Discussion 8

ECS 122A
May 22, 2019

1. Graph Basics

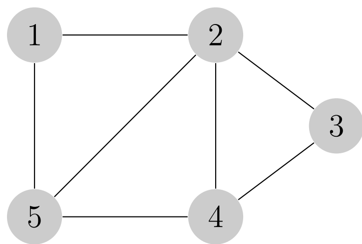


Figure 1: An undirected graph G_1 with 5 vertices and 7 edges

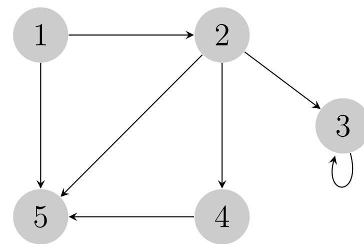
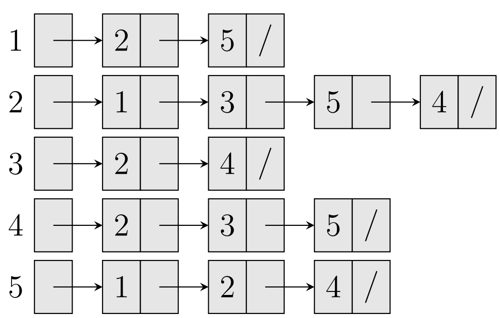


Figure 2: A directed graph G_2 with 5 vertices and 7 edges.

a. Representation: Adjacency List, Adjacency Matrix

What is the adjacency matrix and adjacency list of G_1 and G_2, respectively?



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

Figure 3: Adjacency list and adjacency matrix representation of G_1

How about G_2? It's your turn now!

1. Adjacency matrix needs $\Theta(|V| \times |E|)$ storage space while adjacency list requires $\Theta(|V| + |E|)$.

2. If G is undirected, its adjacency matrix A is symmetric. Namely, $A^T = A$. Further, the main diagonal entries of A are all zeros.

3. **Self-loops**—edges from a vertex to itself—are possible in a directed graph, but are forbidden in an undirected graph.

b. Degree

1. $\sum_{u \in V} \text{degree}(u) = 2|E|$, where G is an undirected graph.

2. $\sum_{u \in V} \text{out-degree}(u) = \sum_{u \in V} \text{in-degree}(u) = |E|$, where G is a directed graph.

3. $\text{degree}(u) = \text{out-degree}(u) + \text{in-degree}(u)$, where $u \in V$ and G is a directed graph.

4. A vertex whose degree is 0 is **isolated**.

c. Path, Connected Component

1. A **path** of length k from a vertex u to a vertex u' in a graph $G =< V, E >$ is a sequence $< v_0, v_1, v_2, ..., v_k >$ of vertices such that $u = v_0$, $u' = v_k$.

2. An undirected graph is **connected** if every vertex is reachable from all other vertices.

3. A directed graph is **strongly connected** if every two vertices are reachable from each other.

2. BFS and DFS

```
BFS(G, s)
// G: input graph; s: source vertex
for each vertex u ∈ V − {s}
      d[u] = +INFTY
d[s] = 0

// create FIFO queue
Q = EMPTY
ENQUEUE(G, s)
while Q not EMPTY
      u = DEQUEUE(G)
      for each v ∈ Adj[u]
            if d[v] = +INFTY
                  d[v] = d[u] + 1
                  ENQUEUE(G, v)
return d
```

Discussion 8

```
DFS(G)
for each vertex u ∈ V
    u.color = WHITE
time = 0
for each vertex u ∈ V
    if u.color = WHITE
        DFS-VISIT(u)
```

```
DFS-VISIT(u)
// white vertex u has just been discovered
time = time + 1
u.discover = time
u.color = GRAY

// explore edge (u, v)
for each vertex u ∈ Adj[u]
    if u.color = WHITE
        DFS-VISIT(u)

// blacken u, it is finished
u.color = BLACK
time = time + 1
u.finish = time
```