1. Find Cycle in Directed Graph

a. https://leetcode.com/problems/course-schedule/

```
bool isCycle = false;
  void dfs(int u)
  {
      visited[u] = true;
      revstack[u] = true;
      for(auto v : list[u])
      {
          if(!visited[v])
              dfs(v);
           else if(revstack[v])
              isCycle = true;
      }
      revstack[u] = false;
  }
```

2. Find Cycle in Undirected Graph

a. geeksforgeeks.org/detect-cycle-undirected-graph/

b. Idea is while traversing using dfs if adjacent node is already visited and other than parent than its cycle

3. Topological Sort

a. DFS

i. Add last line of dfs algo => stack.push(u)

ii. Pop all element of stack to vector this vector contains topological order

b. BFS

i. https://www.youtube.com/watch?v=_LuIvEi_kZk

```
TopSort(G)
    for each u in V
        u.inDegree = 0
    for each u in V
        for each v such that u → v
            v.inDegree++
    Q = new Queue( )
    for each u in V
        if (u.inDegree == 0)
            Q.enqueue(u)
    while Q isn't empty
        u = Q.dequeue( )
        put u next in the topological ordering
        for each outgoing edge u → v
            v.inDegree--
            if (v.inDegree == 0)
                Q.enqueue(v)
```

ii. If visited_nodes != total_nodes

Cycle in graph [since nodes which are in cycle will never have indegree zero they will never come in queue]

c. To find longest topological order length

i. https://leetcode.com/problems/longest-increasing-path-in-a-matrix/discuss/288520/BFS-Implemented-Topological-Sort

ii. USE bfs version 2

1. while(! q.empty())

Lev++

Size = q.size()

for(i=0;i<size;i++)

// //

3. Kruskal

https://cses.fi/problemset/task/1675/

```
vector<vector<ll>> edges;
    while(m--){
        cin >> a >> b >> c;
        edges.push_back({c,a,b});
        edges.push_back({c,b,a});
    }

    sort(edges.begin(),edges.end());

    ll cost = 0, n_edges = 0, i  = 0;
    while(n_edges < n-1 && i < edges.size()){
        auto edge = edges[i++];

        if(dunion(edge[1], edge[2])){
            n_edges++;
            cost += edge[0];
        }
    }
    if(n_edges == n-1)
        cout << cost << endl;
    else
        cout << "IMPOSSIBLE";
```

4. Dijkastra

```
 void dijakastra(int u){
        priority_queue<pair<int,int>, vector<pair<int,int>>,
greater<pair<int, int>>> q;
        q.push({0, u});
        dist[u] = 0;
        while(!q.empty()){
            int w = q.top().first;
            int u = q.top().second;
```

```
            q.pop();
            for(auto x : list[u]){
                int v = x.first; int w = x.second;
                if(dist[v] > dist[u] + w){
                    dist[v] = dist[u] + w;
                    q.push({dist[v], v});
                }
            }
        }
    }
```

https://practice.geeksforgeeks.org/problems/minimum-cost-path/0
https://leetcode.com/problems/swim-in-rising-water/
https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/shortest-path-revisited-9e1091ea/description/ (dijkastra with k relax)

5. Prims Algo

https://www.hackerrank.com/challenges/primsmstsub/problem?isFullScreen=true

```
q.push({0, start});
   while(q.size()){
       auto x = q.top();
       q.pop();
       int u = x.second;
       int ue = x.first;
       cout << ue << " " << u << endl;
       if(!visited[u])
           cost += ue;
       visited[u] = true;
       for(auto y:graph[u]){
           int v = y.first;
           int ev = y.second;
           if(!visited[v])
               q.push({ev, v});
       }
   }
   return cost;
```

6. Bellman Ford

https://leetcode.com/problems/cheapest-flights-within-k-stops/
https://cp-algorithms.com/graph/finding-negative-cycle-in-graph.html (print negative cycle)
https://cses.fi/problemset/result/1065726/

```
        vector<int> dist(n, 1e9), tdist;
        dist[src] = 0;
        for(int i=0;i<v-1;i++){
```

```
            tdist = dist;
            for(int j=0;j<edges.size();j++){
                int from = edges[j][0];
                int to = edges[j][1];
                int weight = edges[j][2];
                dist[to] = min(dist[to], tdist[from] +
weight);
            }
        }
```

7. Euclier Path / Cycle

a. Directed Graph
   i. A Directed Graph has an Euclier circuit
      1) All vertices with nonzero degree belong to a single strongly connected component.
         https://www.geeksforgeeks.org/connectivity-in-a-directed-graph/
      2) In degree is equal to the out degree for every vertex.
         https://www.geeksforgeeks.org/euler-circuit-directed-graph/ (existence)
   ii. A Directed Graph has an Euclier Path
      1) In degree is equal to the out degree for every vertex except 2 vertex.
      2) Single Component.
      3) https://cses.fi/problemset/result/1066523/

https://leetcode.com/problems/reconstruct-itinerary/

```
 void dfs(string u,  unordered_map<string, multiset<string>> & list,
vector<string> &ans){
            while(list[u].size()){
                string next = *list[u].begin();
                list[u].erase(list[u].begin());
                dfs(next, list, ans);
            }
            ans.push_back(u);
    }
Return reverse(ans.begin(),ans.end())
```

b. Undirected Graph
   i. An Undirected graph has an Euler circuit if and only if the degree of every vertex is even.
   ii. An Undirected graph graph has an Euler path if and only if there are at most two vertices with odd degree.

https://cses.fi/problemset/task/1691/

```
unordered_map<int, unordered_set<int>> graph;
vector<int> res;


void eucliar(int u){
    while(graph[u].size()){
        int v = *graph[u].begin();
        graph[u].erase(graph[u].find(v));
```

```
        graph[v].erase(graph[v].find(u));
        eucliar(v);
    }
    res.push_back(u);
}

bool findEucliear(int source, int m, vector<int>& degree){
    for(int u:degree)
        if(u & 1)
            return false;

    eucliar(source);

    return res.size() == m+1;
}
```

8. Strongly Connected Components

a. https://www.interviewbit.com/problems/strongly-connected-components/
b. Topological sort
c. Reverse the graph
d. One by one take output of a.
    i. Apply dfs of each and store in vector

9. Floyd warshal

a. Make diagonal 0 and set INF 9999 (not INT_MAX cause overflow)

```
for (k = 0; k < V; k++)
        for (i = 0; i < V; i++)
            for (j = 0; j < V; j++)
            {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                    dist[i][j] = dist[i][k] + dist[k][j];
            }
        }
    }
```

10. Bipartite Graph

a. If graph is 2 colorable than it is Bipartite
    i. https://leetcode.com/problems/is-graph-bipartite/
b. To color k color to graph
    i. https://leetcode.com/problems/flower-planting-with-no-adjacent/
    ii. https://leetcode.com/problems/possible-bipartition/ code here ****

```
bool dfs(int u, vector<int> list[], vector<int> &color, int col){
        color[u] = col;
```

```
        for(int i=0;i<list[u].size();i++){
            int v = list[u][i];
            if(color[v] == 0){
                if(!dfs(v, list, color, -col)) return false;
            }
            else if(color[v] == color[u])
                return false;
        }
        return true;
    }
```

11. Bridge

u -> v bridge if disc[u] < low[v]

```
    vector<int> list[100001], dicv, low;
    vector<vector<int>> bridges;
    int time = 0;

    void dfs(int u, int p){
        dicv[u] = low[u] = time++;
        for(int v:list[u])
            if(dicv[v] == -1){
                dfs(v, u);
                low[u] = min(low[u], low[v]);
                if(low[v] > dicv[u])
                    bridges.push_back({u, v});
            }
            else if(v != p)
                low[u] = min(low[u], dicv[v]);
    }
```

12. Articulation Point

Root node -> more than 1 child

Non root -> disc[u] <= low[v]

```
unordered_set<int> artipoints;
int ttime;
void dfs(int u, int pu){

    low[u] = dis[u] = ttime++;
    int child = 0;
    for(int v:graph[u]){
        if(v == pu) continue;

        if(dis[v] == -1){
            child++;
```

```
            dfs(v, u);
            if(pu != -1 && low[v] >= dis[u])
               artipoints.insert(u);
            low[u] = min(low[u], low[v]);
        }
        else
            low[u] = min(low[u], dis[v]);

    }
    if(pu == -1 && child > 1)
        artipoints.insert(u);

}
```

13. Largest Distance between any two node in n - array tree

a. www.interviewbit.com/problems/largest-distance-between-nodes-of-a-tree/
b. Find farthest node x from any of the node u of tree - this node will be one of the end point of largest diameter - using bfs / dfs
c. Now find farthest node y from given node x
d. Ans will be distance of y from x

14. Farthest Node dist from each node

a. https://cses.fi/problemset/task/1132/
b. From any node farthest dist can be either of diameter's end point

15. Stepping Numbers

a. https://www.interviewbit.com/problems/stepping-numbers/

```
          5
     54        56
  543 545  565  567
```

START from 0 to 9 generate each stepping no
Boundary case if last digit is 0 or 9

16. Find the centroid of a tree - undirected graph which has a almost same dis to all nodes

a. https://leetcode.com/problems/minimum-height-trees/
b. Delete leaves until there are utmost 2 nodes left

17. No of islands where same row or same columns are connected

a. https://leetcode.com/problems/most-stones-removed-with-same-row-or-column/

18. Number of distinct islands where same islands are not counted

a. https://www.geeksforgeeks.org/find-the-number-of-distinct-islands-in-a-2d-matrix/
b. Use set of vector of points
c. Do a dfs on each components and store its shape on set
d. To find shape of component subtract it from base point
e. Return set size

19. Parallel BFS in Graph

a. https://leetcode.com/problems/rotting-oranges/
b. Start with multiple 0 as a source and put these into queue
c. While loop

Update adjacent point if dis[adjx][adjy] > dist[x][y] + 1 and push into queue

20. Jump Game Bfs pattern

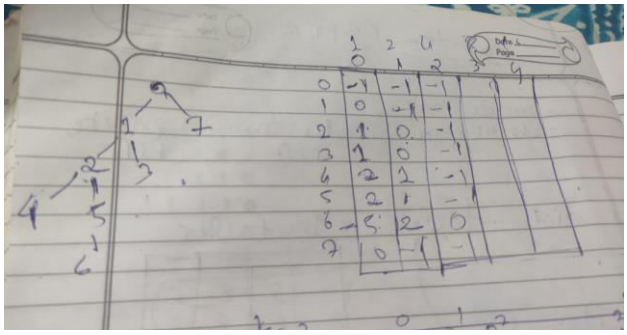
a. https://leetcode.com/problems/minimum-number-of-taps-to-open-to-water-a-garden/discuss/506853/Java-A-general-greedy-solution-to-process-similar-problems

```
 for(int i=0;i<nums.size()&&curReach<n;  curReach = farthestReach;){
            steps++;
            while(i < nums.size() && i <= curReach)
                farthestReach = max(farthestReach,nums[i++]);
            if(curReach == farthestReach)
                   return -1;
}
```

21. Binary lifting

a. Kth ancestor in binary tree

i. https://leetcode.com/contest/weekly-contest-193/problems/kth-ancestor-of-a-tree-node/
ii. Each no can be represented in sum of power of 2
iii. 4 ancestor is 2nd ancestor of 2nd ancestor



```
for(int j=0;j<logn;j++)
         dp[0][j] = 0;
     //fill zeroth column using dfs
     for(int j=1;j<logn;j++){
         for(int i=1;i<n+1;i++){
```

```
                dp[i][j] = dp[dp[i][j-1]][j-1];
            }
        }



//nth ancestor
for(int i=0;i<logn;i++){
        if((diff & (1 << i)) != 0){
            u = dp[u][i];
        }
    }
```

b. LCM using logn per query and nlogn precomputations

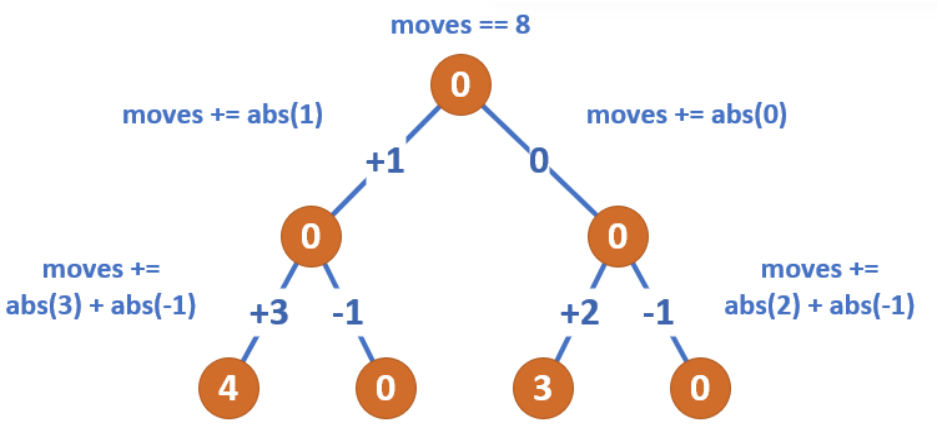
c. https://www.youtube.com/watch?v=02zM-QoKoPg
d. https://www.codechef.com/LRNDSA08/submit/ENOC1

```
Step 1: set both node at same level
Step 2:
      //lcm
   for(int i=logn-1;i>=0;i--){
        if(dp[u][i] != dp[v][i]){
            u = dp[u][i];
            v = dp[v][i];
        }
    }
    return dp[u][0];
```

Leetcode

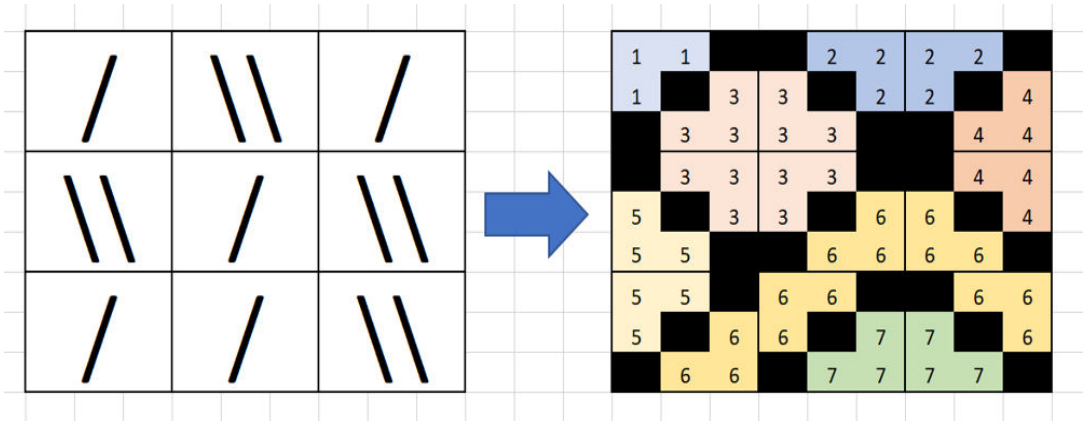
22. Distribute coins in tree

a. https://leetcode.com/problems/distribute-coins-in-binary-tree/



23. Find no of regions

a. https://leetcode.com/problems/regions-cut-by-slashes/discuss/205674/C%2B%2B-with-picture-DFS-on-upscaled-grid

b. 

24. Kth Nodes from target node in binary tree

a. https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/
b. Convert it to graph
c. Apply bfs from target node till kth level

25. Word ladder

https://leetcode.com/problems/word-ladder/discuss/40707/C%2B%2B-BFS
Don't traverse whole dictionary for next word, Rather try to replace all each char by all possible char and check if it is in dict

26. Number of good leaf nodes pair

https://leetcode.com/problems/number-of-good-leaf-nodes-pairs/submissions/
https://leetcode.com/problems/number-of-good-leaf-nodes-pairs/discuss/756198/Java-DFS-Solution-with-a-Twist-100-Faster-Explained

27. Jump game 5

a. https://leetcode.com/problems/jump-game-iv/
b. When we insert all values with same arr[i] make sure to clear mp[arr[i]] to avoid repeated traversal otherwise it will give tle

28. Cherry Pickup

a. https://leetcode.com/problems/cherry-pickup-ii/
b. https://leetcode.com/problems/cherry-pickup/

29. Nodes which does not end in cycle loop

a. https://leetcode.com/problems/find-eventual-safe-states/
b. Do dfs traversal and return if its in cycle

30. Other

1. https://www.interviewbit.com/problems/word-search-board/
2. https://www.interviewbit.com/problems/sum-of-fibonacci-numbers/
    a. Greedy works heres
3. https://www.interviewbit.com/problems/clone-graph/
4. https://leetcode.com/problems/as-far-from-land-as-possible/
5. https://leetcode.com/problems/shortest-path-with-alternating-colors/