## Q1
0 Points

Welcome to DSC 40B's Midterm 02. The exam is open book. You may use your notes, the class materials, a Python interpreter, Google, etc. as references.

By marking the below, you agree to the exam's honor code. You agree to behave honestly and fairly during the exam. In particular, you won't discuss the exam with others while taking it.

◉ I agree.

### Instructions

- In every question below, if it is not mentioned whether a graph is weighted or unweighted, you should assume that it is *un*weighted.
- We will not be able to answer clarifying questions during the exam, but if you think a problem is unclear make a note and let us know with a private post on Ed after the exam.

## Q2
1 Point

Suppose a hash table with 1000 bins stores 2000 numbers. Collisions are resolved with chaining. True or False: it is possible for one of the hash table's bins to contain zero elements.

◉ True

○ False

## Q3
1 Point

Suppose a hash table is implemented so that it has ten bins, and that this number of bins is not increased when new elements are inserted; that is, the hash table is not allowed to "grow". Let $n$ be the number of elements stored in the hash table.

What is the expected time complexity of querying an element in this hash table, as a function of $n$? You may assume that collisions are resolved with chaining, that the bins are linked lists, and that the hash function is "good" in that it appears to uniformly distribution elements among bins.

- ○ $\Theta(1)$
- ○ $\Theta(\log n)$
- ◉ $\Theta(n)$
- ○ $\Theta(n \log n)$
- ○ $\Theta(n^2)$
- ○ $\Theta(n^3)$

## Q4

1 Point

What is the expected time complexity of the code below, assuming that `numbers` is a Python `set`? Recall that Python's `set`s are implemented as hash tables.

```
def foo(numbers):
    """Assume `numbers` is a Python `set` containing n numbers."""
    count = 0
    for x in numbers:
        if -x in numbers:
            count += 1

    return count / len(numbers)
```

- ○ $\Theta(1)$
- ○ $\Theta(\log n)$
- ◉ $\Theta(n)$
- ○ $\Theta(n \log n)$
- ○ $\Theta(n^2)$
- ○ $\Theta(n^3)$

## Q5

1 Point

An undirected graph $G = (V, E)$ has 7 edges. What is the *smallest* number of nodes that it can have?

=5+-0

## Q6

1 Point

Let $G = (V, E)$ be an undirected graph. Suppose $u$ and $v$ are two nodes in $G$ which are in the same connected component. True or False: the edge $(u, v)$ must be in $G$.

- ○ True
- ◉ False

## Q7

1 Point

Let $G = (V, E)$ be an undirected graph with $V = \{a, b, c, d, e, f, g, h\}$ and $E = \{(a, f), (b, e), (b, c), (c, e), (e, d), (c, d)\}$.

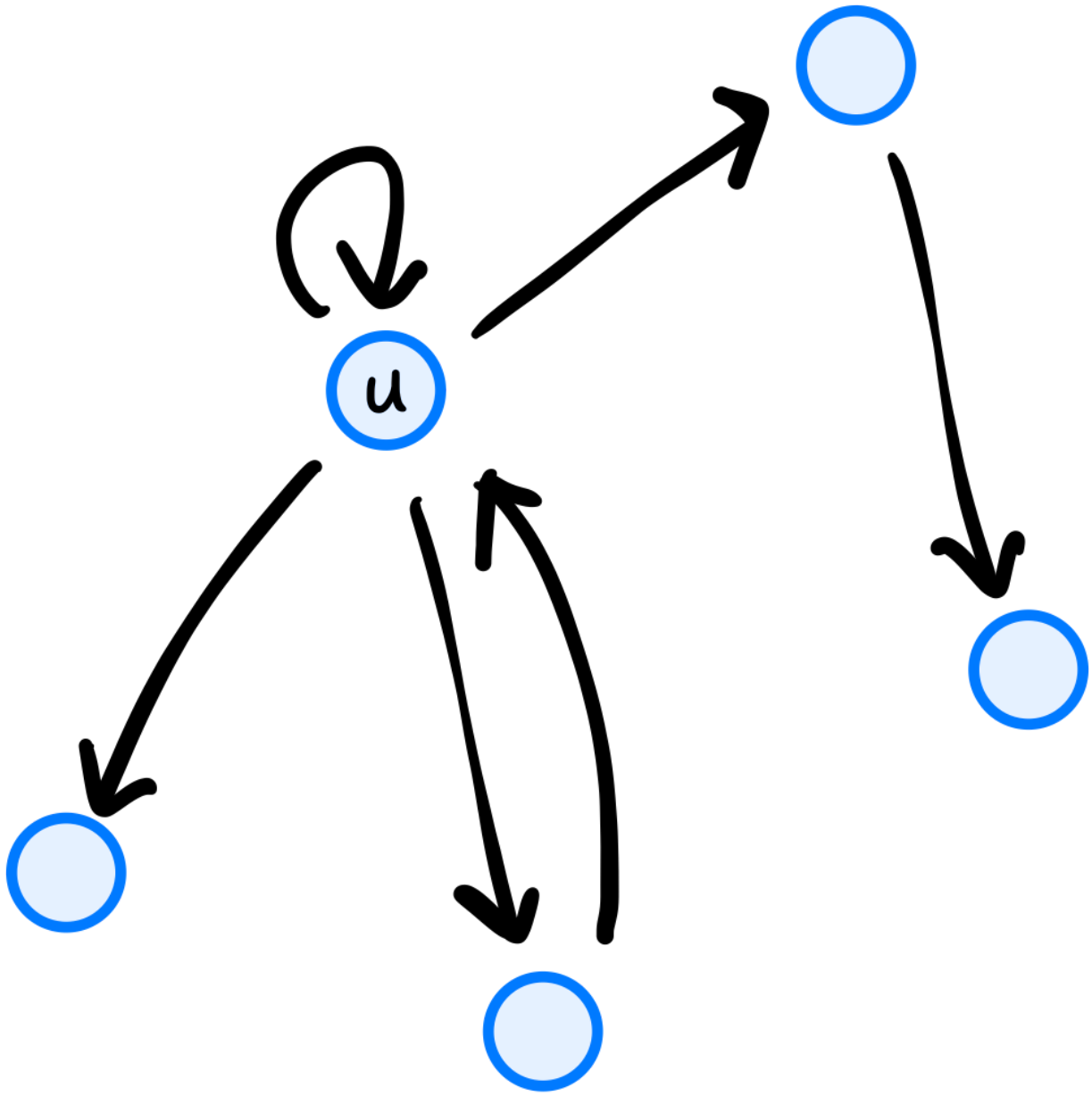
How many connected components does $G$ have?

=4+-0

## Q8

1 Point

What is the *degree* of node $u$ in the graph shown below?




=6+-0

## Q9

1 Point

Let $G = (V, E)$ be an (unweighted) directed graph, and let $s$, $a$, and $b$ be three nodes in the graph. Suppose $(s, u_1, u_2, u_3, a, u_4, u_5, b)$ is a shortest path from $s$ to $b$. True or

False: it is possible that the shortest path distance from $s$ to node $a$ is 3.

- ○ True
- ◉ False

## Q10

1 Point

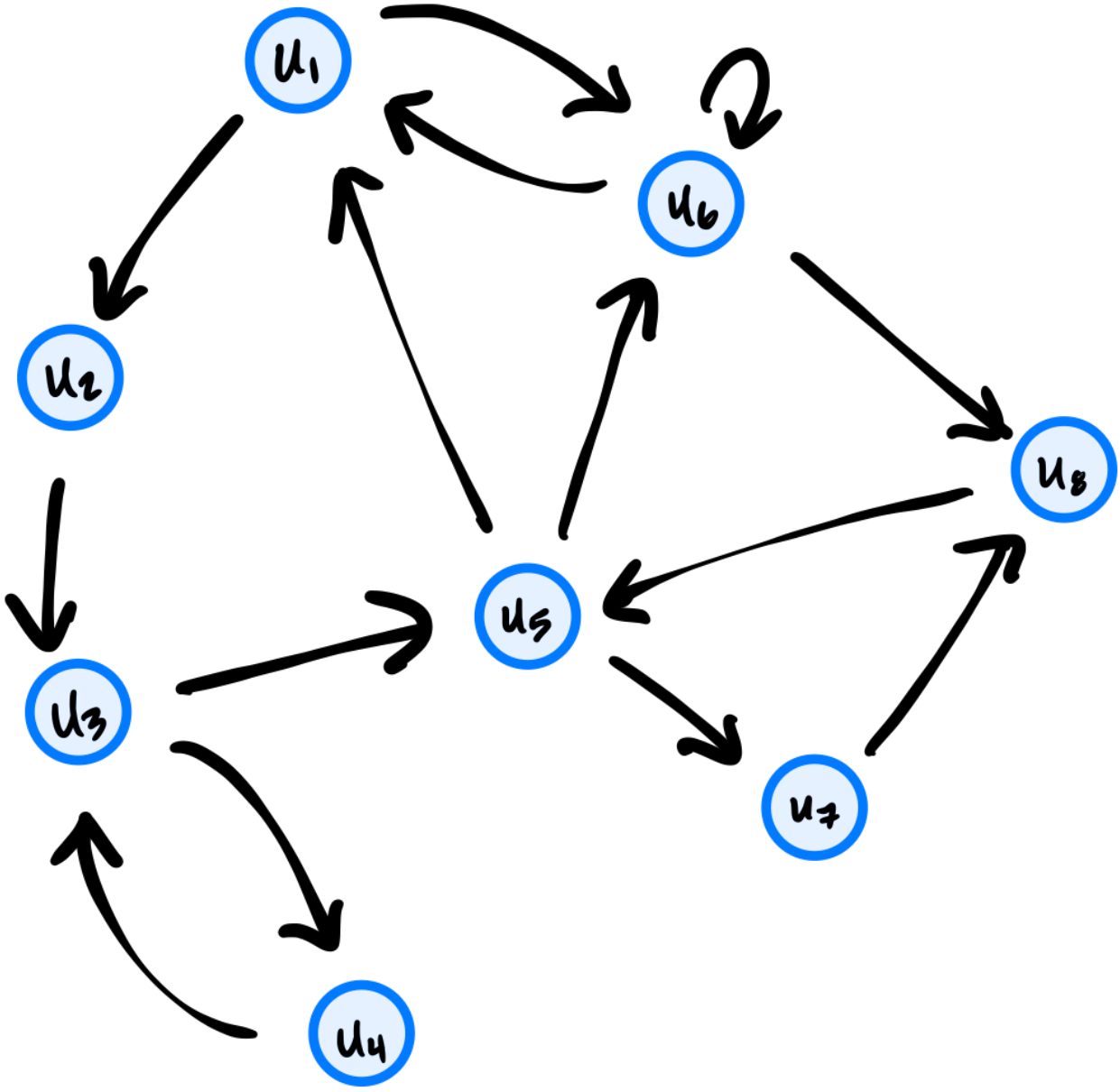
Suppose a graph $G = (V, E)$ is stored using an adjacency list representation. In the worst case, what is the time complexity of finding the maximum degree of any node in the graph? You may assume that your linked list implementation stores the size of the list and can return it in constant time.

- ◉ $\Theta(V)$
- ○ $\Theta(E)$
- ○ $\Theta(V + E)$
- ○ $\Theta(V^2)$
- ○ $\Theta(E^2)$

## Q11

1 Point

Suppose a BFS is run on the graph shown below using node $u_1$ as the source. How many nodes are popped from the queue *before* node $u_7$ is popped? (Your count should include the source node itself.)

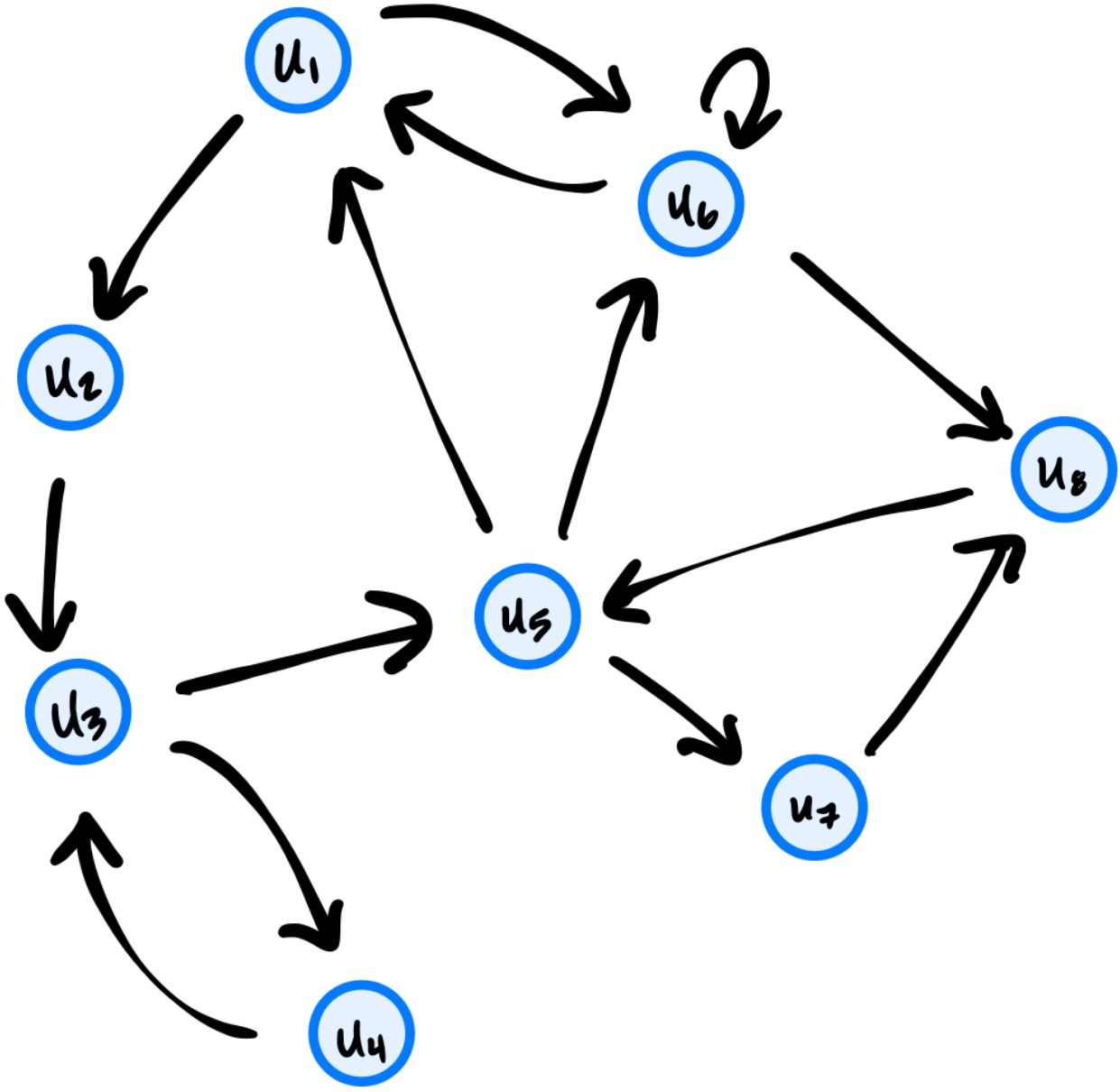



=7+-0

## Q12

1 Point

Suppose a BFS is run on the graph shown below using node $u_1$ as the source. Adopt the convention that a node's neighbors are produced in ascending order by their label. What is the BFS predecessor of node $u_8$?




- $u_1$
- $u_2$
- $u_3$
- $u_4$
- $u_5$
- $u_6$
- $u_7$
- $u_8$

## Q13

1 Point

Suppose that $s$, $u$, and $v$ are nodes in an undirected graph $G = (V, E)$. Suppose that the shortest path distance from $s$ to $v$ is 4, while the shortest path distance from $s$ to $v$ is 6.

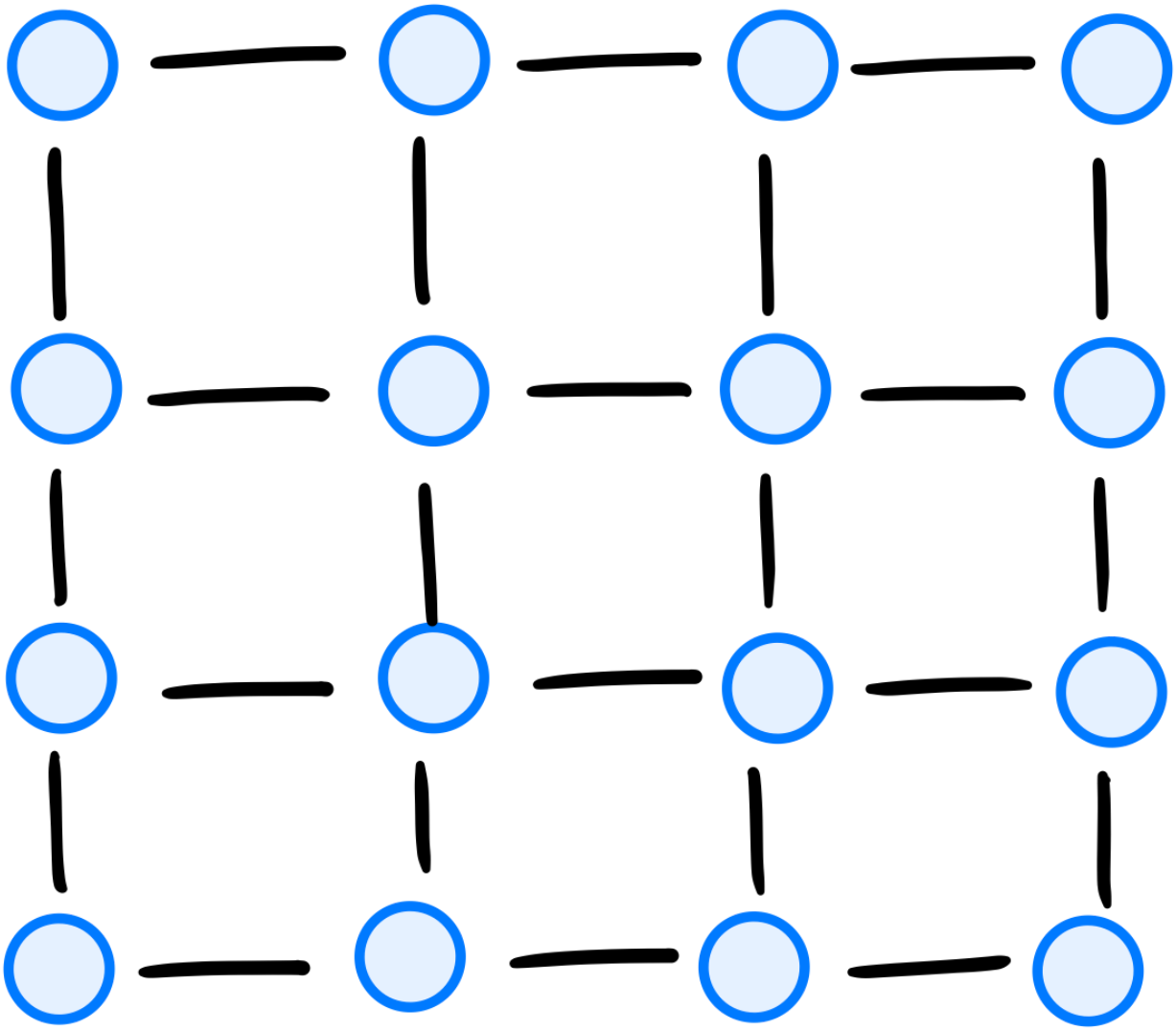
True or False: it is possible that, during a BFS using $s$ as the source, the pending queue contains $u$ and $v$ at the same time.

- ○ True
- ◉ False

## Q14

1 Point

An $n \times n$ *lattice graph* is an undirected graph with $n^2$ nodes in which each node is placed on a grid, and an edge is added between the node and the nodes to the immediate north, south, east, and west. For example, a $4 \times 4$ lattice graph looks like this:



What is the time complexity of BFS when run on an $n \times n$ lattice graph?

- $\Theta(n)$
- $\Theta(n \log n)$
- $\Theta(n^2)$
- $\Theta(n^3)$
- $\Theta(n^4)$

## Q15

1 Point

Suppose the code below is run on a graph $G = (V, E)$. What is its time complexity? You may assume that $|E| \geq 1$, but you should not assume anything else about the number of

edges. You may assume that the graph is stored using an adjacency list representation.

```
for u in graph.nodes:
    for v in graph.neighbors(u):
        print(u, v)
```

- ○ $\Theta(E)$
- ○ $\Theta(V)$
- ◉ $\Theta(V+E)$
- ○ $\Theta(V^2)$
- ○ $\Theta(VE)$

## Q16
1 Point

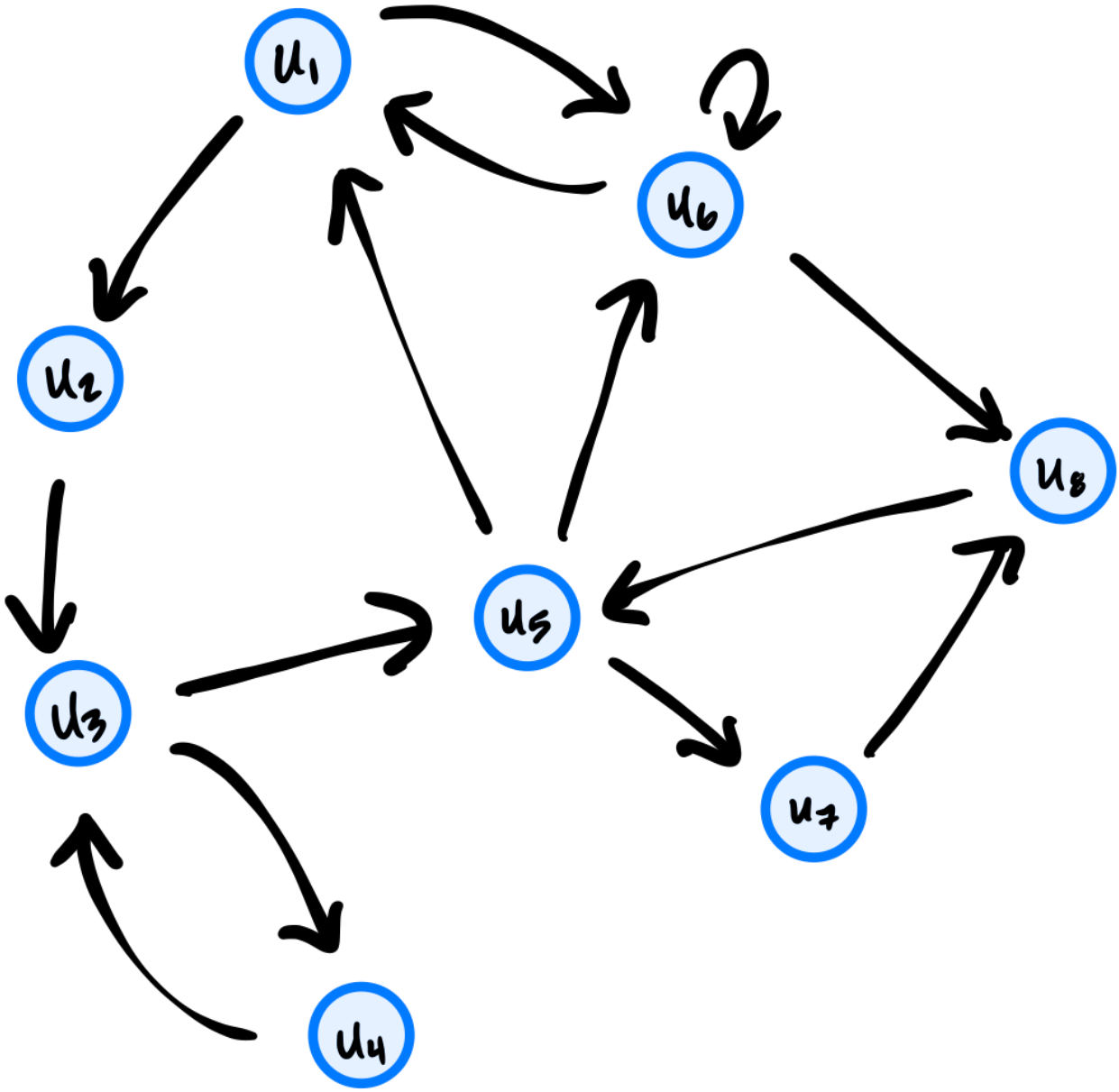
Suppose $s$ and $u$ are nodes in an unweighted, undirected graph $G = (V, E)$, and that there are several shortest paths from $s$ to $u$. Suppose two BFSs are run starting at node $s$; one of them uses the convention that neighbors are produced in *ascending* order by label, and the other uses a different convention.

Which one of the following claims is true about the results?

- ○ The shortest path distance to $u$ as well as $u$'s BFS predecessor must be the same in both BFSs.
- ○ $u$'s BFS predecessor must be the same in both BFSs, but the shortest path distances to $u$ may be different
- ◉ The shortest path distance to $u$ must be the same in both BFSs, but $u$'s BFS predecessor may be different
- ○ Both the shortest path distance to $u$ and $u$'s BFS predecessor may be different between the BFSs

## Q17
1 Point

Suppose a DFS is run on the graph shown below using node $u_1$ as the source and the convention that neighbors are produced in ascending order by label. What will be the DFS predecessor of node $u_6$?



- ○ $u_1$
- ○ $u_2$
- ○ $u_3$
- ○ $u_4$
- ◉ $u_5$
- ○ $u_6$
- ○ $u_7$
- ○ $u_8$

## Q18
1 Point

Suppose $G = (V, E)$ is an undirected graph with 10 connected components, each of which is a tree. That is, $G$ is a forest of 10 trees.

What is the time complexity of full DFS when run on $G$?

- ○ $\Theta(1)$
- ◉ $\Theta(V)$
- ○ $\Theta(V^2)$
- ○ $\Theta(E^2)$
- ○ $\Theta(VE)$

## Q19
1 Point

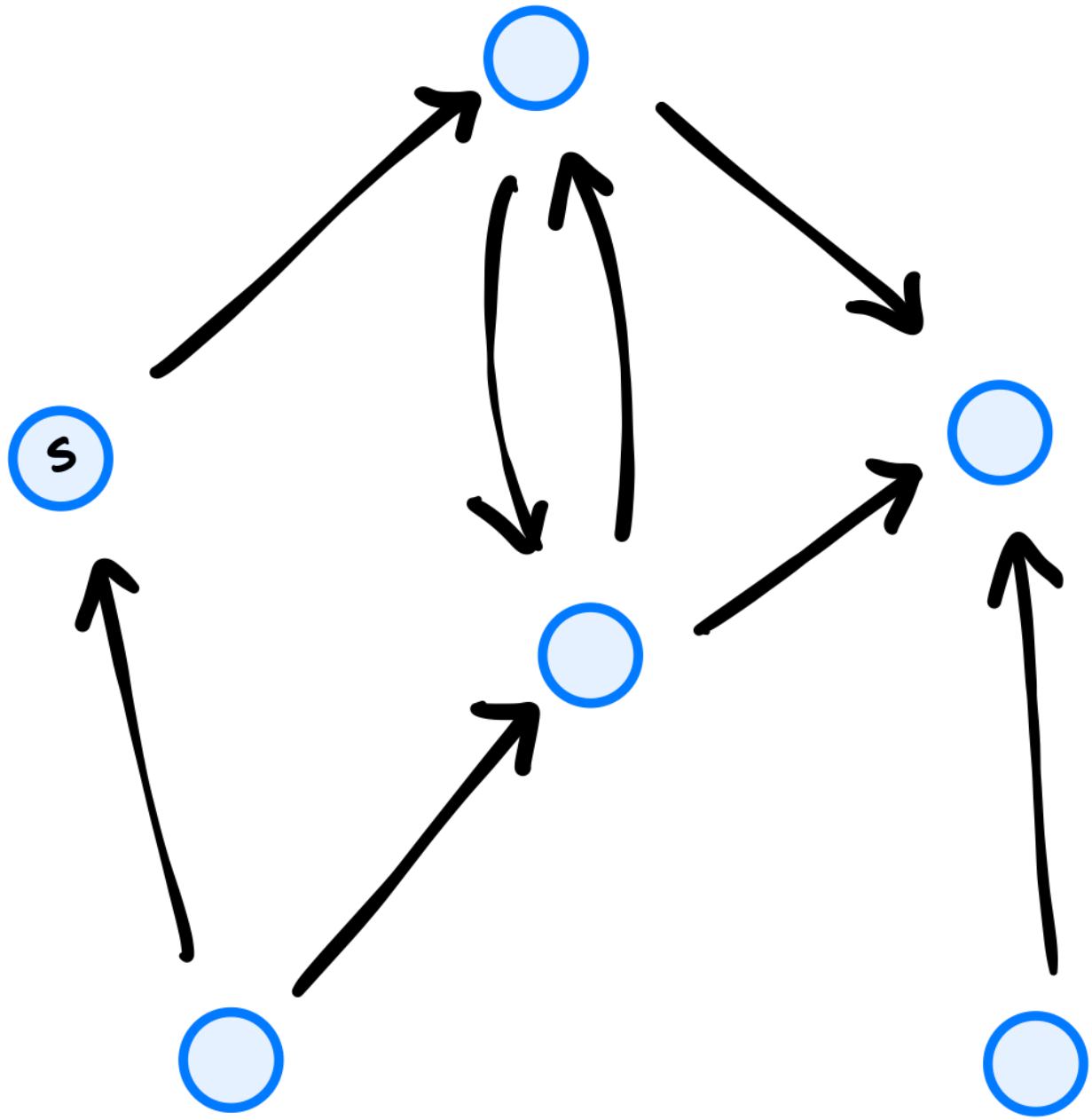
Consider the code shown below. It shows DFS as seen in lecture, with one modification: a `print` statement in the for-loop.

```
def dfs(graph, u, status=None):
    """Start a DFS at `u`."""
    # initialize status if it was not passed
    if status is None:
```

```
        status = {node: 'undiscovered' for node in graph.nodes}

    status[u] = 'pending'
    for v in graph.neighbors(u):
        print("Hello!")
        if status[v] == 'undiscovered':
            dfs(graph, v, status)
    status[u] = 'visited'
```

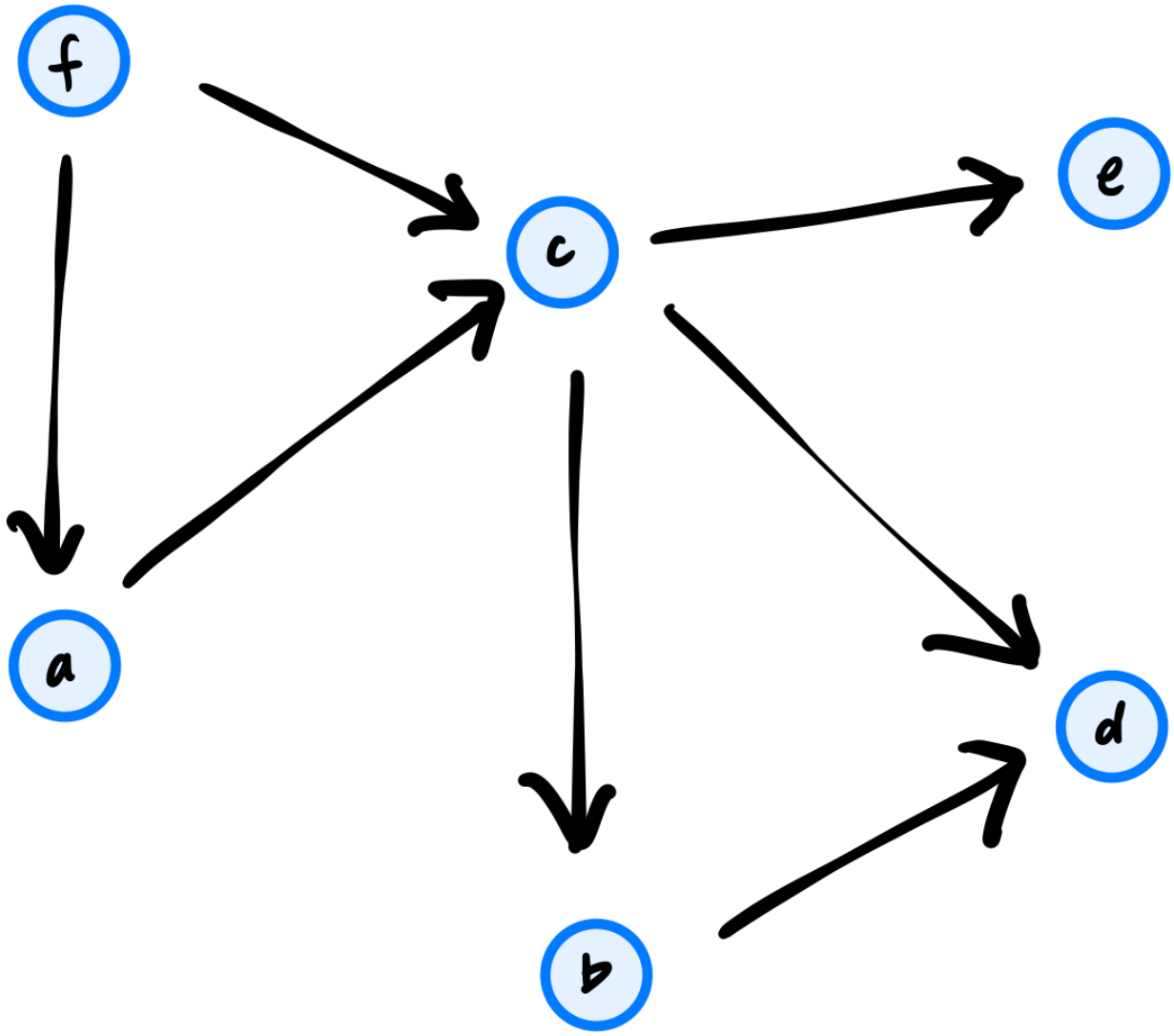
Suppose the above code is run on the graph shown below using node $s$ as the source. How many times will "Hello!" be printed?




=5+-0

## Q20
2 Points

In this three-part problem, consider the graph below:




Suppose a *full* DFS is run on this graph using the convention that neighbors are produced in ascending order by label. You should also assume that when the nodes of the graph are looped over, as in `for node in graph.nodes`, the nodes are also produced in ascending order by label.

## Q20.1
0.5 Points

What will be the start time of node $e$?

=7+-0

## Q20.2
0.5 Points

What will be the finish time of node $a$?

=10+-0

## Q20.3
1 Point

Give a topological sort of the graph's nodes.

## Q21
1 Point

During a call to `dfs` on a node $u$, 23 nodes are marked as visited, including node $u$ itself. What will be the difference between the finish time of $u$ and the start time of $u$? That is, what is `finish[u] - start[u]`?

=45+-0

## Q22
2 Points

In this two-part problem, consider the function `foo`, defined below:

```
def foo(graph, u):
    count = 1
    for v in graph.neighbors(u):
        count += foo(graph, v)
    return count
```

## Q22.1
1 Point

Suppose a directed graph $G$ has $n$ nodes and $n-1$ edges, and every node in the graph is reachable from node $u$.

What is the time complexity of `foo` when run on this graph with node $u$ as the source?

- ○ $\Theta(1)$
- ○ $\Theta(\log n)$
- ◉ $\Theta(n)$
- ○ $\Theta(n \log n)$
- ○ $\Theta(n^2)$
- ○ $\Theta(n^3)$

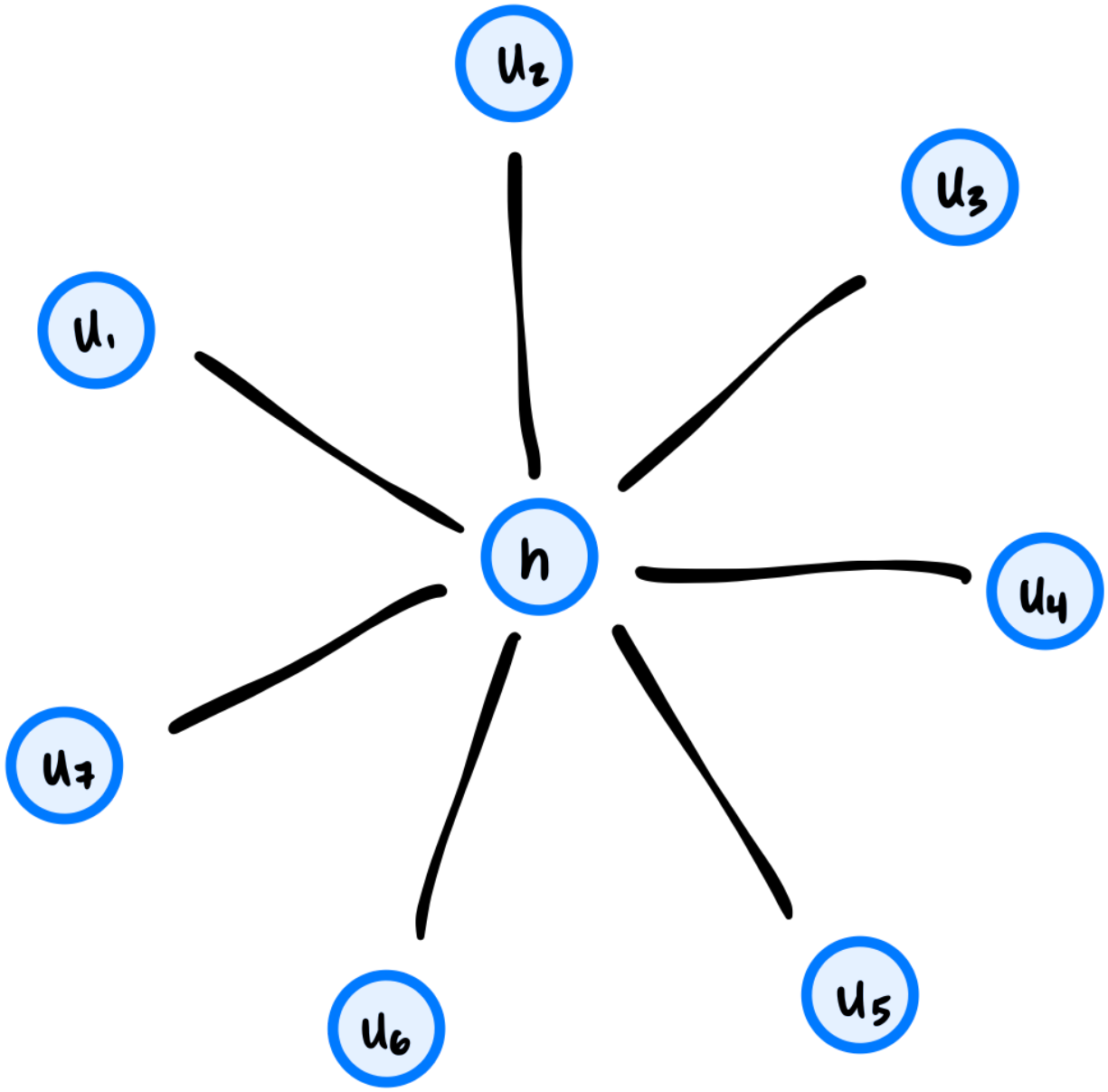
## Q22.2
1 Point

Now suppose the directed graph $G$ has 90 nodes and 89 edges, and every node is reachable from node $u$. What is returned by `foo` when it is called on node $u$?

=90+-0

## Q23
1 Point

A "hub and spoke" graph with $n$ nodes consists of a single "hub" node, $h$, along with $n-1$ "spoke" nodes, $u_1, \ldots, u_{n-1}$, and $n-1$ edges from the hub node $h$ to each of the spoke nodes. For example, a "hub and spoke" graph with 8 nodes looks like the below:

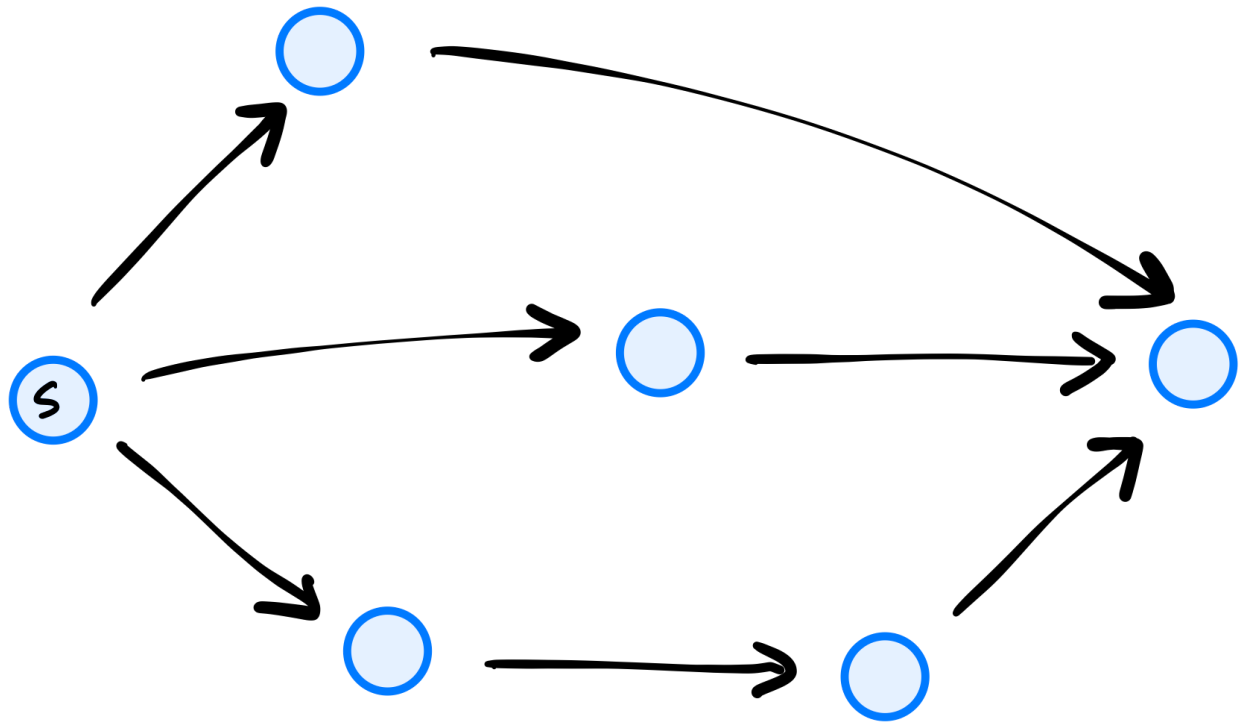



What is the worst case time complexity of an efficient algorithm which takes as input a hub and spoke graph and returns the label of the hub node? You may assume that the graph is represented as an adjacency list.

- ◉ $\Theta(1)$
- ○ $\Theta(\log n)$
- ○ $\Theta(n)$
- ○ $\Theta(n \log n)$
- ○ $\Theta(n^2)$
- ○ $\Theta(n^3)$

## Q24
1 Point

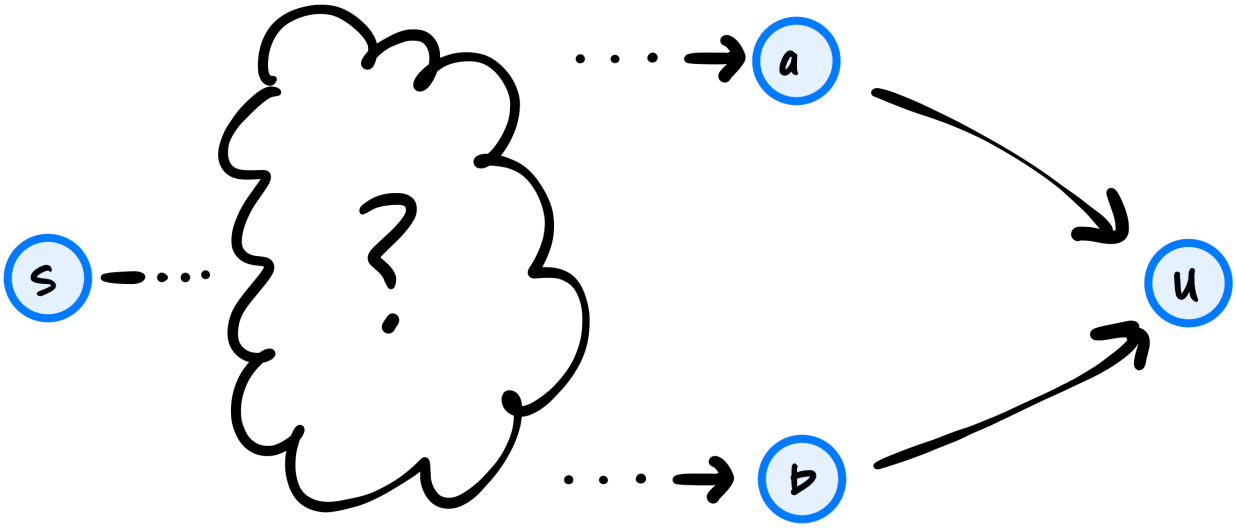
Suppose the Bellman-Ford algorithm with early stopping is run on the weighted graph shown below using node $s$ as the source (the edge weights are purposely not shown). In the worst case, how many iterations of Bellman-Ford will be performed? That is, how many iterations of the outer `for`-loop will occur?




=4+-0

## Q25
1 Point

Suppose that $s$, $a$, $b$, and $u$ are nodes in a directed, weighted graph $G = (V, E, \omega)$, and that $a$ and $b$ are the only predecessors of $u$, as shown in the picture below:




The cloud with the `?` is meant to represent the rest of the graph, whose structure is unknown.

Suppose the Bellman-Ford algorithm is run on $G$ with $s$ as the source. At some moment, $u$'s estimated predecessor is node $a$; that is, in the current shortest path from $s$ to $u$, node $a$ comes before $u$. Later, the algorithm is paused, and it is seen that $u$'s estimated predecessor is node $b$.

Suppose the algorithm is paused once again at a later time. True or False: it is possible that node $u$'s estimated predecessor is again node $a$. You may assume that the graph has no negative cycles.

- (●) True
- ( ) False

## Q26
1 Point

Consider the situation described in the previous problem, except now we are running Dijkstra's algorithm. That is, suppose Dijkstra's algorithm is run on $G$ with $s$ as the source. At some moment, $u$'s estimated predecessor is node $a$; that is, in the current shortest path from $s$ to $u$, node $a$ comes before $u$. Later, the algorithm is paused, and it is seen that $u$'s estimated predecessor is node $b$.

Suppose the algorithm is paused once again at a later time. True or False: it is possible that node $u$'s estimated predecessor is again node $a$. You may assume that the graph has

no negative edge weights.

- ○ True
- ◉ False

## Q27

1 Point

Suppose a weighted graph $G$ contains negative edge weights, but no negative cycles. True or False: Dijkstra's algorithm is guaranteed to find the correct weighted shortest path from the source to each node in $G$.

- ○ True
- ◉ False