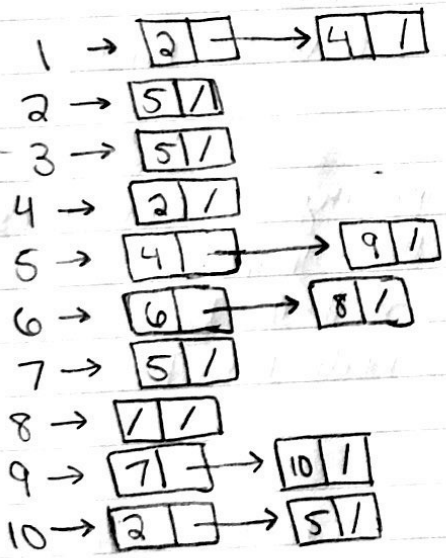
I pledge my honor that I have abided by the Stevens Honor System.

Suzy Shailush
Homework 3

1.

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	1	0
6	0	0	0	0	0	1	0	1	0	0
7	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	1
10	0	0	1	0	1	0	0	0	0	0

2.



3. 1, 2, 4, 5, 9, 7, 10, 3, 6, 8

4. 1, 2, 5, 4, 9, 7, 10, 3, 6, 8

5. n = number of vertices
a) $O(n^2)$
b) $O(n^2)$

6. n = number of vertices
m = number of edges
a) $O(n + m)$
b) $O(n + m)$

use adjacency lists for large & sparse graphs.

7. An adjacency list is a clear winner in the efficiency of your algorithm because of how much less space they take up than adjacency matrices. Adjacency matrices store every possible edge, including ones that don't exist. Adjacency lists only store the ones that do.

8. You can use a BFS to find cycles in an undirected graph by checking every visited vertex to see if there is an adjacent vertex that has already been visited that is not the original vertex's parent. If such a vertex exists, there is a cycle in the graph.

9. Both BFS and DFS take a time complexity of O(vertices + edges) to find a cycle in an undirected graph. However, DFS is more memory efficient than BFS. You can backtrack quicker to see if a node has been visited before. So, DFS is faster.

10. A topological is not possible on this graph because this graph contains cycles, such as between 2, 4, and 5. This is because you will never be able to satisfy the conditions for removing those vertices (must have an in-degree of 0) one at a time.

11. 1, 4, 2, 5, 9, 7, 10, 3, 6, 8