1. By definition of the incidence matrix B of the directed graph G, the (i,j) element of BB^T is

$$(BB^T)_{ij} = \sum_{e \in E} b_{ie} b_{ej}^T = \sum_{e \in E} b_{ie} b_{je}.$$

   Furthermore,

   - if $i = j$, then $b_{ie} b_{je} = 1$ (It is $1 \cdot 1$ or $(-1) \cdot (-1)$ whenever e enters or leaves vertex i, and 0 otherwise.)
   - if $i \neq j$, then $b_{ie} b_{je} = -1$ when $e = (i,j)$ or $e = (j,i)$, and 0 otherwise.

   Therefore,

$$(BB^T)_{(i,j)} = \begin{cases} \text{degree of } i = \text{in-degree} + \text{out-degree} & \text{if } i = j \\ -(\#\text{ edges connecting } i \text{ and } j) & \text{if } i \neq j \end{cases}$$

   The count for the number of edges between the two vertices includes in either direction, so it can be 0, 1 or 2.

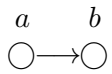
2. (a) Note that each edge (u,v) contributes 1 to $d(u)$ and 1 to $d(v)$. Hence each edge contributes exactly 2 to the sum $\sum_{v \in V} d(v)$, which gives $\sum_{v \in V} d(v) = 2|E|$.

   (b) Let V_o be the set of vertices with odd degree and V_e be the set of vertices with even degree. Then

$$\sum_{v \in V_o} d(v) + \sum_{v \in V_e} d(v) = 2|E| \quad \Longrightarrow \quad \sum_{v \in V_o} d(v) = 2|E| - \sum_{v \in V_e} d(v).$$

   The right-hand-side of this equation is even, and the left-hand-side is a sum of odd numbers. A sum of odd numbers can be even only if it is the sum of an even number of odd numbers. Hence, the number of vertices in V_o must be even.

   (c) No. The following graph provides a counter-example as only one vertex b has odd indegree.



3. Let us first show that there is an $O(|V|)$ algorithm for determining whether a directed graph contains a sink. The algorithm begins with testing whether there is a directed edge (u,v):

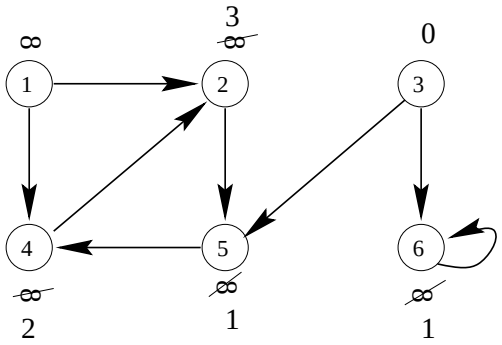
   - If there is an edge from u to v, then u is NOT sink.
   - If there is no edge from u to v, then v is NOT sink.

   Therefore, by $|V| - 1$ tests, we can eliminate all but one vertex.

   The second step of the algorithm is to make sure this one is a sink. It takes $|V| - 1$ tests for edges coming in (in-degree), and $|V| - 1$ tests for edges going out (out-degree).

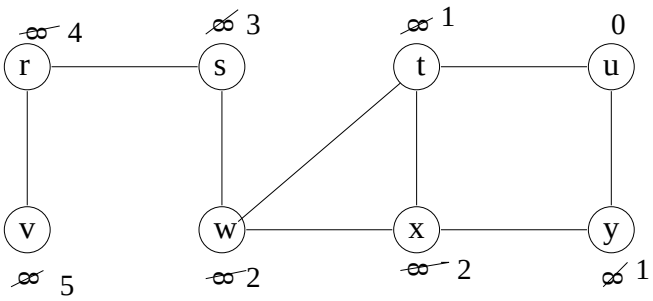
   The run time of the algorithm is then $|V| - 1 + |V| - 1 + |V| - 1 = 3|V| - 3 = O(|V|)$.

4. The result of running/illustrating breadth-first search



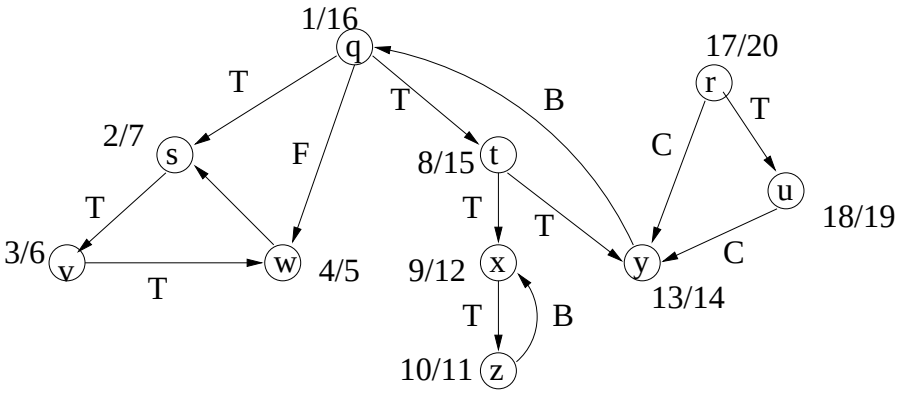
Q: 3 5 6 4 2

5. The result of running/illustrating breadth-first search



Q: u t y w x s r v

6. The result of running/illustrating depth-first search

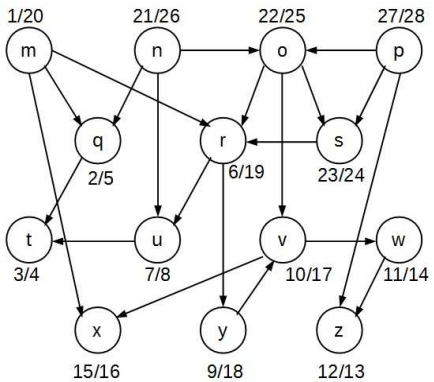


7. The comments in the following pseudocode show the changes to DFS to assign the desired *cc* label to the vertices.

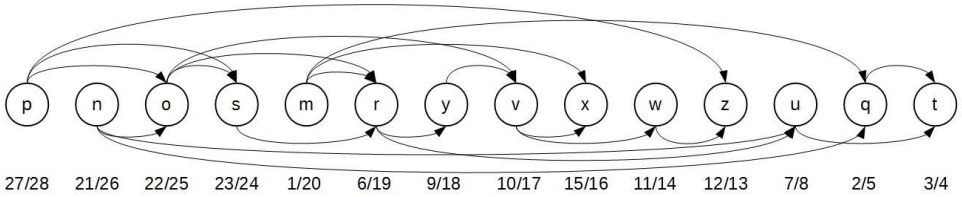
```
DFS(G)
for each vertex u ∈ V
    color[u] = white
    π[u] = nil
endfor
time = 0
counter = 0                                          // new counter
for each vertex u ∈ V
    if color[u] = white
        counter = counter + 1                        // increment counter
        DFS − Visit(u, counter)                      // pass counter argument
    endif
endfor

DFS-Visit(u, counter)                                // counter is argument
color[u] = gray
cc[u] = counter                                      // label the vertex
time = time + 1
d[u] = time
for each v ∈ Adj[u]
    if color[u] = white
        π[v] ← u
        DFS − Visit(v, counter)                      // pass unchanged counter
    endif
end for
color[u] = black
time = time + 1
f[u] = time
```

8. After running DFS, we generate the following graph



The above graph can be shown in the following topologically sorted graph, with its vertices arranged from left to right in order of decreasing finishing time. All directed edges go from left to right. :



9. An undirected graph is acyclic (i.e., a forest) iff a DFS yields no back edges. The following is the proof.

   - Argue that acyclic ⇒ no back edges – Trivial (because back edge ⇒ acyclic).
   - Argue that no back edge ⇒ acyclic, because
     no back back edge ⇒ only tree edges (by Theorem 23.9) ⇒ forest ⇒ acyclic.

   Thus can run DFS: if find a back edge, there is a cycle.

   Time $O(V)$, (not $O(V+E)$!). If ever see $|V|$ distinct edges, must have seen a back edge because in acyclic (undirected) forest, $|E| \leq |V| - 1$.

10. Let us identify the sets V_1 and V_2 with the colors RED and BLUE. We perform a DFS on the graph and color alternate levels of the DFS tree as RED and BLUE (clearly they must have different colors). Then the graph is bipartite **iff** there is no monochromatic edge. This can be checked during DFS itself as such an edge must be a back-edge, since tree edges are never monochromatic by construction and DFS on undirected graphs produces only tree and back edges.