Graphs Meet - Programming Club

Notes by Arihant Tripathy

15 April, 2024

Contents

1 Why Study Graphs

They aren't a black box, they are helpful in solving a variety of seemingly random problems. The points is to develop a mindset that allows you to see the problem as a graph problem, and not get caught in traps of thinking that the problem is a graph problem when it isn't.

2 Graphs

A graph is a collection of nodes and edges. The nodes are sometimes called vertices. The edges are the lines that connect the nodes.

3 Directed vs Undirected Graphs

In a directed graph, the edges have a direction. In an undirected graph, the edges do not have a direction. In other words, undirected graphs are basically bidirected graphs. The adjacency matrix of an undirected graph is symmetric.

In Degree - The number of edges that end on a node.

Out Degree - The number of edges that start from a node.

4 Sparse vs Dense Graphs

A graph is sparse if the number of edges is much less than the number of nodes. A graph is dense if the number of edges is close to the number of nodes. A graph with n nodes can have max ${}^nC_2$ edges. Most of the times you'll get an undirected and unweighted graph. Weighted graphs are used in optimisation problems:

- Shortest distance between nodes
- Least cost of a subgraph
- Minimize the cost over paths
- Maximize the gain over paths

5 Graph Representation

5.1 Pointer Representation

In this representation, we have a bunch of nodes (structs) and each node has pointers to the nodes it is connected to. This method is inefficient because it leads to cache misses.

5.2 Adjacency Matrix

The adjacency matrix is a 2D array of size $V \times V$ where V is the number of vertices in the graph. In an unweighted graph, the adjacency matrix is a boolean matrix where $a[i][j]$ is true if there is an edge from i to j, otherwise false. In a weighted graph, the numbers in the matrix are the weights of the edges.

5.2.1 Checking path of length k using adjacency matrix

To check if there is a path of length k between two nodes, we can raise the adjacency matrix to the power of k. Each element in the resulting matrix will be the number of paths of length k between the two nodes.

5.3 Adjacency List

For every node, we make a list of its neighbours. For a weighted graph, each element in the list is a pair, where the first element is the node and the second element is the weight of the edge.

6 Directed Acyclic Graphs (DAGs) / Trees

A tree is just a graph with the minimum number of edges required to connect all the nodes. It's $n-1$ edges for n nodes. When you have directed edges in a graph with no cycles, it is a **Directed Acyclic Graph** (DAG). There exists only one path between two nodes in a tree.

6.1 Rooting a Tree / DAG

Imagine "picking up" a node and making it the root of a tree. Then, for every node, you can find the path from the root to that node. This is called rooting a tree.

Practice Problem 1: USACO Silver Grass Planting - Solution

7 Graph Traversals

If you need to copy paste code for a graph traversal, you're doing it wrong.

7.1 Depth First Search (DFS)

DFS is a recursive algorithm that starts at the root node and explores as far as possible along each branch before backtracking.

Practice Problem 2: USACO Bronze Livestock Lineup - Solution
Practice Problem 3: USACO Bronze Milk Factory - Solution

7.1.1 Floodfill and implicit graphs

Floodfill is a technique used to traverse implicit graphs. An grid can often be considered as an implicit graph, where each cell is a node and the edges are the connections between the cells. Similar to the infamous Gotham Knightmare problem in CPro 2023.

```
int dx[] = {0, 1, 0, -1};
int dy[] = {1, 0, -1, 0};

void floodfill(int x, int y, ...) {
    if (x < 0 || x >= n || y < 0 || y >= m || grid[nx][ny]) return;
    if (visited[x][y]) return;

    visited[x][y] = true;

    for (int i = 0; i < 4; i++)
        floodfill(x + dx[i], y + dy[i], grid, visited, ...);
}
```

Practice Problem 4 CSES Counting Rooms - Solution

7.2 Breadth First Search (BFS)

BFS is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

7.2.1 The Concept of Equivalent Vertices

It is common in BFS to consider two vertices as "equivalent". In other words, being on any one of the equivalent vertices is the same as being on the other. One common trick is to add edges of weight 0 between such vertices. What would happen if you put a set of equivalent vertices in a BFS queue?
You would be able to visit all of them in the same step. The same concept is used in multi-source Dijkstra. Dijkstra is essentially a generalised BFS.

8 Not So Lite Graph Problems

Practice Problem 5: USACO Silver Milk Pails - Solution
Practice Problem 6: Codeforces Row Major
Practice Problem 7: CodeForces Rudolf and the Ball Game
Practice Problem 8: CodeForces Shifting String