1.

a. Lists are vertical

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	20			16	44	94	12		13	
				5	88	39	23			
					11					

b.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
11	39	20	5	16	44	88	12	23	13	94

c.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
20		16	11	39	44	88	12	23	13	94

d. Assuming the mod11 is applied last

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
11	23	20	16	44	5	94	12	88	13	39

2.

```
put(k,v):
    i <- h(k)
    j <- 0
    while A[i] != NULL or A[i] = SpecialMarker do
        if A[i].key = k then
                A[i] <- (k,v) //replace the old value with same key
        j <- j+1
        i <- h(k) + j*j
    A[i] <- (k,v) //found a nice empty index

get(k):
    i <- h(k)
    j <- 0
    while A[i] != NULL do
        if (A[i].key = k) then
                return A[i]
        j <- j+1
        i <- h(k) +j*j
    return NULL //not found

remove(k):
    i <- h(k)
```

```
j <- 0
while A[i] != NULL do
    if (A[i].key = k) then
            temp <- A[i]
            A[i] <- SpecialMarker //a non-NULL key that will never be used by regular values to
represent a removed element
            return temp
    j <- j+1
    i <- h(k) + j*j
return NULL //key not found
```

3.

Algorithm matrixDFS(G,v,P)

Input: A graph G,with n vertices labeled 0,...,n−1,represented as an adjacency matrix, a starting vertex v, and an array C of size n that is initially 0s

Output: An integer array of size n, containing the preorder labelling of the vertices.

1. P[0] <- v
2. C[v] <- 1
3. for i from 0 to n do
4.   if G[v][i] is not 0 then
5.   if C[i] = 1 then skip
6.   Else append DFS(G,i,C) to P
7. return P

4.

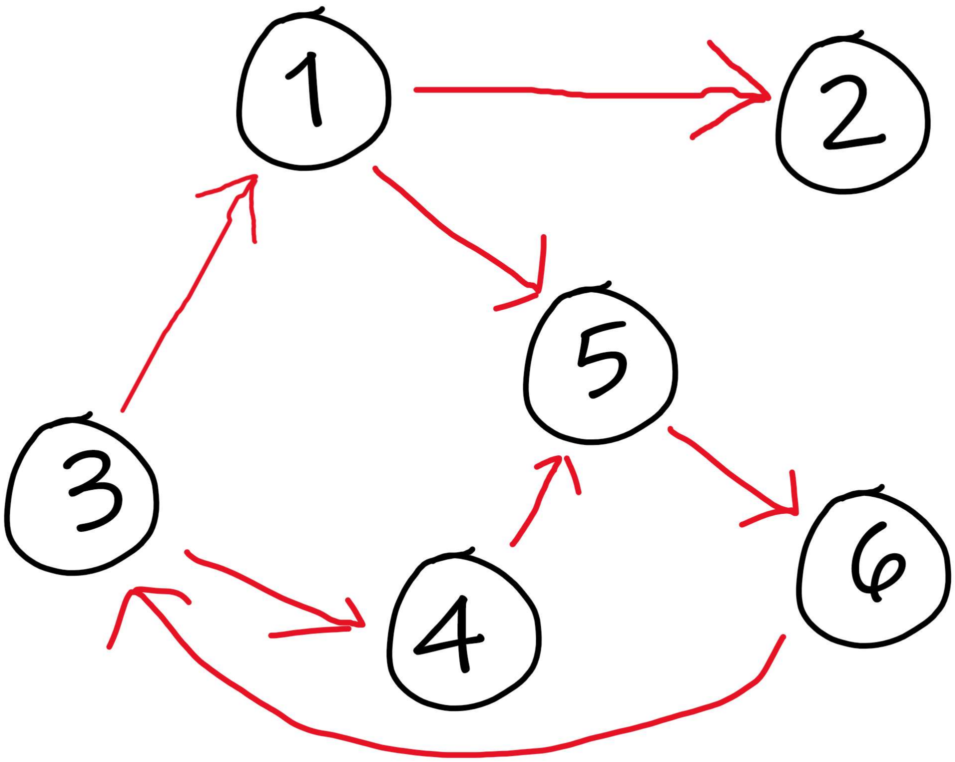
1. Takes 1 unit of time to append
2. Takes 1 unit of time
3. Loops n times per iteration
4. Takes 1 time per loop
5. Takes 1 unit of time to check
6. Recursive call
7. Takes 1 unit of time

This algorithm is O(n^2) in the best and worst case, because it will always conduct n loops over n iterations.

The adjacency list representation has a best/worst case of O(|V| + |E|), where |V| is the number of vertices in the graph and |E| is the number of edges in the graph.

5.

a.



G_0 =

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	0	0	0	0	0	0
3	1	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	0	0	1	0	0	0

Rows are from, columns are to.

b.

G_1 =

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	0	0	0	0	0	0
3	1	1	0	1	1	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	0	0	1	0	0	0

G_2 has no changes since 2 has no outgoing edges

G_3 =

	1	2	3	4	5	6
1	0	1	0	0	1	0

2	0	0	0	0	0	0
3	1	1	0	1	1	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	1	1	1	1	1	0

G_4 has no changes since its only incoming edges, (3,4) and now (6,4), already have outgoing edges towards 5.

G_5 =

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	0	0	1	1
2	0	0	0	0	0	0
3	1	1	0	1	1	1
4	0	0	0	0	1	1
5	0	0	0	0	0	1
6	1	1	1	1	1	0

G_6 =

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	1	1	1	1
2	0	0	0	0	0	0
3	1	1	0	1	1	1
4	1	1	1	0	1	1
5	1	1	1	1	0	1
6	1	1	1	1	1	0