4. Traversal Graph

Angel Napa

September 26, 2022

Graph

Review Definition

Graphs consists of vertices connected by some edges.

- Vertices
  - Cities
  - People
  - Computers
  - Objects

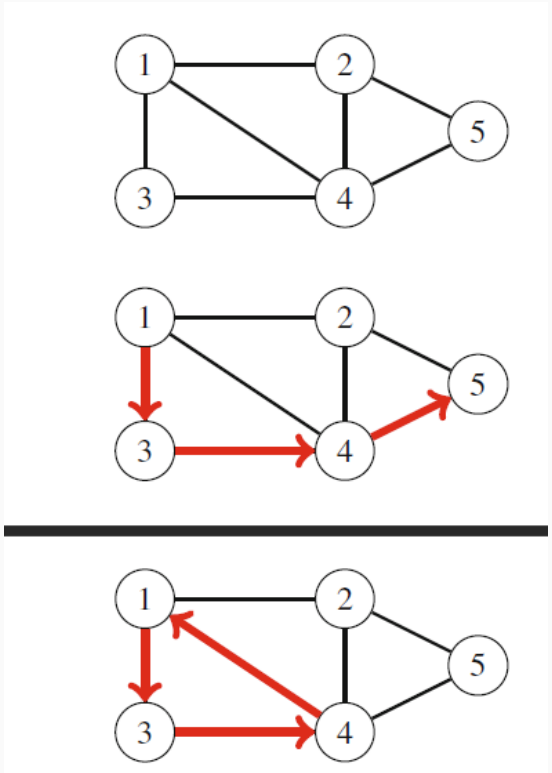
Review Definition

Graphs consists of vertices connected by some edges.

- Vertices
  - Cities
  - People
  - Computers
  - Objects
- Edges
  - Roads
  - Friendship
  - Ethernet cables
  - Relation between objects

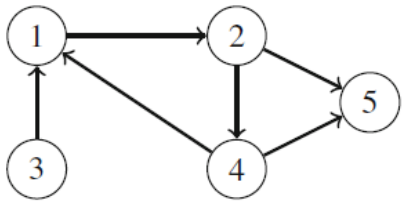
Definitions in simple Graph

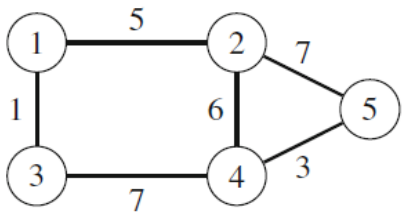
- Path
- Cycle
- Tree
- Neighbor of a vertex
- degree of a vertex

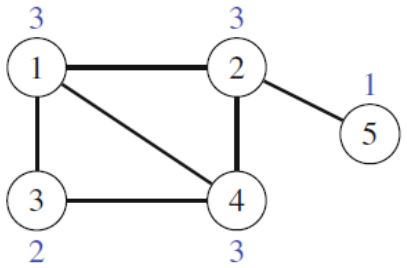


Type of Graph

- Unweighted or Weighted
- Undirected or directed
- Connected or not Connected

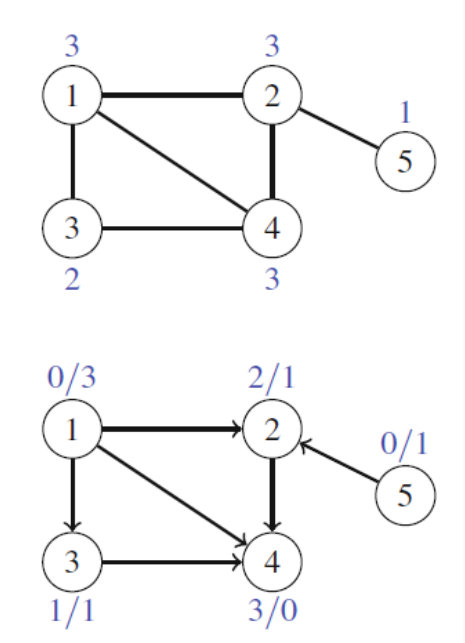






Extra definitions

- Weight of an edge
- Indegree of a vertex
- Outdegree of a vertex

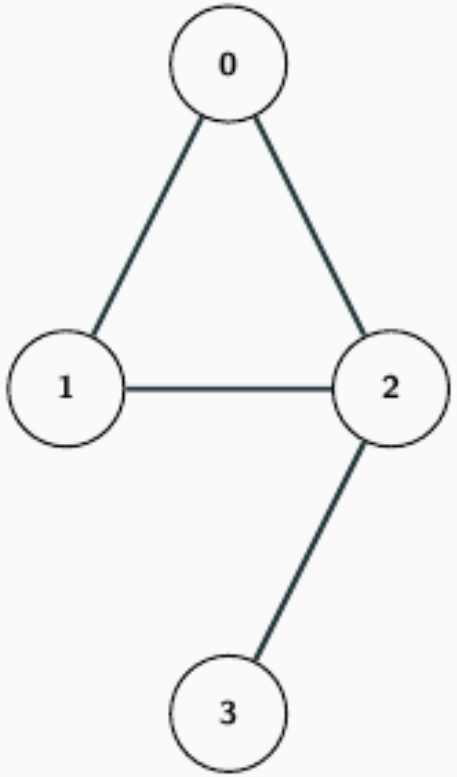


Graph Representation

Adjacency list

```
0: 1, 2
1: 0, 2
2: 0, 1, 3
3: 2
```

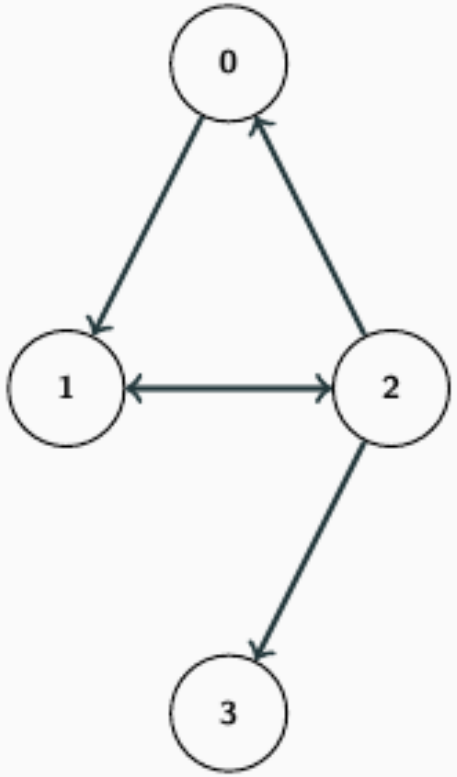
```
vector<int> adj[4];
adj[0].push_back(1);
adj[0].push_back(2);
adj[1].push_back(0);
adj[1].push_back(2);
adj[2].push_back(0);
adj[2].push_back(1);
adj[2].push_back(3);
adj[3].push_back(2);
```



Adjacency list

```
0: 1
1: 2
2: 0, 1, 3
3:
```

```
vector<int> adj[4];
adj[0].push_back(1);
adj[1].push_back(2);
adj[2].push_back(0);
adj[2].push_back(1);
adj[2].push_back(3);
```



Adjacency list weighted graph

```
vector<pair<int,int>> adj[N];
```

```
adj[1].push_back({2,5});
adj[2].push_back({3,7});
adj[2].push_back({4,6});
adj[3].push_back({4,5});
adj[4].push_back({1,2});
```

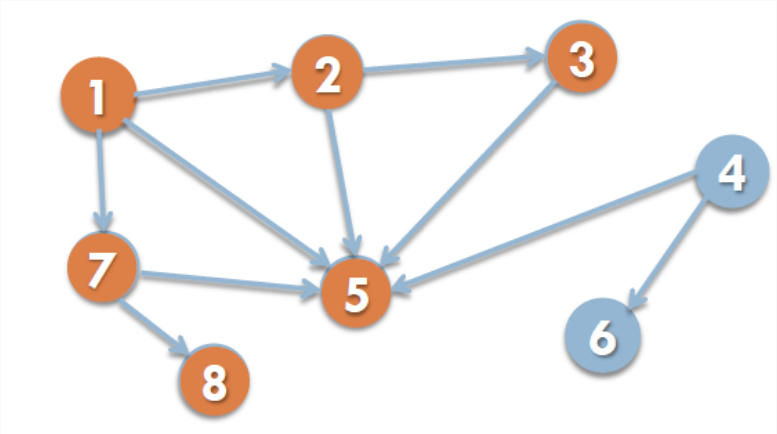
Graph Traversal

Definition

- Goal: visit each vertex reachable from some starting vertex
- Doing efficiently.

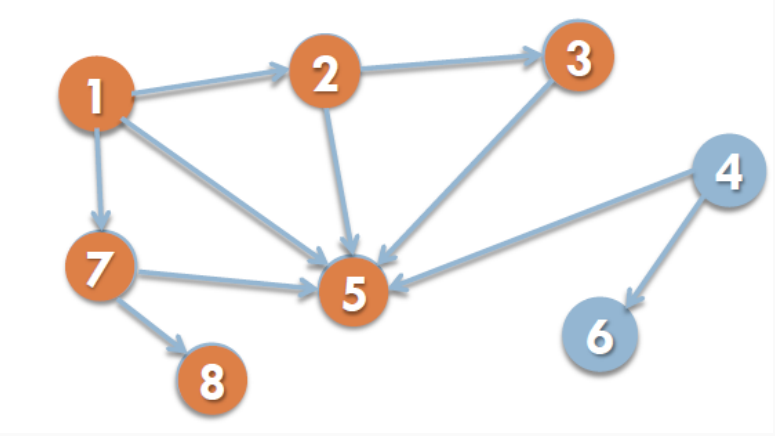
Depth-First Search

Idea: Recursively visit each unvisited neighbor.



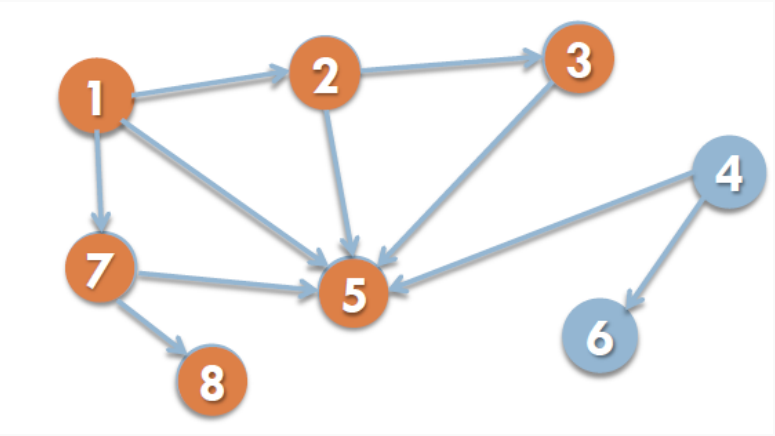
Depth-First Search, Recursive implementation

```
void dfs(Vertex v) {
    mark v visited;
    for all edges (v,u):
        if u is unmarked:
            dfs(u);
}
```



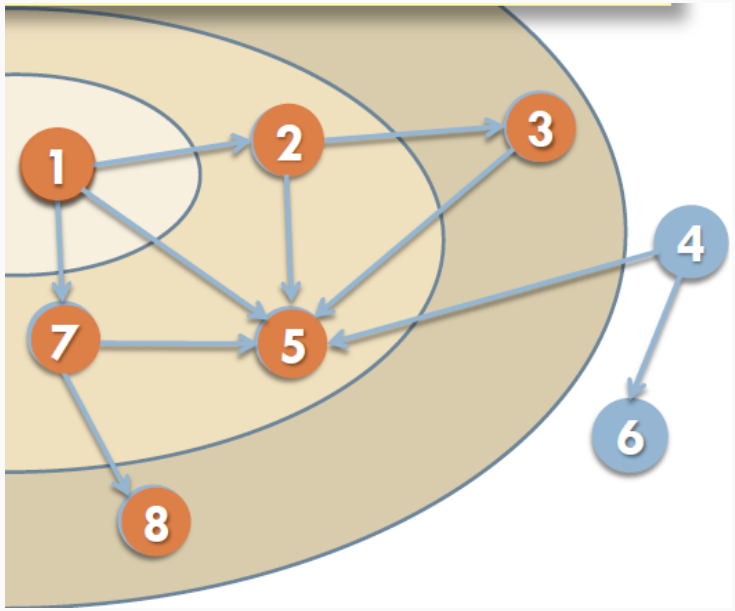
Depth-First Search,non recursive implementation

```
void dfs(Vertex u) {
    Stack s= new Stack();
    s.push(u);
    while (s is not empty) {
        u= s.pop();
        if (u not visited) {
            visit u;
            for each edge (u, v):
                s.push(v);
        }
    }
}
```



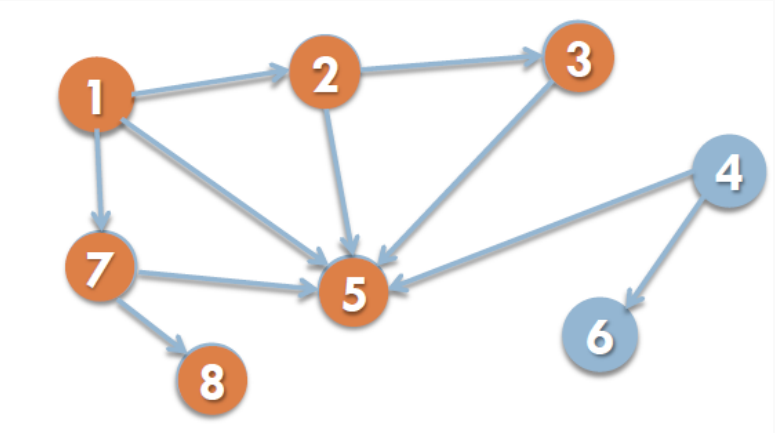
Breadth-First Search

Idea: Iterative process the graph in "layers" moving further away from the vertex.



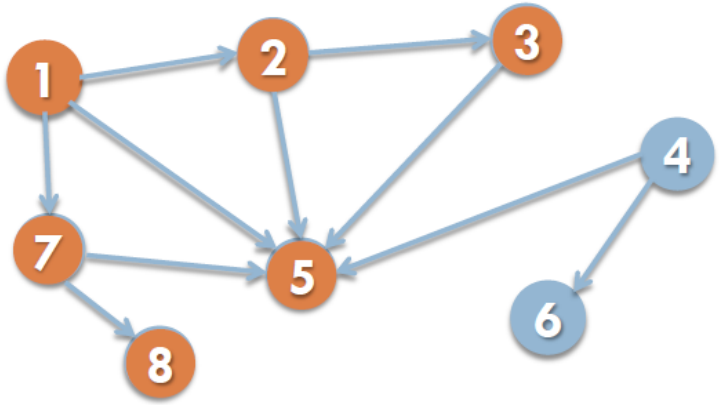
Breadth-First Search

```
void bfs(int u) {
    Queue q= new Queue()
    q.add(u);
    while ( q is not empty ) {
        u= q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                q.add(v);
            }
        }
}
```



Breadth-First Search

```
void bfs(int u) {
    Queue q= new Queue
    q.add(u);
    while ( q is not empty ) {
        u= q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                if (v not encountered) {
                    mark v as encountered;
                    q.add(v);
} } } }
```



Exercises

- Kattis: coast, buttonbashing
- SPOJ: NAKANJ
- CSES: Message route, labyrinth
- Codeforces: 277A, 60B, 769C

Gracias!