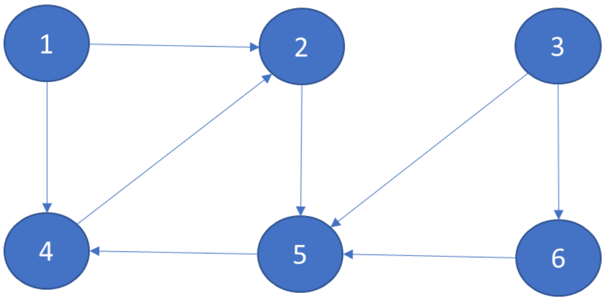
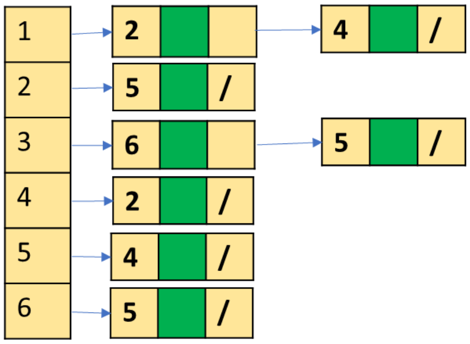
HOMEWORK-10
Total Points: 65

Q1. [20 Points] Let’s consider the adjacency lists representation for directed graphs. The adjacency lists entries correspond to outgoing edges. For graph G, an adjacency list node also contains a 1024-byte payload for every edge. The payload is the same for a given edge, whether we treat it as an incoming edge or the outgoing edge. The type of a node in the adjacency list node is the following:

```
struct node {
  int vertex;
  char payload[1024];
  struct node *next;
};
```

The array of vertices simply stores a reference to the head of the corresponding adjacency list. How do you efficiently store incoming edges too in the adjacency list of G without replicating the payload? What would be the type of adjacency list node in your design? Also, draw the adjacency lists generated using your algorithm for the following graph.

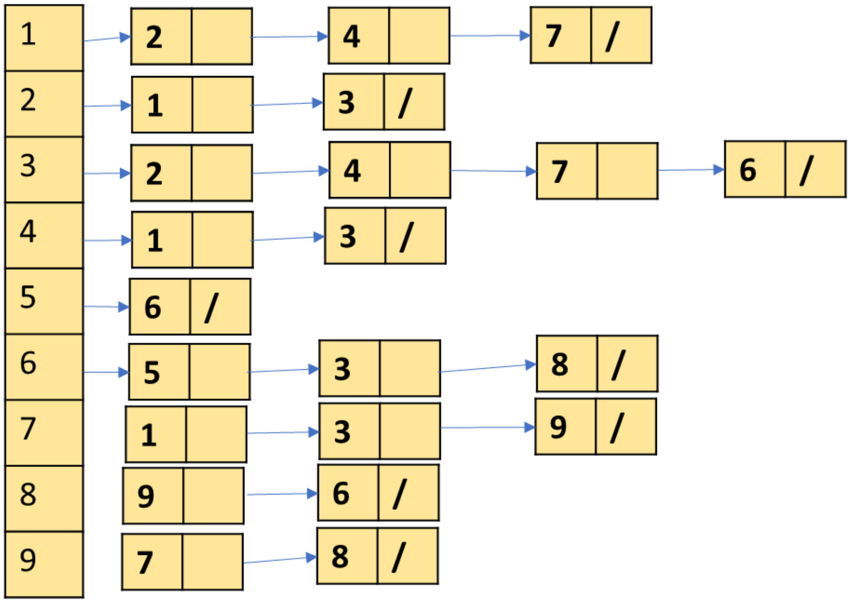




Q2. [20 points] Give an algorithm for the findPath procedure that takes a graph G, a source vertex s, and a length l, and returns true if a non-circular path of length l from s to any other vertex exists; otherwise, it returns false.

Q3. [10 Point] What would be the time complexity of the BFS algorithm if we use an adjacency matrix instead of adjacency lists to store the edges? Justify your answer.

Q4. [15 Points] Write down the state of the entire queue after every insertion and deletion in the queue during the BFS algorithm running for vertex 1 on the adjacency list shown on the next page. Notice that an element of the queue is a pair of vertex and its distance from the source.


1
2
4
7 /
2
1
3 /
3
2
4
7
6 /
4
1
3 /
5
6 /
6
5
3
8 /
7
1
3
9 /
8
9
6 /
9
7
8 /