DSC 40B - Midterm 02 Review

Problem 1.

The goal of contact tracing is to determine how the spread of a virus occurs. Which type of graph would be best for modelling the spread of a virus?

- ☐ Directed graph
- ☐ Undirected graph

Solution: Directed graph

Problem 2.

A directed graph has 7 nodes. What is the maximum number of edges it can have?

Solution: 49

Problem 3.

An undirected graph has 12 nodes. What is the maximum number of connected components it can have?

Solution: 12

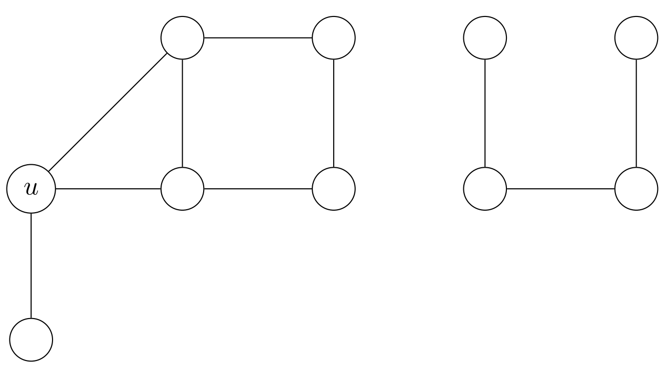
Problem 4.

A directed graph has 5 nodes. What is the largest degree that a node in the graph can possibly have?

Solution: 10

Problem 5.

How many nodes are reachable from node u in the graph?



Solution: 6

Problem 6.

Both BFS and DFS can be used to count the number of connected components in an undirected graph.

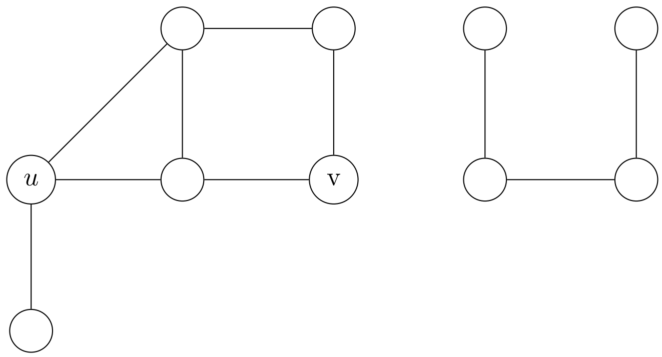
☐ True

☐ False

Solution: True

Problem 7.

How many paths are there from node u to node v in the graph below?



☐ Infinitely many

☐ 4

☐ 3

☐ 5

Solution: Infinitely many

Problem 8.

In an unweighted graph, there is at most one shortest path between any pair of given nodes.

☐ True

☐ False

Solution: False

Problem 9.

An undirected graph has 5 nodes. What is the smallest number of connected components it can have?

Solution: 1

Problem 10.

In a full BFS of a graph G=(V, E), the number of times that something is popped form the queue is 2V if the graph is undirected and V if the graph is directed.

☐ True

☐ False

Solution: False

Problem 11.

In BFS it is possible for the queue to simultaneously contain a node whose distance from the source is 3 and node whose distance from the source is 5.

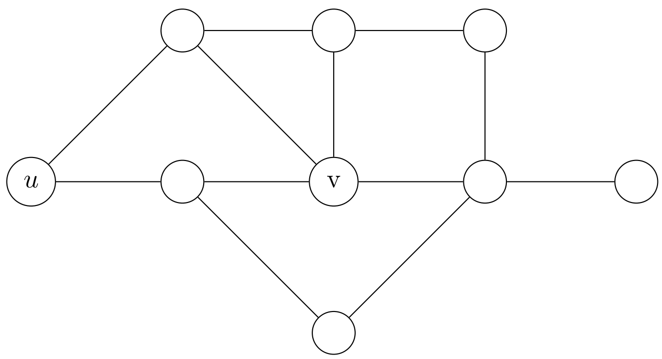
☐ True

☐ False

Solution: False

Problem 12.

Suppose a BFS is run on the graph below with u as the source.



Of course, u is the first node to be popped of the queue. Suppose that node v is the kth node popped from the queue.

a) What is the smallest that k can possibly be?

Solution: 4

b) What is the largest that k can possibly be?

Solution: 5

Problem 13.

Consider the modified mergesort given below:

```python
def bfs(graph, source, status=None):
    if status is None:
        status = {node: 'undiscovered' for node in graph.nodes}

    status[source] = 'pending'
    pending = deque([source])

    # while there are still pending nodes
    while pending:
        u = pending.popleft()
        for v in graph.neighbors(u):
```

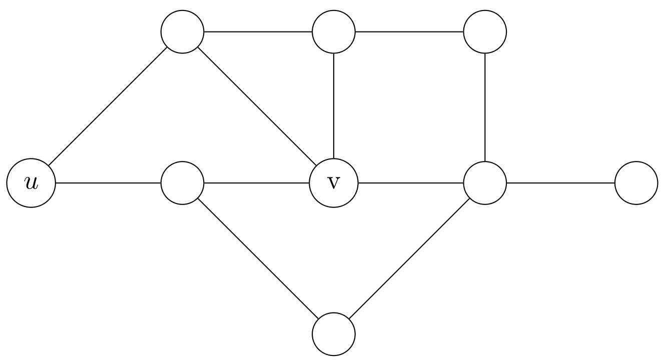
```
            # explore edge (u,v)
            if status[v] == 'undiscovered':
                print ("Hey")
                status[v] = 'pending'
                # append to right
                pending.append(v)
        status[u] = 'visited'
```

Suppose this code is run on a connected undirected graph with 12 nodes. Exactly how many times will 'Hey' be printed?

Solution: 11

Problem 14.

Suppose a DFS is run on the graph below with u as the source.



Node u will be the first node marked pending. Suppose that node v is the kth node marked pending.

a) What is the smallest that k can possibly be?

Solution: 3

b) What is the largest that k can possibly be?

Solution: 9

Problem 15.

If DFS is called on a complete graph, the time complexity is $\theta(V^2)$

☐ True

☐ False

Solution: True

Problem 16.

What is the result of updating the edge (u,v) when the est[u], est[v] and weight(u,v) are given as follows?

a) est[u] = 7, est[v] = 11, weight(u,v) = 3

Figure 1: Bellman Ford update subroutine

```
def update(u, v, weights, est, predecessor):
    if est[v] > est[u] + weights(u,v):
       est[v]=est[u]+weights(u,v)
       predecessor[v]=u
       return True
    else:
       return False
```

Solution: est[v] is updated to 10

b) est[u] = 15, est[v] = 12, weight(u,v) = -3

Solution: est[v] is not updated

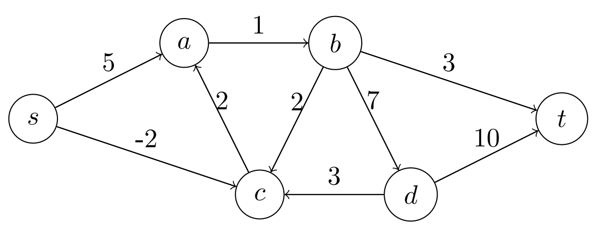
c) est[u] = 12, est[v] = 14, weight(u,v) = 3

Solution: est[v] is not updated

Problem 17.

Run Bellman-Ford on the following graph using node s as the source. Below each node u, write the shortest path length from s to u. Mark the predecessor of u by highlighting it or making a bold arrow.

```
def bellman_ford(graph, weights, source):
    est={node:float('inf') for node in graph.nodes}
    est[source]=0
    predecessor={node: None for node in graph.nodes}
    for i in range(len(graph.nodes)-1):
        for(u, v) in graph.edges:
            update(u, v, weights, est, predecessor)
    return est, predecessor
```



Solution:
predecessor : {'s':None, 'a':'c', 'b':'a', 'c':'s','d':'b', 't':'b'}
est : {'s':0, 'a':0, 'b':1, 'c':-2, 'd':8', 't':4}

Problem 18.

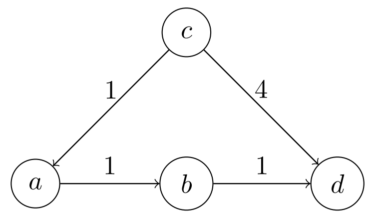
State TRUE or FALSE for the following statements:

a) If (s, v_1, v_2, v_3, v_4) is a shortest path from s to v_4 in a weighted graph, then (s, v_1, v_2, v_3) is a shortest path from s to v_3

Solution: True. Assume for the sake of contradiction that there is a path P from s to v_3 whose weight is lesser than (s, v_1, v_2, v_3). Then we can find a path from s to v_4 by combining P with the edge (v_3, v_4) whose weight is lesser than (s, v_1, v_2, v_3, v_4). This contradicts the fact that (s, v_1, v_2, v_3, v_4) is a shortest path from s to v_4.

b) Let P be a shortest path from some vertex s to some other vertex t in a directed graph. If the weight of each edge in the graph is increased by one, P will still be a shortest path from s to t.

Solution: False.



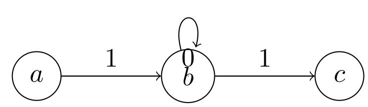
Consider the graph given above. The shortest path from c to d is (c,a,b,d) which is of weight 3. However, if the weight of each edge is increased by 1, the shortest path from c to d would be (c,d) of weight 5.

c) Suppose the update function is modified such the est[v] is updated when est[v] $\geq$ est[u] + weight(u,v) instead of strictly greater than. The est values of all nodes at the end of the algorithm would still give the shortest distance from the source.

Solution: True.

d) Suppose the update function is modified such the est[v] is updated when est[v] $\geq$ est[u] + weight(u,v) instead of strictly greater than. We can still find the shortest path from the source to any node using the predecessors using the new algorithm.

Solution: False.



Consider the graph given above. Let a be the source in the execution of Bellman Ford algorithm. Updating the edge will result in est[b] = 1 and predecessor[b] = a. However, updating the edge (b,b) will result in predecessor[b] = b according to the new update algorithm. Therefore, it would not be possible to find the shortest path from a to b using the predecessors.