DSC 40B - Quiz 06

Name: | Student ID:

About the quizzes:

- Quizzes in DSC 40B are *optional* and graded pass/fail.
- A score of 70% or higher earns a "pass" and 1.5 credits toward your final grade.
- If you don't pass, no credits are earned, but it doesn't hurt your grade.
- You have 30 minutes to complete the quiz.
- Unfortunately, we can't answer clarifying questions during the quiz. If you think a question has a bug or is unclear, please let us know in a private post on Campuswire after the quiz, and we'll take it into account when grading.

Problem 1.

An **undirected** graph has 7 nodes. What is the greatest number of edges it can have?

◯ 7 ◯ 21 ◯ 42 ◯ 49

Problem 2.

Let $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, a), (a, c), (d, b), (c, a)\}$ be the nodes and edges of a **directed** graph. How many predecessors does node b have?

◯ 1 ◯ 2 ◯ 3 ◯ 4

Problem 3.

Consider the **undirected** graph $G = (V, E)$ with

$$V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$E = \{(0, 4), (4, 8), (1, 5), (2, 6), (6, 1), (3, 7)\}$$

How many connected components does G have?

◯ 3 ◯ 4 ◯ 5 ◯ 6

Problem 4.

Suppose an **undirected** graph $G = (V, E)$ has two connected components, C_1 and C_2, where C_1 has 5 nodes and C_2 has 3 nodes. What is the largest that $|E|$ can possibly be?

◯ 8 ◯ 13 ◯ 16 ◯ 28

Problem 5.

Let $G = (V, E)$ be an undirected graph, and suppose that u, v, and w are three nodes in V. Suppose that u and v are in the same connected component, but u and w are not in the same connected component. True or False: it is possible that v and w are in the same connected component.

◯ True

◯ False

Problem 6.

An *isolated* node in an undirected graph is a node that has no neighbors.

a) Suppose a graph $G = (V, E)$ is stored as an **adjacency matrix**. What is the best possible worst-case time complexity of an efficient algorithm that determines whether a **given node** v is isolated?

◯ $\Theta(1)$ ◯ $\Theta(V)$ ◯ $\Theta(V + E)$ ◯ $\Theta(V^2)$

b) Now suppose G is stored as an **adjacency list**. What is the best possible worst-case time complexity of an efficient algorithm that determines whether a **given node** v is isolated?

◯ $\Theta(1)$ ◯ $\Theta(V)$ ◯ $\Theta(V + E)$ ◯ $\Theta(V^2)$

Problem 7.

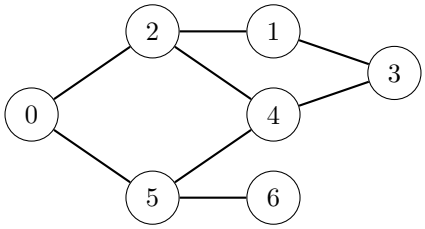
Suppose a **connected**, **undirected** graph $G = (V, E)$ is stored as an adjacency list in the variable `adj`. What is the time complexity of the code below?

```
total = 0
seen = {u: False for u in range(len(adj))}
for u in range(len(adj)):
    for v in adj[u]:
        total += 1
        if not seen[v]:
            seen[v] = True
            for w in range(len(adj)):
                total += 1
    print(u, total)
```

◯ $\Theta(V)$ ◯ $\Theta(V + E)$ ◯ $\Theta(V^2)$ ◯ $\Theta(V \cdot E)$

Problem 8.

Suppose a breadth-first search is run on the undirected graph below, starting at node 0, and using the convention that a node's neighbors are produced in ascending order by label. In which order are the nodes visited?



◯ 0, 1, 2, 3, 4, 5, 6 ◯ 0, 2, 1, 3, 4, 5, 6 ◯ 0, 2, 5, 1, 4, 6, 3 ◯ 0, 5, 2, 6, 4, 1, 3

Problem 9.

The adjacency matrix of a **directed** graph with nodes $0, 1, \ldots, 5$ is shown below. As in lecture, the entry in row i and column j is 1 if $(i, j) \in E$ and 0 otherwise.

	0	1	2	3	4	5
0	0	**1**	0	0	0	0
1	0	0	**1**	0	0	0
2	**1**	0	0	**1**	0	0
3	0	**1**	0	0	0	0
4	0	0	0	**1**	0	**1**
5	**1**	0	0	0	0	0

True or False: there is a path from node 0 to node 5.

◯ True

◯ False

Problem 10.

The code below is the full BFS from lecture with a new `print` statement, marked with a comment.

```
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 pending:
        u = pending.popleft()
        for v in graph.neighbors(u):
            if status[v] == 'undiscovered':
                status[v] = 'pending'
                pending.append(v)
        status[u] = 'visited'

def full_bfs(graph):
    status = {node: 'undiscovered' for node in graph.nodes}
    for node in graph.nodes:
        if status[node] == 'undiscovered':
            print("bar") # <--- this is new
            bfs(graph, node, status)
```

Suppose `full_bfs` is run on an **undirected** graph and prints `"bar"` 4 times. Which of the following must be true about the graph?

◯ It has exactly 4 connected components.

◯ It has exactly 5 connected components.

◯ It has exactly 4 nodes.

◯ It has exactly 4 edges.