution to 1.1

2. Based on the constructed graph, write the order that the nodes are visited **for each type of graph search** when starting at node **B**. Assume that neighbours of the same node are visited in alphabetical order.:

   a. Breath-first Search (BFS)

b. Depth-first Search (DFS)

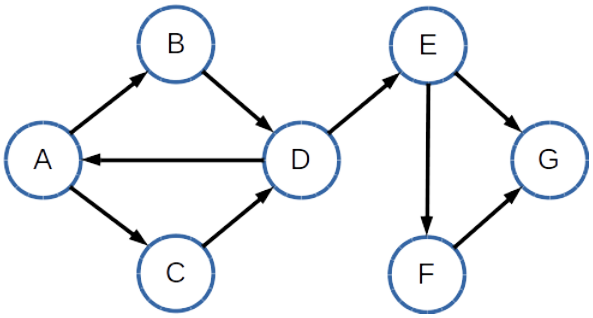
Solution:

a. B, C, F, A, D, E

b. B, C, A, F, E, D

3. For the DFS, perform a topological sort and write the order of nodes in **decreasing** order of finish times, starting from node **B**. Assume that neighbours of the same node are visited in alphabetical order.
Solution: B, C, D, A, F, E

4. Perform DFS on the following graph, starting at node **D**.
Label every edge of the graph with 'T' if it is a tree edge, 'B' if it is a back edge, 'F' if it is a forward edge, 'C' if it is a cross edge. Assume that neighbours of the same node are visited in alphabetical order. *Please refer to CLRS pg 609 for the definitions of the edges, under the "Classification of edges" heading if unsure.*



Solution:

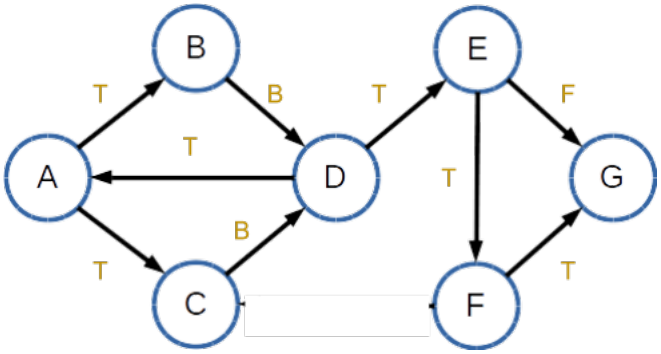
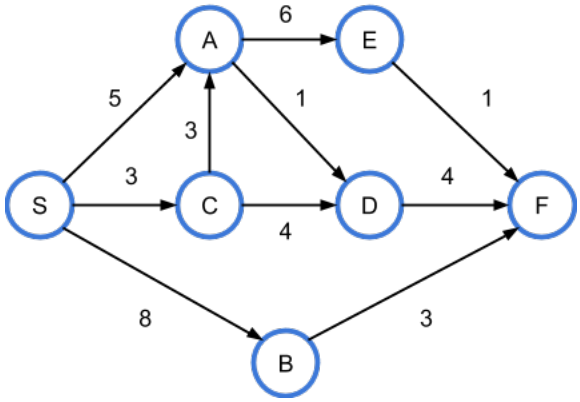


Figure 2: Solution to 1.4

Exercise 2

1. Run Dijkstra's algorithm on the graph below, starting from vertex S. What is the order in which vertices get removed from the priority queue? What is the resulting shortest-path tree? *Follow alphabetical order if 2 nodes are of the same distance.*



Solution: Order of visit of vertices: S, C, A, D, B, F, E

Resultant shortest path tree is as such:

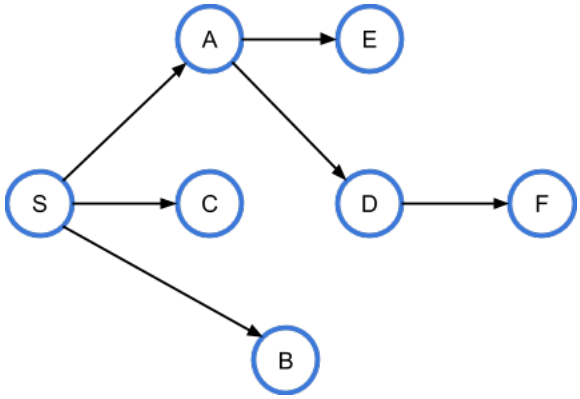
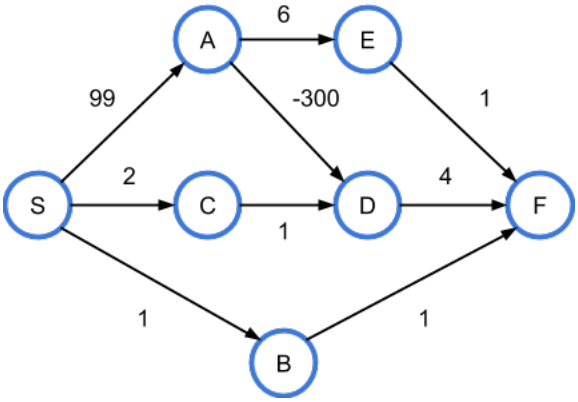


Figure 3: Solution to 2.1

2. Now run Dijkstra's algorithm on graph below, starting from vertex S. What is the order in which vertices get removed from the priority queue? If unable to run Dijkstra's algorithm, please explain why not.

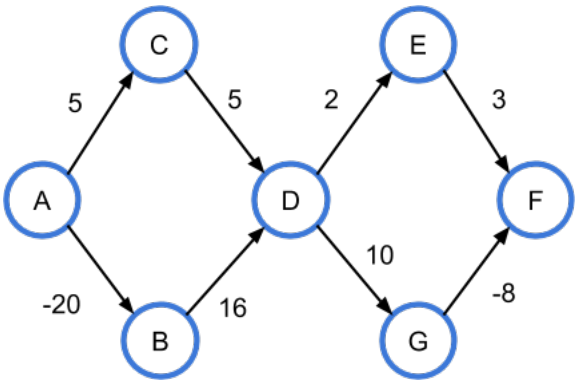
Solution: It is not possible to use Dijkstra's algorithm to obtain the shortest path in the given graph because there exists a negative weight. Upon multiple relaxation steps, vertex D will have a distance value of -201 which clearly shows the shortest path will involve vertex D. However, the result of Dijkstra's will return the shortest path as S$\rightarrow$ B$\rightarrow$F instead.



Exercise 3

1. Under what condition does Bellman-Ford algorithm fail to produce the correct shortest path?
   Solution: The algorithm fails to produce the correct shortest path when there exist a negative-weight cycle.

2. Use the Bellman-Ford algorithm on the graph below. Assuming that the starting node is A, show the iteration table and the resulting shortest path tree to **all the other nodes**. Please also provide the order of relaxations of the edges. .



Solution: Please note that other answers are acceptable as well, as long as it makes sense.

Order of relaxations (that I used): A-C, C-D, A-B, B-D, D-E, E-F, D-G, G-F

Shortest path to B: A $\rightarrow$ B
Shortest path to C: A $\rightarrow$ C
Shortest path to D: A $\rightarrow$ B $\rightarrow$ D
Shortest path to E: A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ E
Shortest path to F: A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ G $\rightarrow$ F
Shortest path to G: A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ G

Iterations	A	B	C	D	E	F	G
0	0	∞	∞	∞	∞	∞	∞
1	0	-20	5	~~10~~ -4	-2	~~1~~ -2	6
2	0	-20	5	-4	-2	-2	6

Table 1: Iteration table

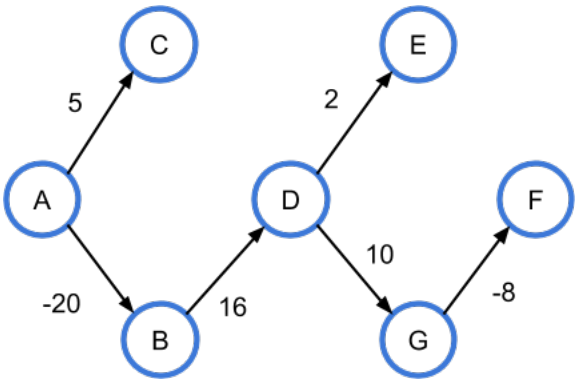


Figure 4: Solution to 3.2

Exercise 4

You would like to buy some groceries far from where you are staying and to avoid a wasted trip due to lack of money, you would want to carry as much money as you can before leaving. Since you would be travelling far, you want to limit your carrying weight to a **maximum of 7g**. Assuming that only coins of value **$1, $5 and $10** are available with their corresponding weights as **1g, 2g and 5g** respectively, you would like to maximise the amount of money you carry while not carrying above your limit. Table 2 illustrates the value of the coins and their corresponding weights.

Weight	1g	2g	5g
Value	$1	$5	$10

Table 2: Coins' weight and value

1. Please solve the problem using the **Greedy approach**. Explain your solution and comment whether it is optimal and why.
   Solution: The greedy solution involes taking the maximum valued coin possible. Using this strategy, a $5 and $10 coin will be taken. Total value will be $15 at maximum load of 7g. However, this will not be the true optimal maximum value that can be carried, since the maximum value is $16 with 3 $5 coins and 1 $1 coin at the same maximum load.

2. Please solve the problem using the **Dynamic programming approach**. Write down the DP equation. (You can answer with a table, pseudocode, text description, or any format that can

clearly present your idea.)
Solution:

Let i be the coin index, j be the weight limit
DP[i,j] be the maximum value that can be obtained by considering coin indices from 1 to i with the weight limit at j.

DP equation: $DP[i,j] = \max\{DP[i,j-w_i] + v_i, DP[i-1,j]\}$, where w_i is the weight of the coin at index i and v_i be the value of the i^{th} coin.
Base case: $DP[i,0] = 0$; $DP[0,j] = 0$

Note that at each index, one can decide whether to take it or not and if decided not to take it, can decrement the index by 1.

		Weight limit							
		0	1	2	3	4	5	6	7
Coin Index	0	0	0	0	0	0	0	0	0
	1	0	1	2	3	4	5	6	7
	2	0	1	5	6	10	11	15	16
	3	0	1	5	6	10	11	15	16

Therefore, max value is \$16.