DSA Notes

Md Rabiul Hosen Apu

May 27, 2025

1 Graph

1.1 Depth-First Search (DFS) Algorithm

```
void dfs(int node) {
    visited[node] = true;
    cout << node << " ";

    for (int child : adj[node]) {
        if (!visited[child]) {
            dfs(child);
        }
    }
}
```

1.2 Breadth-First Search (BFS) Algorithm

```
void bfs(int start) {
    queue<int> q;
    q.push(start);
    visited[start] = true;

    while (!q.empty()) {
        int node = q.front();
        q.pop();
        cout << node << " ";

        for (int child : adj[node]) {
            if (!visited[child]) {
                q.push(child);
                visited[child] = true;
            }
        }
    }
}
```

1.3 Topological Sorting

1.3.1 DFS Based

```
void dfs(int node) {
    visited[node] = true;
    for (int child : adj[node]) {
        if (!visited[child]) {
            dfs(child);
        }
    }
    st.push(node);
}
```

1.3.2 BFS Based (Kahn's Algorithm)

```
void kahn_topo_sort(int n) {
    queue<int> q;
    vector<int> indegree(n + 1, 0);

    for (int i = 1; i <= n; ++i) {
        for (int child : adj[i]) {
            indegree[child]++;
        }
    }

    for (int i = 1; i <= n; ++i) {
        if (indegree[i] == 0) q.push(i);
    }

    while (!q.empty()) {
        int node = q.front();
        q.pop();
        cout << node << " ";

        for (int child : adj[node]) {
            indegree[child]--;
            if (indegree[child] == 0) q.push(child);
        }
    }
}
```

1.4 Dijkstra's Algorithm (Shortest Path)

```
void dijkstra(int source, int n) {
    vector<int> dist(n + 1, 1e9);
    set<pair<int, int>> st;
    dist[source] = 0;
    st.insert({0, source});

    while (!st.empty()) {
        auto it = *st.begin();
        st.erase(it);

        int d = it.first;
        int node = it.second;

        for (auto child : adj[node]) {
            int childNode = child.first;
            int weight = child.second;

            if (d + weight < dist[childNode]) {
                st.erase({dist[childNode], childNode});
                dist[childNode] = d + weight;
                st.insert({dist[childNode], childNode});
            }
        }
    }

    for (int i = 1; i <= n; ++i) {
        cout << "Distance to " << i << " is " << dist[i] << endl;
    }
}
```

2 Data Structure

3 Dynamic Programming