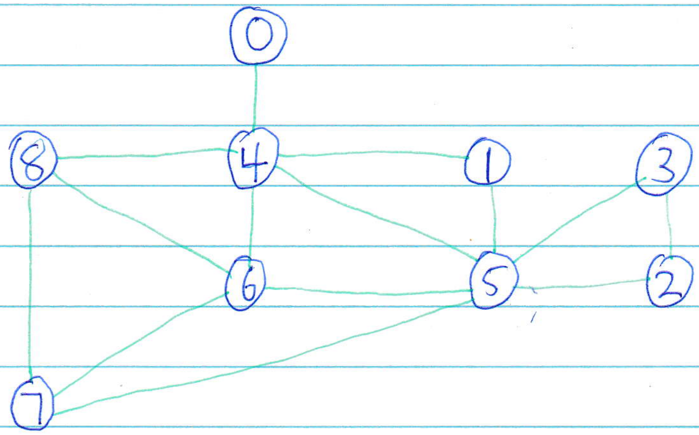
CSC 225 Assignment 4
Nelson Dai - V00815253
July 19/2016

1.



Adjacency list representation:

0 → 4 /
1 → 4 → 5 /
2 → 3 → 5 /
3 → 2 → 5 /
4 → 0 → 1 → 5 → 6 → 8 /
5 → 1 → 2 → 3 → 4 → 6 → 7 /
6 → 4 → 5 → 7 → 8 /
7 → 5 → 6 → 8 /
8 → 4 → 6 → 7 /

Adjacency matrix representation:

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	1	1	0	0	0
2	0	0	0	1	0	1	0	0	0
3	0	0	1	0	0	1	0	0	0
4	1	1	0	0	0	1	1	0	1
5	0	1	1	1	1	0	1	1	0
6	0	0	0	0	1	1	0	1	1
7	0	0	0	0	0	1	1	0	1
8	0	0	0	0	1	0	1	1	0

2. DFS: 0→4→1→5→2→3→6→7→8
   BFS: 0→4→1→5→6→8→2→3→7

3. As we told that a tree T with n vertices has n-1 edges, then $F = \bigcup_{i=1}^{K}$ is the forest - a union of trees. n_o is the number of vertices of each tree → $|V(F)| = \sum_{i=1}^{K} n_o = n$. since forest is combined by trees, we have $n_o - 1$ edges for T_o → $|E(F)| = \sum_{i=1}^{K} (n_o - 1) = n - K$.

4.
```
initialize a empty List L
initialize a stack S
push all vertices with no incoming edges into stack
while stack is not empty do        if stack has more than one item
                                     stop return false (the topologic order
                                                        is not unique)
  V ← stack.pop()
  add v to L
 for all the vertices w with an edge e from v to w do
        if there is more than one 'w'
                stop and print "The topological order is not unique."
    remove edge e from G
    if w has no other incoming edge then
     push w into stack
    if G has edges left then
    return false ( G is not topological ordered)
    else
    return L ( the topologic order is uniaue)
```

```
5. Put all vertices into a list L
   color first vertex to white
   Creat a queue of vertex number and enqueue the first vertex
   while queue is not empty
       dequeue a vertex from queue
       for all non-colored adjacent vertices
           if an edge from u to v exists and v is not colored
               assign alternate color to v
           else if an edge from u to v exist and v is colored same as u
               return false
       return true
```

Since we are useing BFS here, the running time is $O(n+m)$