組別: 姓名: 學號: 分數: 批改者:

一人寫一份，可討論。 上台講解且正確者，全組自行加一分。

Algorithms

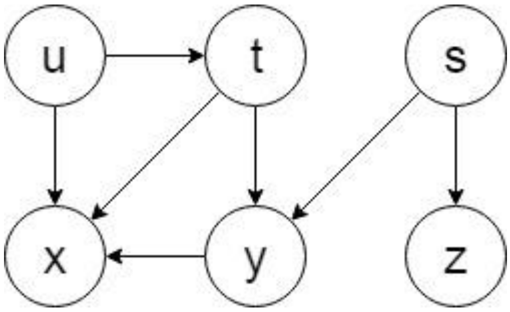
Classwork #6 Video 5.1-5.4

Each problem is 3 points.

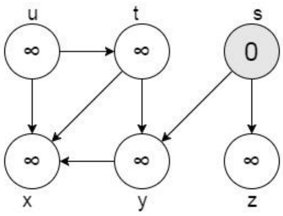
3 = correct. 2 = one mistake. 1 = many mistakes. 0 = no answer.

1. Breadth First Search [5.2]

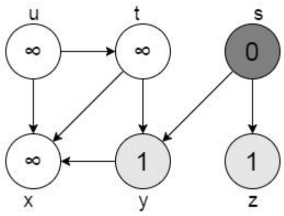
Show the *d* value and queue *Q* that result from running breadth-first search on the directed graph of figure below, using vertex *s* as the source. Please show your BFS procedure step-by-step in the same form on lecture slide. Please break tie in alphabetical order (choose *a* if both *a*, *b* are candidates).



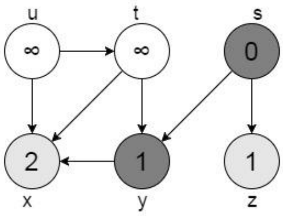
Sol:



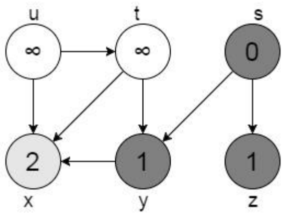
Q:s



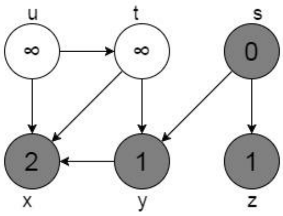
Q:y,z



Q:z,x



Q:x



Q:Ø

2. Graph in Adjacency List [5.1]

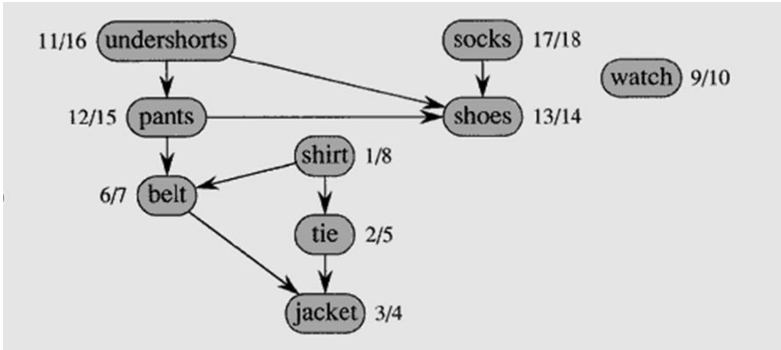
Given an adjacency-list representation of a directed graph, what is the time complexity to compute the in-degree of every vertex? Please briefly explain your algorithm in sentence.

Sol:

To compute in-degree for each vertex requires checks through all the edge, thus at least $\Theta(|E|)$.

3. Topological Sort [5.4]

Another way to perform topological sorting on a directed acyclic graph $G = (V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. For example, we can take vertices in order: watch, socks, undershorts, pants, shoes, shirt, belt, tie, jacket.



Please fill the blank in the pseudo-code to finish the algorithm. Note that a cycle detection scheme is also employed in this algorithm (line 24). Please also finish the *if* statement to detect it if there's cycle within G.

```
TOPOLOGICAL-SORT(G)
//Initialize in-degree, Θ(V) time
for each vertex u ∈ G.V
        u.in-degree = 0
//Compute in-degree, Θ(V + E) time
for each vertex u ∈ G.V
        for each v ∈ G.Adj[u]
                v.in-degree = v.in-degree + 1
//Initialize Queue, Θ(V) time
Q = ∅
for each vertex u ∈ G.V
        if ________________________
                ENQUEUE(Q, u)
//while loop takes O(V + E) time
while Q ≠ ∅
        u = DEQUEUE(Q)
        output u
        // for loop executes O(E) times total
        for each v ∈ G.Adj[u]
                v.in-degree = ________________________
                if ________________________
                        ENQUEUE(Q, v)
//Check for cycles, O(V) time
for each vertex u ∈ G.V
        if ________________________
                report that there is a cycle
```

Sol:

```
TOPOLOGICAL-SORT(G)
//Initialize in-degree, Θ(V) time
for each vertex u ∈ G.V
      u.in-degree = 0
//Compute in-degree, Θ(V + E) time
for each vertex u ∈ G.V
      for each v ∈ G.Adj[u]
            v.in-degree = v.in-degree + 1
//Initialize Queue, Θ(V) time
Q = ∅
for each vertex u ∈ G.V
      if u.in-degree == 0
            ENQUEUE(Q, u)
//while loop takes O(V + E) time
while Q ≠ ∅
      u = DEQUEUE(Q)
      output u
      // for loop executes O(E) times total
      for each v ∈ G.Adj[u]
            v.in-degree = v.in-degree - 1
            if v.in-degree == 0
                  ENQUEUE(Q, v)
//Check for cycles, O(V) time
for each vertex u ∈ G.V
      if u.in-degree ≠ 0
            report that there is a cycle
```

4. Depth-first Search [5.3]

Prove that in DFS of undirected graph, every edge is either a tree edge or a back edge (no cross edge or forward edge exist).

Sol:

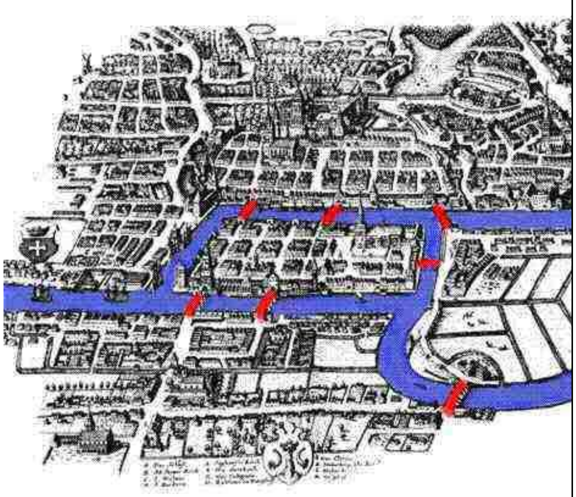
Let (u,v) be an edge. Suppose u is discovered first. Then, v will become u's descendent so that $v.f < u.f$.

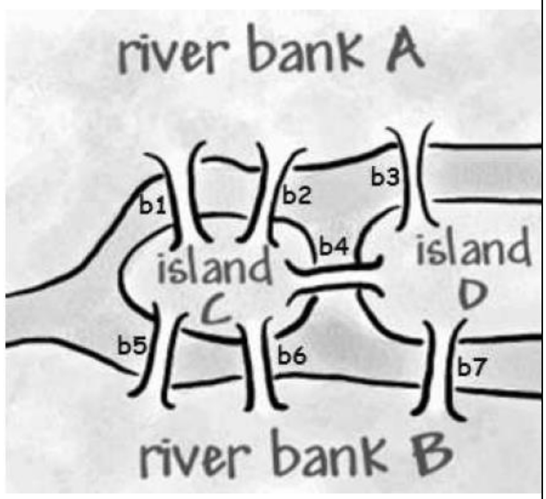
If u discovers v, than (u,v) is tree edge.

Else, (u,v) is explored after v discovered. Then, (u,v) must be explored from v and because $v.f < u.f$, so (u,v) is back edge.

5. **Euler tour [5.1]**

Leonhard Euler in 1736 first considered the following mathematical puzzle: The city of Konigsberg has seven bridges b_i, $i = 1, \ldots, 7$, with two river banks A and B and two islands C and D, as shown in the figures below. Euler wondered if it is possible to start at some place in the city, cross every bridge exactly once, and return to the starting place. We construct an undirected graph $G = (V, E)$ to solve this problem. $V = \{A, B, C, D\}$ and every edge $e = (v_i, v_j) \in E$ if there is a bridge connecting v_i and v_j. Does such a tour exist? If it exists, please mark the path. If it does not exist, please explain the reason.





Sol:

No.

We can observe that a Euler tour will go through all edges in G without repetition with the same source and target. It is clear that, for each vertex v' in the cycle, the numbers of edges that go in and go out v' must be equal. However, the degree of every vertex in G is odd. So the Euler tour does not exist.