

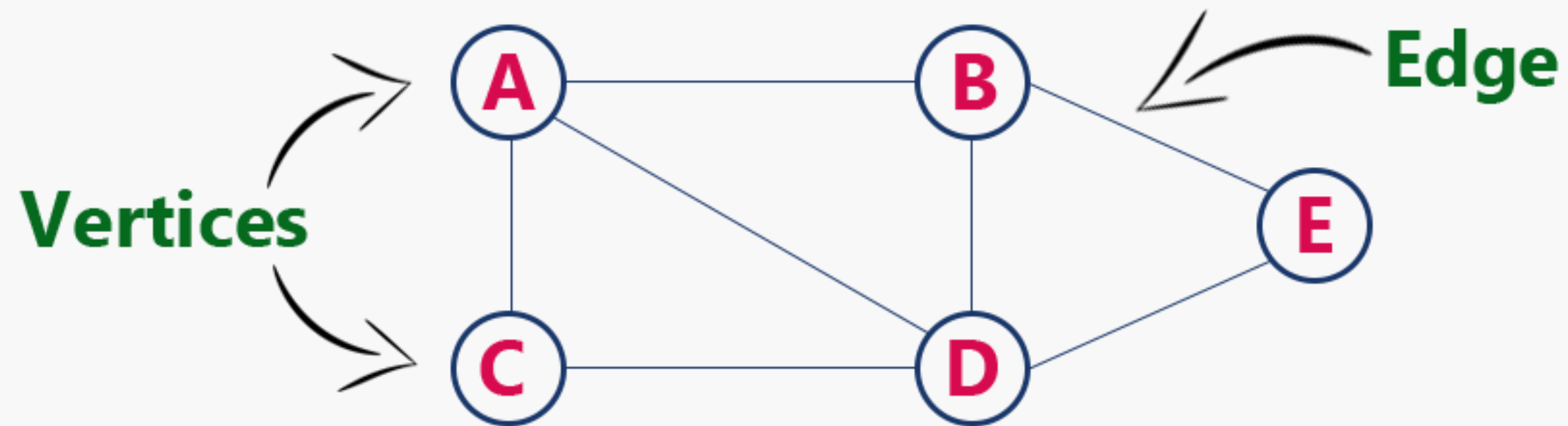


GRAPHS 101

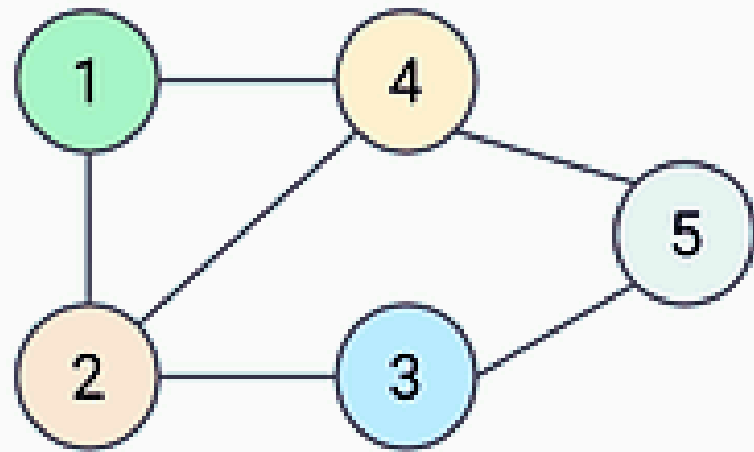
Bharat Singla
Yiheng

What are Graphs?

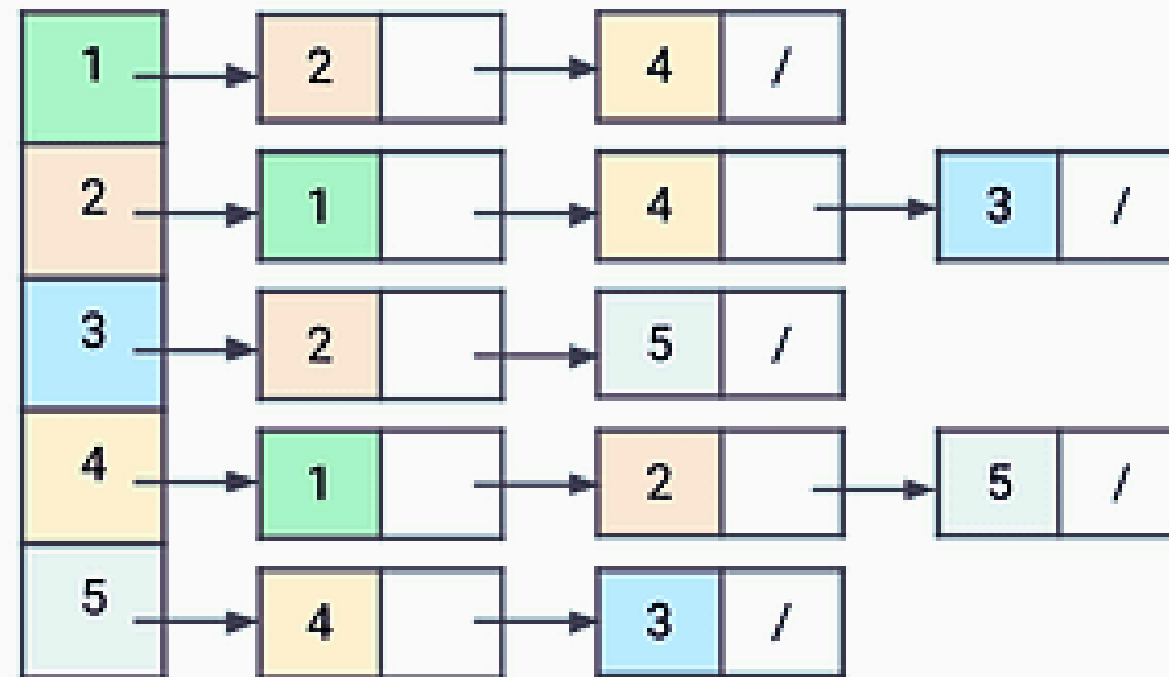
- A Graph is a Data Structure that represents relation among entities
- Graph: $G(V, E)$, where V is the set of Vertices and E is the set of Edges



Ways to Store/Represent a Graph



Undirected Graph

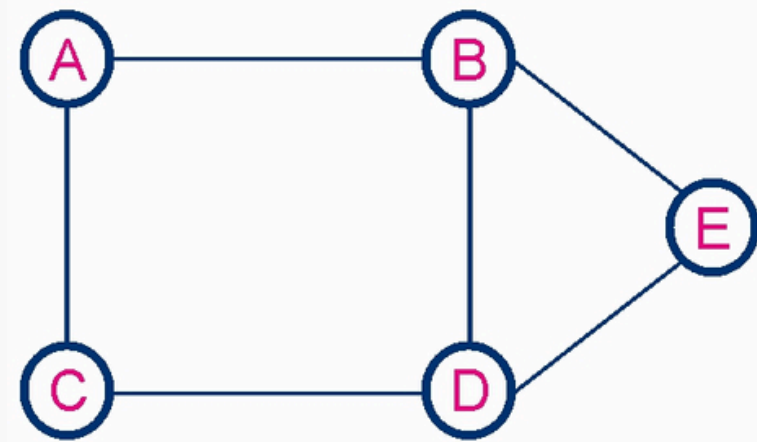


Adjacency List Representation

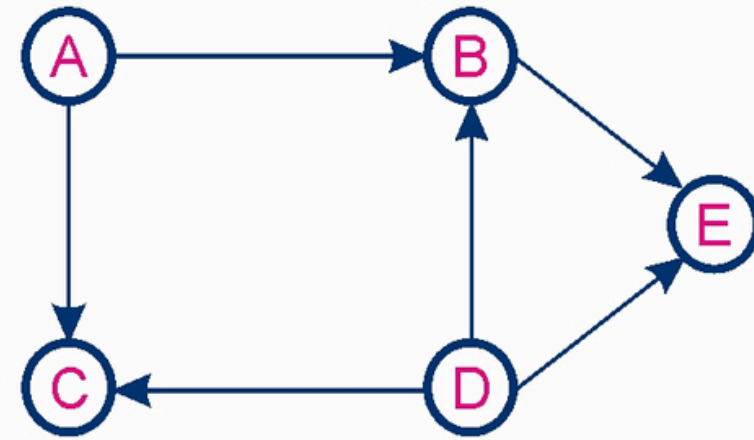
	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	0	1
4	1	1	0	0	1
5	0	0	1	1	0

Adjacency Matrix Representation

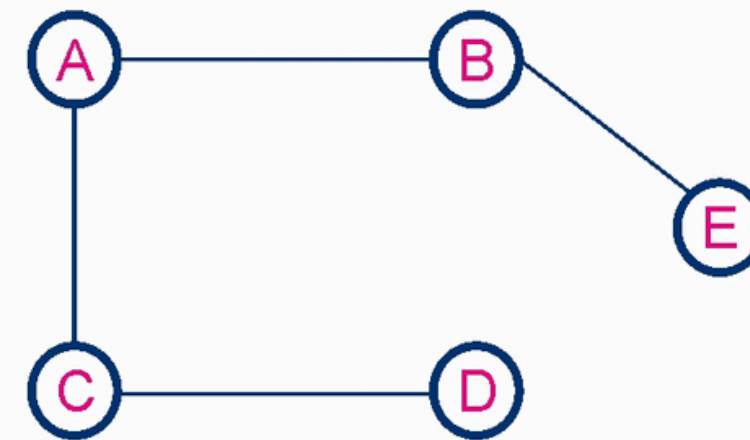
Graph Terminology



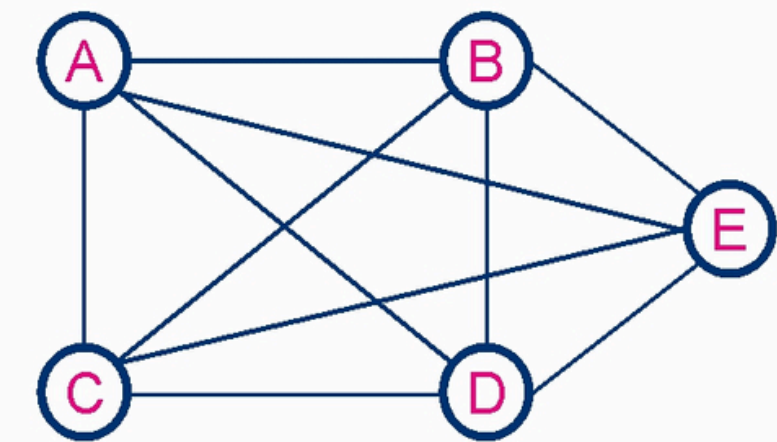
Undirected Graph



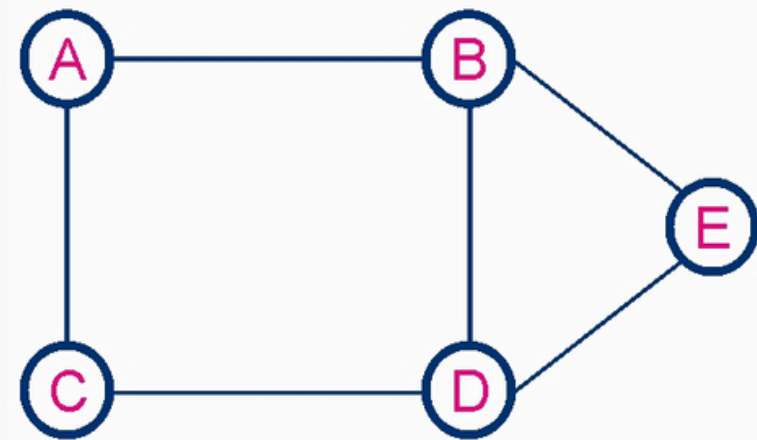
Directed Graph



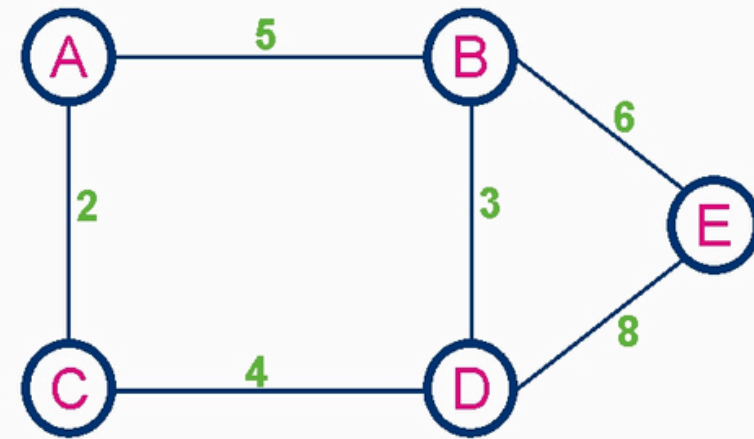
Sparse Graph



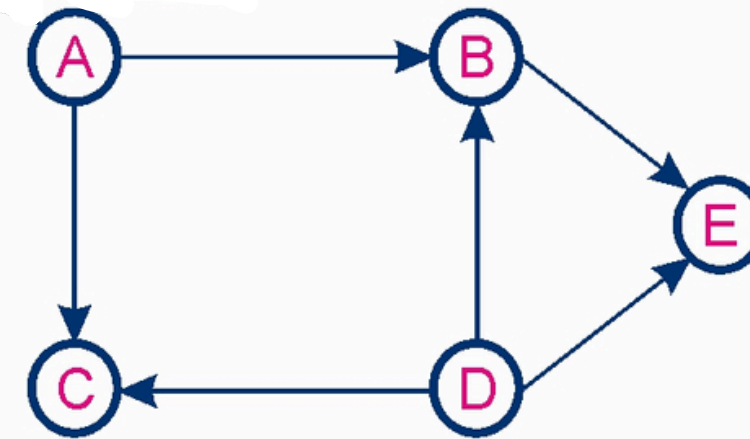
Complete Graph (Dense)



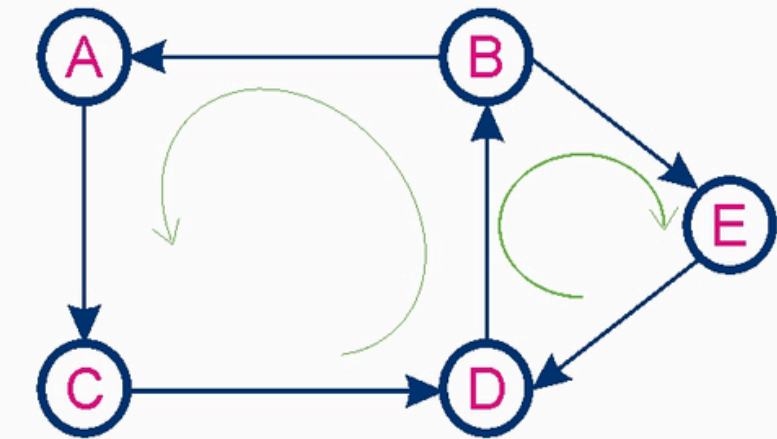
Unweighted Graph



Weighted Graph



Acyclic Graph



Cyclic Graph


Graph Traversal Algorithms

→ Breadth First Search (BFS)


- Visits nodes layer-by-layer starting from the src
- Used to find shortest path
- Time complexity: $O(V + E)$

→ Depth First Search (DFS)

- Visits nodes till as far as possible until all neighbours of a node are exhausted
- Concise recursive implementation
- Time complexity: $O(V + E)$



```
1 void bfs(Graph g, Vertex src) {
2     Queue q = QueueNew();
3     int *vis = calloc(GraphNumVertices(g), sizeof(bool));
4
5     QueueEnqueue(q, src);
6     vis[src] = true;
7
8     while (QueueSize(q) > 0) {
9         Vertex v = QueueDequeue(q);
10
11         struct adjNode *curr = g->edges[v];
12         while (curr != NULL) {
13             Vertex w = curr->v;
14             if (!vis[w]) {
15                 QueueEnqueue(q, w);
16                 vis[w] = true;
17             }
18             curr = curr->next;
19         }
20     }
21
22     free(vis);
23     QueueFree(q);
24 }
```



```
1 void dfs(Graph g, Vertex src) {
2     Stack s = StackNew();
3     int *vis = calloc(GraphNumVertices(g), sizeof(bool));
4
5     StackPush(s, src);
6     vis[src] = true;
7
8     while (StackSize(s) > 0) {
9         Vertex v = StackPop(s);
10
11         struct adjNode *curr = g->edges[v];
12         while (curr != NULL) {
13             Vertex w = curr->v;
14             if (!vis[w]) {
15                 StackPush(s, w);
16                 vis[w] = true;
17             }
18             curr = curr->next;
19         }
20     }
21
22     free(vis);
23     StackFree(s);
24 }
```


What are Trees?

A tree is a type of graph

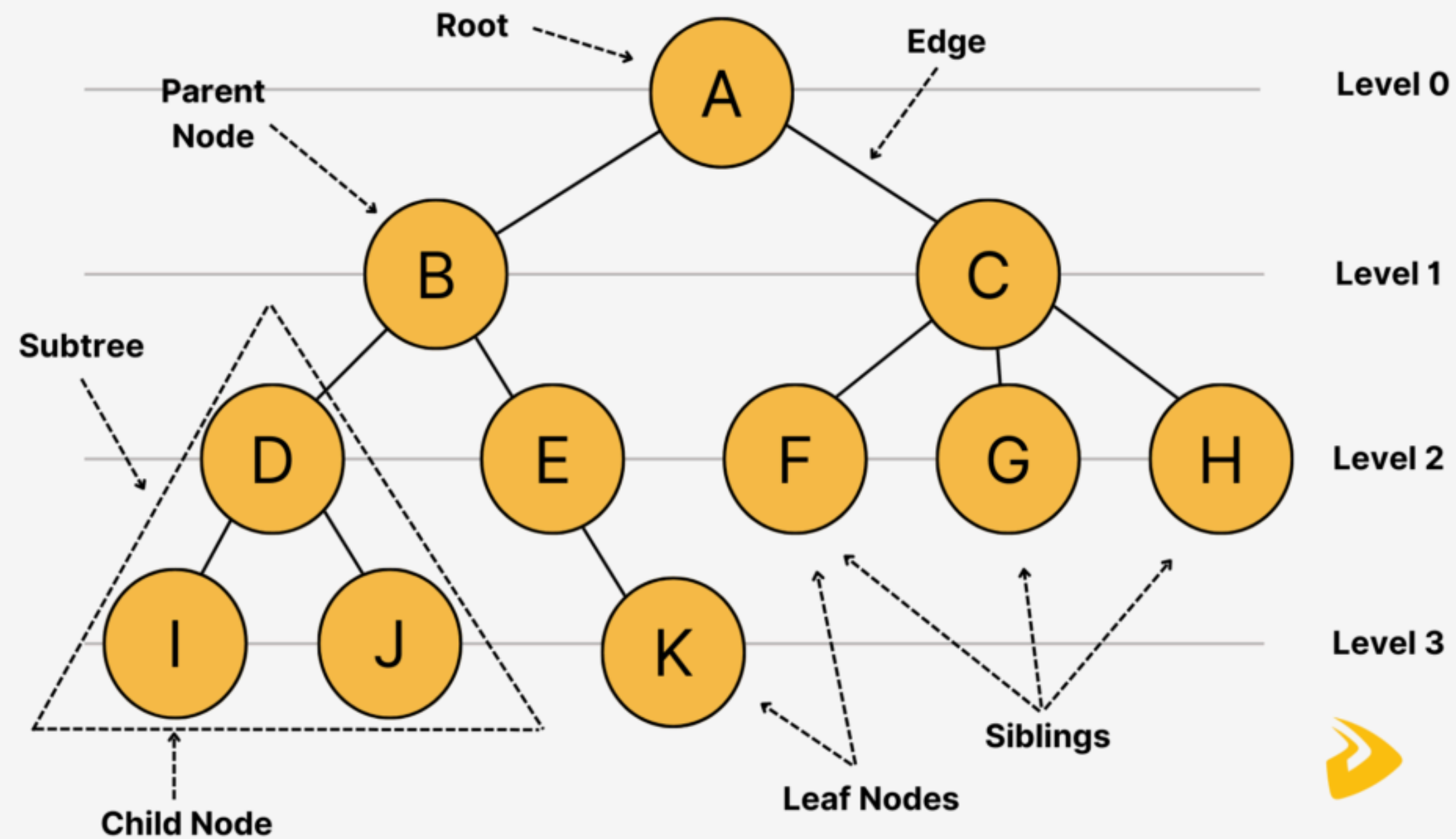
HOW NORMAL PEOPLE SEE TREES



HOW COMPUTER SCIENTISTS SEE TREES



Tree Terminology



Properties of Trees

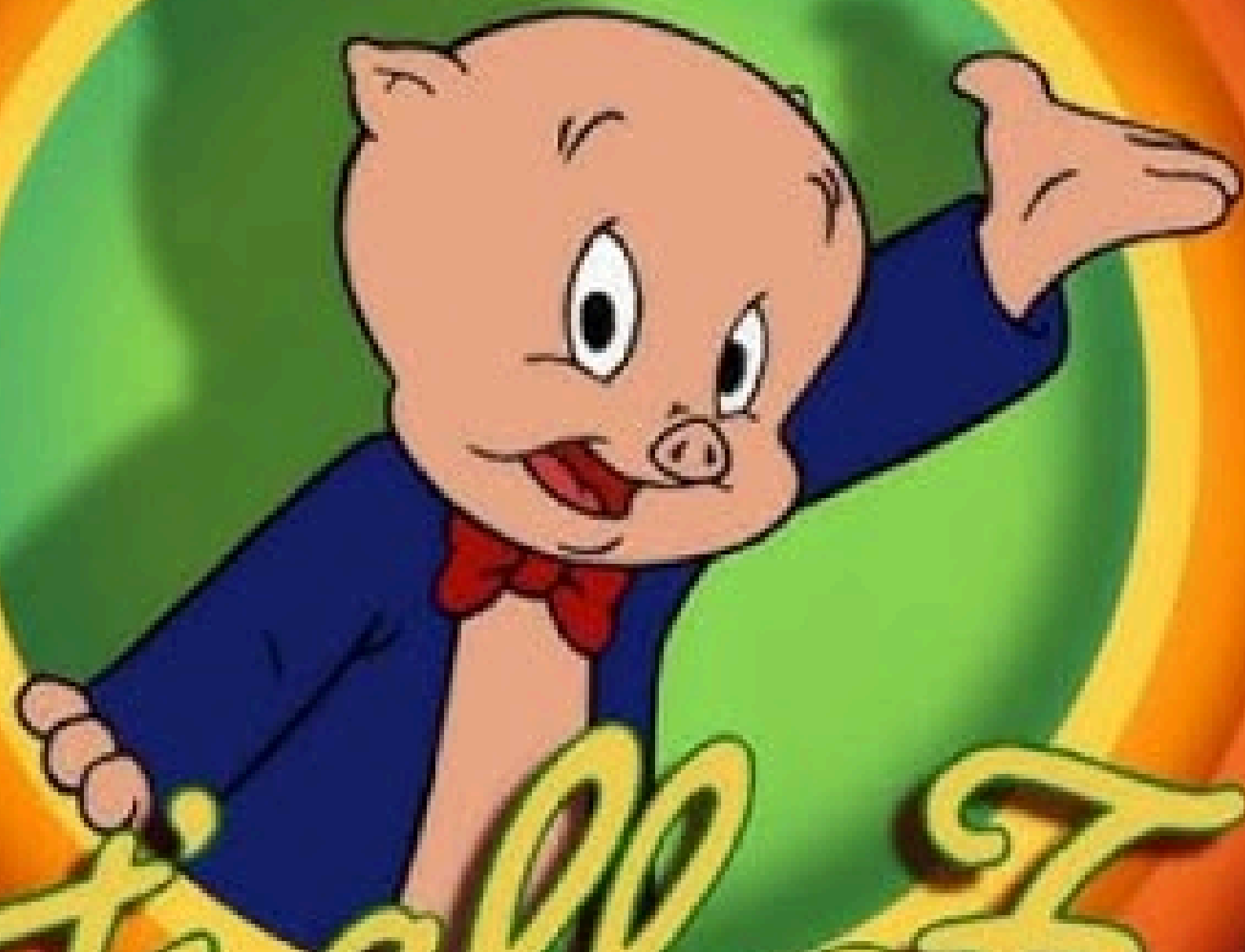
- Have n nodes and $n - 1$ edges
- Can be hierarchical like a family tree
- Connected and acyclic
- Bipartite
- Exactly one path between any pair of nodes
- Adding any edge creates a cycle
- Similarly, removing any edge disconnects the tree

Problem: Num of Islands

<https://leetcode.com/problems/number-of-islands/description/>

```
1  int numIslands(vector<vector<char>>& grid) {
2      int n = grid.size(), m = grid[0].size(), islands = 0;
3      for (int i = 0; i < n; i++) {
4          for (int j = 0; j < m; j++) {
5              if (grid[i][j] == '1') {
6                  islands++;
7                  dfs(grid, i, j);
8              }
9          }
10     }
11     return islands;
12 }
13
14 void dfs(vector<vector<char>>& grid, int i, int j) {
15     int n = grid.size(), m = grid[0].size();
16     if (i < 0 || i ≥ n || j < 0 || j ≥ m || grid[i][j] == '0') return;
17     grid[i][j] = '0';
18     dfs(grid, i - 1, j);
19     dfs(grid, i + 1, j);
20     dfs(grid, i, j - 1);
21     dfs(grid, i, j + 1);
22 }
```

LOONEY TUNES



"That's all Folks!"