Algorithms (COT 6405): Solutions 10

Problem 1
Write algorithms for converting (a) an adjacency-list representation of a graph into an adjacency matrix and (b) an adjacency matrix into adjacency lists.

We denote the adjacency list of a vertex u by *Adj-List*$[u]$, and the adjacency-matrix element for vertices u and v by *Adj-Matrix*$[u, v]$. The time complexity of both algorithms is $\Theta(V^2)$.

(a) Converting adjacency lists into a matrix.

```
LISTS-TO-MATRIX(G)   ▷ G is represented by adjacency lists
for each u ∈ V[G]
    do for each v ∈ V[G]
            do Adj-Matrix[u, v] ← 0
        for each v ∈ Adj-List[u]
            do Adj-Matrix[u, v] ← 1
```

(b) Converting an adjacency matrix into lists.

```
MATRIX-TO-LISTS(G)   ▷ G is represented by an adjacency matrix
for each u ∈ V[G]
    do initialize an empty list Adj-List[u]
        for each v ∈ V[G]
            do if Adj-Matrix[u, v] = 1
                    then add v to Adj-List[u]
```

Problem 2
Suppose that G is an undirected graph, and you need to check whether G has cycles. Design an algorithm that returns TRUE if G is acyclic, and FALSE if G has cycles.

The key observation is that an acyclic undirected graph has at most $V-1$ edges. To determine whether a graph G is acyclic, we first count its edges. If the edge counter reaches V, we immediately return FALSE without counting the rest of edges. On the other hand, if the number of edges is less than V, we apply DFS to search for cycles. In either case, the overall running time is $O(V)$.