Prepared by: Ji Wang | ECS 122A | May 22, 2019

Discussion 8

1. Graph Basics

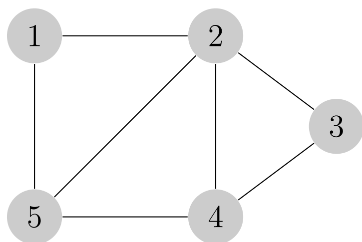


Figure 1: An undirected graph G_1 with 5 vertices and 7 edges

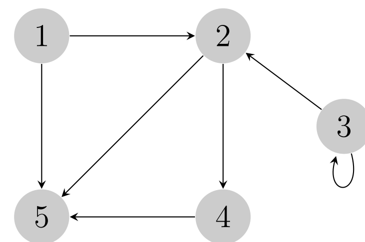
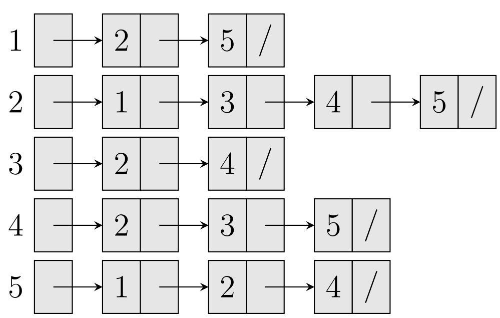


Figure 2: A directed graph G_2 with 5 vertices and 7 edges.

a. Representation: Adjacency List, Adjacency Matrix

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



	1	2	3	4	5
1	0	1	0	0	1
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 is of size $|V| \times |V|$ while adjacency list needs $\Theta(|V| + |E|)$ space.

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.

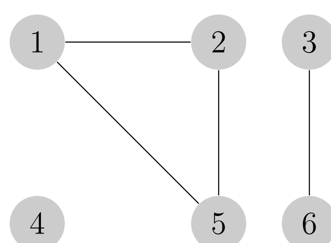


Figure 4: An undirected graph G_3 with 3 connected components:
$\{1, 2, 5\}$, $\{3, 6\}$ and $\{4\}$

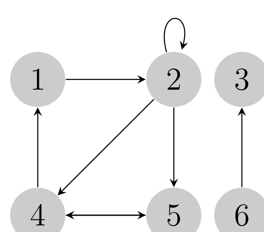


Figure 5: A directed graph G_4 with 3 strongly connected components:
$\{1, 2, 4, 5\}$, $\{3\}$ and $\{6\}$

2. BFS and DFS

BFS(G, s)

```
// G: input graph (sorted in alphabetical/ascending order);
// 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
```

Let's run BFS on graph G_1!

Three-color is used to indicate search status of vertices

- WHITE = a vertex is undiscovered
- GRAY = a vertex is discovered, but its processing is incomplete
- BLACK = a vertex is discovered, and its processing is complete

Classification of edges (When we explore the edge, line 7-9 in DFS-VISIT(u)):

- T = Tree edge = encounter new vertex (GRAY to WHITE)
- B = Back edge = from descendant to ancestor (GRAY to GRAY)
- F = Forward edge = from ancestor to descendant (GRAY to BLACK)
- C = Cross edge = any other edges (between trees and subtrees) (GRAY to BLACK)

```
DFS(G)
// G: input graph (sorted in alphabetical/ascending order);
for each vertex u ∈ V
      u.color = WHITE
time = 0
for each vertex u ∈ V
      if u.color = WHITE
            // recursive routine/function
            DFS-VISIT(u)
```

```
DFS-VISIT(u)
 1  // white vertex u has just been discovered
 2  time = time + 1
 3  u.discover = time
 4  u.color = GRAY
 5
 6  // explore edge (u, v)
 7  for each vertex v ∈ Adj[u]
 8        if v.color = WHITE
 9              DFS-VISIT(v)
10
11  // blacken u, it is finished
12  u.color = BLACK
13  time = time + 1
14  u.finish = time
```

Try DFS on graph G_2!