Tutorial 12: Graphs

April 22nd, 2025

Introduction to Graphs

A **graph** $G = (V, E)$ consists of a set V of vertices (nodes) and a set E of edges connecting pairs of vertices. Graphs can be directed or undirected:

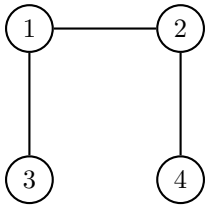
- **Directed Graph**: Edges have direction.
- **Undirected Graph**: Edges have no direction.

Graph Representation

Adjacency Matrix

A square matrix A, where $A[i, j] = 1$ if an edge exists from vertex i to vertex j; otherwise, 0.

Example for an undirected graph:



Adjacency Matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Adjacency List

An array of lists, where each index represents a vertex and stores a list of its adjacent vertices.

Adjacency List for the same graph:

$$1 \to [2, 3]$$
$$2 \to [1, 4]$$
$$3 \to [1]$$
$$4 \to [2]$$

Breadth-First Search (BFS)

BFS explores vertices level by level, using a queue structure.

Algorithm 1 BFS(G, s)

```
1: for each vertex u ∈ V − {s} do
2:     color[u] = WHITE
3:     distance[u] = ∞, parent[u] = NIL
4: end for
5: color[s] = GRAY, distance[s] = 0, parent[s] = NIL
6: Q ← empty queue; enqueue(Q, s)
7: while Q is not empty do
8:     u ← dequeue(Q)
9:     for each vertex v adjacent to u do
10:        if color[v] = WHITE then
11:            color[v] = GRAY, distance[v] = distance[u] + 1
12:            parent[v] = u, enqueue(Q, v)
13:        end if
14:    end for
15:    color[u] = BLACK
16: end while
```

Example: BFS from vertex 1

Initial graph (above example):
Queue: $Q = [1]$; Visited: 1 (distance=0)

Dequeue 1: enqueue neighbors 2,3
Queue: $Q = [2, 3]$; Visited: 1,2,3

Dequeue 2: enqueue neighbor 4
Queue: $Q = [3, 4]$; Visited: 1,2,3,4

Dequeue 3: no unvisited neighbors
Queue: $Q = [4]$

Dequeue 4: no unvisited neighbors
Queue empty, BFS complete.

BFS traversal: $1 \to 2 \to 3 \to 4$.

Depth-First Search (DFS)

DFS explores as far as possible along each branch before backtracking, using recursion or a stack.

Algorithm 2 DFS(G)

```
1: for each vertex u ∈ V do
2:     color[u] = WHITE, parent[u] = NIL
3: end for
4: time = 0
5: for each vertex u ∈ V do
6:     if color[u] = WHITE then
7:         DFS-Visit(u)
8:     end if
9: end for
```

Algorithm 3 DFS-Visit(u)

1: color[u] = GRAY, time = time + 1, discovery[u] = time
2: **for** each vertex v adjacent to u **do**
3: **if** color[v] = WHITE **then**
4: parent[v] = u, DFS-Visit(v)
5: **end if**
6: **end for**
7: color[u] = BLACK, time = time + 1, finish[u] = time

Example: DFS starting from vertex 1

- Visit 1 (time=1), explore neighbor 2
- Visit 2 (time=2), explore neighbor 4
- Visit 4 (time=3), no unvisited neighbors, finish 4 (time=4)
- Finish 2 (time=5), back to 1, explore neighbor 3
- Visit 3 (time=6), finish 3 (time=7)
- Finish 1 (time=8)

DFS traversal: $1 \to 2 \to 4 \to 3$.
Discovery/Finish times:

$$1 :(1,8) \quad 2 : (2,5) \quad 4 : (3,4) \quad 3 : (6,7)$$