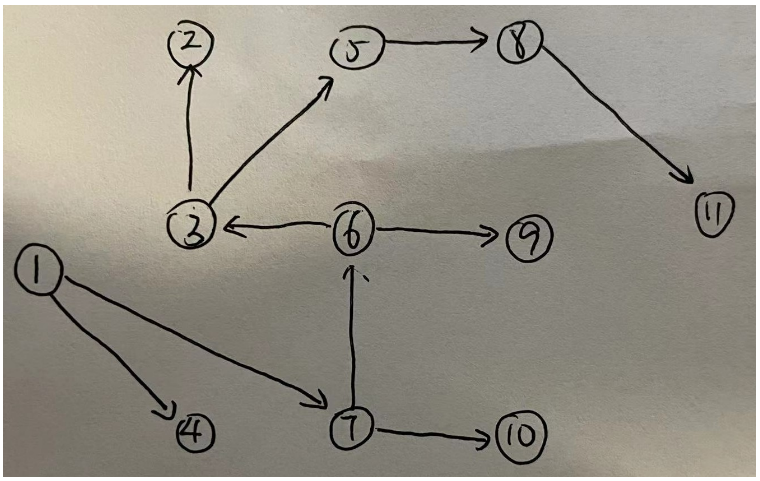
Problem 1

(a)

The BFS tree is shown as below, And here is the table of $d(i)$ and $p(i)$.

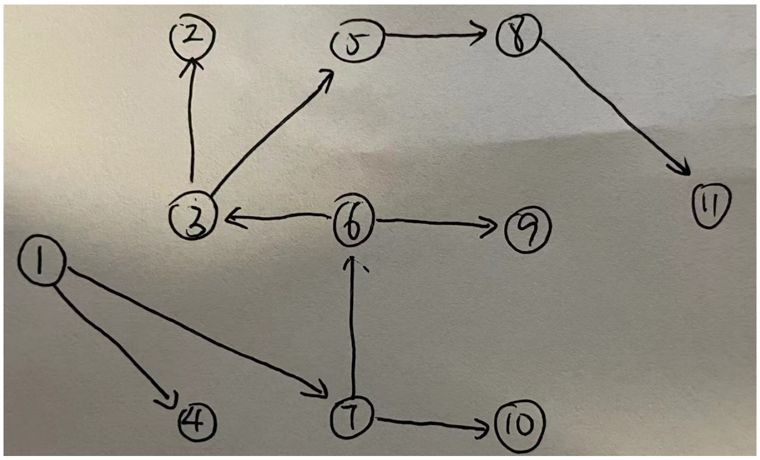


node	d(i)	p(i)
1	0	-
2	4	3
3	3	6
4	1	1
5	4	3
6	2	7
7	1	1
8	5	5
9	3	6
10	2	7
11	6	8

And the order of the nodes leave the queue is 1, 4, 7, 6, 10, 3, 9, 2, 5, 8, 11.

(b)

The DFS tree is shown as below,

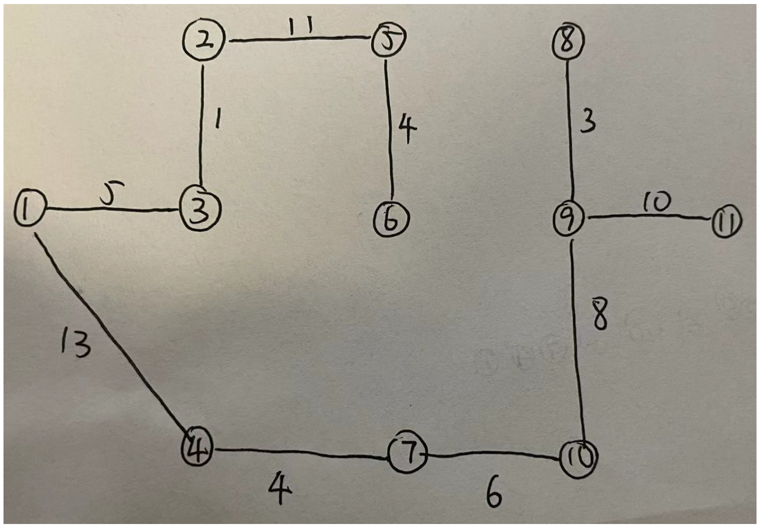


And the order of the nodes enter the stack is 1, 4, 7, 6, 3, 2, 5, 8, 9, 11, 10.

The order of the nodes leave the stack is 4, 2, 9, 10, 11, 8, 5, 3, 6, 7, 1.

If we reverse the second list, it will be 1, 7, 6, 3, 5, 8, 11, 10, 9, 2, 4. And it will be a topological sort. Because if the elements at the bottom of the stack wants to leave it, the element above it must be first left. And there must have a way between them. So it must be a topological sort.

Problem 2



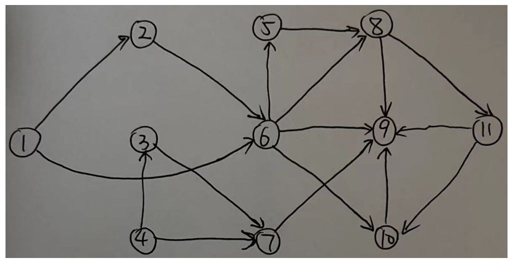
The edge in order is (2,3), (8,9), (4,7), (5,6), (1,3), (7,10), (9,10), (9,11), (2,5), (1,4).

If we use Prim's algorithm, the tree will be the same and the edge in order will be (1,3), (2,3), (2,5), (5,6), (1,4), (4,7), (7,10), (9,10), (8,9), (9,11).

Problem 3

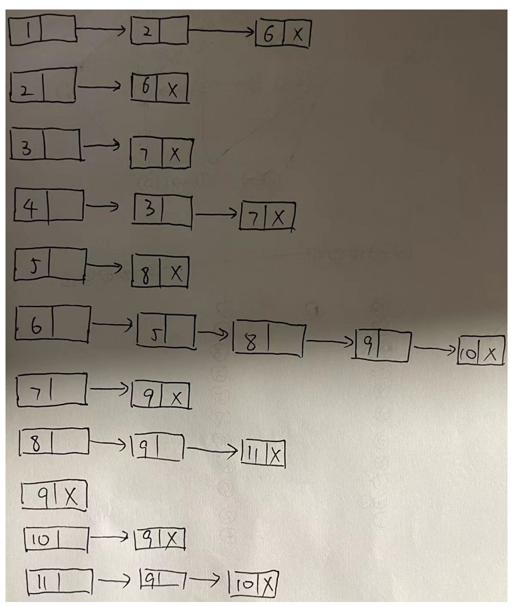
(a)

The graph is shown as below,



(b)

The adjacency list is shown as below,



(c)

Yes, it is a DAG, no cycle appears in this graph.

Problem 4

I cannot find an algorithm with worst case O(m), I think that is impossible. And my algorithm is shown as below. Input V, E
Output Bool
randomly choose a u from V
let u.color = black
if (u,v) is in E
if (v.color=u.color) return False
else v.color = different with u.color
do this until all the edges be used
The final time complexity is O(n+m). n is the number of vertical and m is the number of edge.

Problem 5

(a)

For Q_3, it has 12 edges and 8 vertices. The minimum cycle is length 4. $2n - 4 = 12$ and it satisfy $e \leq 2n - 4$. So Q_3 is a planar graph.

(b)

For Q_4, it has 32 edges and 16 vertices. The minimum cycle is length 4. $2n - 4 = 28$ and it does not satisfy $e \leq 2n - 4$. So Q_4 is not a planar graph.

Problem 6

Input minimum spanning tree T with all vertices and edges, new added vertex and edges
Output new minimum spanning tree $T^{'}$
Sort the weight of all w_{nvi} in ascending order
Add new edges with the minimum weight to the spanning tree
if (new cycles appears) delete the maximum weight edges in this cycle
else (add this edges to the spanning tree and go to the next edge)
until all new edges have been used
If we assume there has n new edges and m edges in the original spanning tree. The worst time complexity is O(nm) and the average time is O(n+m).