```
VER1(G)
Comment: Go through the adjacency list
1  Some initialization here ...
2  for each vertex u ∈ V do
3      for each vertex v ∈ Adj[u] do
4          Process edge (u, v)
```

```
VER2(G, w)
1  Initialize matrix of values M[i, j]
2  for k ← 1 to n do
4      for i ← 1 to n do
5          for j ← 1 to n do
6              Process recurrence for M[i, j]
```

```
VER3(G, w, s)
1  Initialize priority values
2  Q ← ∅
3  while Q is not empty do
4      u ← EXTRACTMIN(Q)
5      for each vertex v ∈ Adj[u] do
6          Process edge (u, v) with weight w(u, v)
```

```
VER4(G, u)
1  color[u] ← gray
2  VISITVERTEX(u)
3  for each vertex v ∈ Adj[u] do
4      VISITEDGE(u, v)
4      if color[v] = white then
5          π[v] ← u
6          VER4(G, v)
9  color[u] ← BLACK
```

Figure 1: Graph Algorithms

1. Compute in-degree and out-degree of a given directed graph.
2. Compute the **transpose** of a directed graph.
3. Compute the **complement** of an undirected graph.
4. Compute the **square** of a directed graph.
5. Find an algorithm that can help you get out of a maze given a sufficiently large number of pennies.
6. Design an efficient algorithm to update the MST if
   (a) the weight of an edge not in the tree decreases
   (b) the weight of a tree edge increases
   (c) an edge is added to the graph with a given weight
   (d) a node is deleted from the graph along with all incident edges
   (e) a node is inserted into the graph along with some additional edges
   (f) weight of every edge is increased by amount d.
   (g) weight of every edge is decreased by amount d.
   (h) if you subdivide an existing edge and introduce a new vertex on it.

7. Solve the previous problem, but replacing MST with the shortest path tree for a specific source vertex s.

8. What is the time complexity of each graph algorithm if the edge weights come from a fixed range of integers $[1..c]$ (c is a fixed constant).

9. Show how to modify Floyd-Warshall's algorithm to compute reachability between every pair of vertices in a directed graph. Is this the most efficient way to compute reachability?

10. Short Questions

    (a) What is better for sparse graphs – adjacency lists or adjacency matrices?

    (b) What is the time complexity of checking if an edge (u, v) exists in the two representations?

    (c) Argue that BFS is the same as Dijkstra's algorithm for unweighted graphs.

    (d) What is the difference betwee the following: *DFS-tree*, *BFS-tree*, *SpanningTree*, *MST*, *SP-tree*

    (e) Why do you need to use the Disjoint-Set data structure to implement Kruskal's algorithm?

    (f) Use Theorem 23.1 to argue that Prim's algorithm is correct.

    (g) Is a minimum-weight edge always part of a MST?

    (h) Is the maximum-weight edge always not missing from a MST?

    (i) Explain why subpaths of shortest paths must be shortest paths too.

    (j) Explain how Dijkstra's algorithm is a form of DP.

    (k) Explain how Floyd-Warshall's algorithm is a form of DP.

    (l) Define the classes $\mathcal{P}, \mathcal{NP}, \mathcal{C}\wr - \mathcal{NP}, \mathcal{NP} - Complete$.

    (m) Define a reduction and explain why it is useful to prove $\mathcal{NP}-$Completeness.

    (n) What was the first problem ever shown to be $\mathcal{NP} - Complete$.

    (o) What is a polynomially-verifiable problem? How is is different from polynomially-solvable problems? Give an example of each.

    (p) What is 2-SAT? Is it $\mathcal{NP} - Complete$?