Graphs Handout - IICPC

Aditya Mishra

June 29, 2024

Contents

Chapter 1

Graph Traversals

A lot of graph problems boil down to simply traversing the graph in a certain manner, and using the information obtained at each step in the traversal process.

1.1 Breadth First Search

Think of it as traversing layer-by-layer, moving 1 layer away from the source at each step. It is implemented using a queue, where each element on the periphery of each element in the given layer is pushed to the queue, thereby being accessed after the entire current layer has already been accessed.
Here are some problems that can be solved by traversing over the graph layer-by-layer:

- CSES-Building Roads
- Number of connections
- CSES-SSSP
- CSES-Building Teams
- CSES-Monsters

Visualization: BFS

1.2 Depth First Search

It is a recursive process, where we first travel as far away from the node as we can in one branch, and then we backtrack to the last accessible branch.
Some problems involving DFS:

- Monsters
- Islands

- Path Sum
- Maximum area of island
- House Robber
- Contain Virus

Visualization: DFS

Comparison BFS vs DFS: BFS vs DFS

Chapter 2

Shortest Path Algorithms

2.1 Djikstra's Algorithm

Initialise distance to the source node as 0, and distance to every other node as infinity. Start with the source node and mark distance to all its neighbours as the distance from the source to that node. Then visit the closest node update the distance to all its neighbours, then visit the closest unvisited node. Keep repeating this process till you have visited all the nodes. The way we store the unvisited nodes is using a priority queue, for efficient retrieval of closest nodes.
Time Complexity: [E+V]log(V)

- CSES-Shortest Routes-1
- Path-with-maximum-probability
- Network delay time
- Reachable nodes in subdivided graph
- CSES-Flight Discount

One problem with Djikstra's algorithm is that it does not work for graphs with negative edges (**Think why!**).

2.2 Bellman-Ford algorithm

Though much slower than Djikstra's algorithm, Bellman-Ford can handle negative weight edges! Initialise distance to source as 0 and to all nodes as infinity, then traverse through the list of edges. During each traversal, if you encounter an edge that leads to the path to a certain node getting updated and decreased, then relax that node's distance. Here is a good video explanation: Bellman-Ford
Some problems involving Bellman-Ford:

- CSES-High Score

- CSES-Negative cycle detection

Termination of the edge-iterations: Terminate when after a full traversal through edges, no edge is relaxed. If there is no termination after (**Num_Nodes**) iterations, it means there is a negative edge cycle! Since the maximum number of traversal is N (**Num_Nodes**), the time complexity is
O(E*N).

2.3 Floyd-Warshall Algorithm

Used to determine the minimum distance between all pairs of nodes, this algorithm can handle negative weight cycles too! Initialise a matrix M where M[A][B] is the shortest distance between node A and node B = [**Edge_length** if there is a an edge between A and B], [0 if A==B], [INF if there is no edge between A and B]. Run a traversal over the set of nodes. For each node that the traversal is currently hanging over, update every entry of the matrix that does not involve the current node, as
M[A][B]=min(M[A][B],M[A][C]+M[C][B]).
Once you would have traversed over all pairs of nodes, you will have the minimum distance for all pairs of nodes. Here are some problems involving Floyd-Warshall Algorithm:

- CSES-Shortest Routes II
- Smallest number of neighbours at threshold distance

Chapter 3

Topological Sort

In essence this method is the chopping of leaves sequentially, till you have no leaf left. This is the soul of numerous interesting problem types, including diameter finding and course scheduling. How it is done is basically you store all the leaf nodes in a data structure (Queue or stack or anything) and then decrease the number of neighbours of their neighbour. If the neighbour now has number of neighbours==1, then it is a leaf and it is pushed in the data structure storing leaves! This way you sequentially chop leaves and head towards to the center!

Implement it:

- CSES- Subordinates
- CSES- Tree Diameter
- CSES- Course Schedule
- CSES- Game Routes
- CSES- Longest Flight Route
- CSES- Course Schedule II

Chapter 4

Disjoint Set Union

The broad problem statement is: you have 2 sets of connected nodes A and B, and you add an edge between an element of A and an element of B, thus creating a single connected component C. This process is called a disjoint set union. DSU involves mainly 2 kinds of operations:

- Find: Finding which set a given node belongs to.
- Union: Uniting the sets of 2 different nodes.

For carrying out DSU, we have a special kind of DSU data structure that consists of a forest of trees. Each tree has a representative element that has no parent, and each element in the tree except for the representative element has a parent.

Find: Keep going to the parent as long as you don't reach the representative element of the tree. Once you reach the representative element, its parent attribute will be the size of the tree instead of a node's address.

Union: To unite the trees of 2 nodes, first find their parents, and then make the one with the larger tree size, as the parent of the other one.

Time Complexity: ack(V) where ack is the Inverse Ackermann Function
Here is a good video explanation: DSU
Practice!:

- CSES-Road Construction
- Connect Manhattan
- Regions cuts and slashes

Chapter 5

Spanning Trees

Spanning tree is a connected subset of a connected graph having no cycles. **Minimum spanning tree** is the spanning tree with minimum sum of edge weights.

5.1 Kruskal's Algorithm

GREED!
Sort edges. Then keep adding the minimum edge as long as you don't get a cycle. Skip an edge if it leads to a cycle. Do this process N-1 times, as N - 1 [**Num_Nodes** - 1] is the number of edges in a spanning tree.

Time Complexity: O(ElogE+E*ack(V)) where ack(V) is the inverse Ackerman function.

Cycle detection:
Maintain a DSU data stricture throughout the process, and for each node, see if it is being added within a connected component or not. If it is, it will lead to a cycle and hence it is to be skipped.

5.2 Prim's Algorithm

GREED ONCE AGAIN!
Start with an arbitrary vertex as the sole node of the tree initially, and then keep adding the closest vertices among the ones directly connected to some node of the tree and not already there in the tree. Keep doing this as long as you haven't added all the nodes to the tree.
Using priority-queue, the **time complexity** is **O(ElogV)**

Practice MST!

- Repairing Roads
- Inverse the problem
- GCD and MST
- Find critical and pseudo-critical edges

Chapter 6

DP on trees

Use subtrees as subproblems!

- Red-Black
- CSES-Tree Matching
- Distance in Tree
- CSES-Tree Distances I
- CSES-Tree Distances II

Chapter 7

IICPC Resources

- Lectures
- POTD list