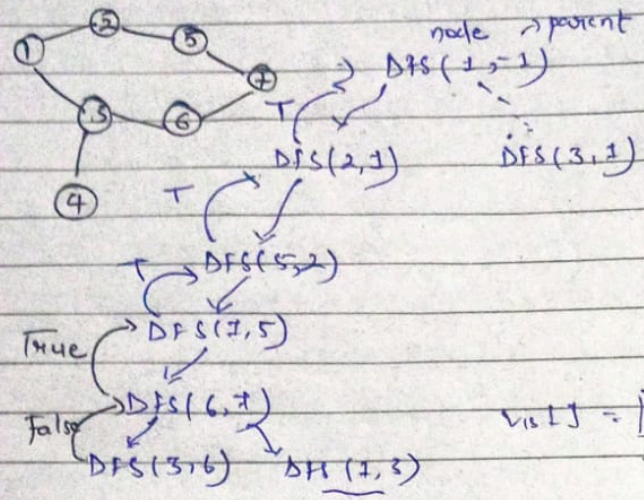
Graph_Day_3

• Detect the cycle in undirected graph {DFS}



adj list
1 → {2,3}
2 → {1,5}
3 → {1,4,6}
4 → {5}
5 → {2,7}
6 → {3,7}
7 → {5,6}

vis[] = | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
(1 1)

Pseudocode:

```
dfs(node, parent) {
    vis[node] = 1

    for(auto it : adj[node]) {
        if(vis[it] == 0) {
            if(dfs(it, node) == true)
                return true;
        }
        else if(it != parent)
            return true;
    }
    return false;
```

Code: def has_cycle(self, v, visited, parent):

- Mark current node visited — visited[v] = True
- Recursively visit all unvisited neighbors — for neighbor in self.graph[v]:
- Pass the current node v as the parent to the next recursive call — if not visited[neighbor]:
  if self.has_cycle(neighbor, visited, v):
  return True
  elif neighbor != parent:
  return True
  return False