Lab08 - Matrix

Problem 1:

We have focused primarily on *time complexity* in this course, but when choosing data structures, space complexity is often as important of a constraint. Given an adjacency matrix, what is the 'space complexity' in Big-O. That is, given *n* nodes, how much space (i.e. memory) would I need to represent all of the relationships given. Explain your response.

ANS

For each vertex, it takes *n* blocks of space, and its adjacency matrix takes n^2 blocks of space to represent all of the relationships given.

Therefore, given n nodes, the total space required by the adjacency matrix is $n^2 + n$.

The space complexity in Big-O is n^2.

Notice that undirected adjacency matrix is symmetrical, one of its diagonals from left top to right bottom is always zero, so it takes $\frac{n(n-1)}{2}$ blocks of space to represent all of the relationships given.

The Big-O is also n^2.

Problem 2:

Will it ever make sense for ROWS != COLUMNS in an adjacency matrix? That is, if we want to be able to model relationships between every node in a graph, must rows always equal the number of columns in an adjacency matrix? Explain why or why not.

ANS

it doesn't make sense if ROWS != COLUMNS in an adjacency matrix.

An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph[1].

All the relationships between the nodes is n^2, so ROWS must equals to COLUMNS.

[1] Adjacency matrix https://en.wikipedia.org/wiki/Adjacency_matrix

Problem 3:

Can you run topological sort on a graph that is undirected?

ANS

We can’t run topological sort on a graph that is undirected.

Topological sort on a graph is a linear ordering of its vertices such that for every directed edge *uv* from vertex *u* to vertex *v*, u comes before v in the ordering.

There is at least one loop in undirected graph, the relationship between the nodes is not linear. When we try to run tsort with such graph, It is fail to output all relationships in graph.

Problem 4:

Can you run topological sort on a directed graph that has cycle?

ANS

We can’t run topological sort on a directed graph that has cycle.

It has the same reason as #3. Because there is at least loop in graph, the relationship between the data is inconsistent.

Problem 5:

For question number 4, how would you know you have a cycle in a graph? What algorithm or strategy could you use to detect the cycles? **Hint** we have already learned about this traversal.

ANS

We could use DFS algorithm to detect whether there is a cycle in the graph.

*Here's **an excerpt** from wikipedia's pseudocode of Topological sorting:*

```
L ← Empty list that will contain the sorted nodes
while exists nodes without a permanent mark do
    select an unmarked node n
    visit(n)

function visit(node n)
    if  n has a permanent mark then
        return
     if n has a temporary mark then
        stop   (not a DAG)

     mark n with a temporary mark

     for each node m with an edge from n to m do
        visit(m)

     remove temporary mark from n
     mark n with a permanent mark
     add n to head of L
```

For each node *n,* we consider all other nodes *m* which *n* has directed-edges in to. Then we add *n* to list *L.* If the graph has a cycle which means there is a node *m* has directed-edges in to *n.* Since each edge and node can only be visited once. We can definitely tell if the graph has a cycle or not.

[2] Topological sorting https://en.wikipedia.org/wiki/Topological_sorting