

# Graph Theory - I

# Today's Plan

- Introduction
- Special Graphs
- Various Representations
- Depth First Search
  - Solve a problem from GCJ
- Breadth First Search
  - Solve a problem from SPOJ
- Dijkstra's Algorithm
  - Solve a problem from SPOJ

# Special Graphs

- Undirected Graphs
- Edge Weighted Graphs
- Directed Graphs
- Trees
- Directed Acyclic Graphs
- Bi-Partite Graphs

# Representation - I

- Adjacency matrix
  - 2 D Array  $\mathbf{M}$  of size  $|V| \times |V|$
  - $\mathbf{M}[i][j] = 1$  if  $V_i$  and  $V_j$  are connected by an edge and 0 otherwise.
- Adjacency List
  - Each vertex maintains a list of vertices that are adjacent to it.
  - We can use: `vector< vector<int> >`

# Representation - II

- Edge List
- Checking if edge  $(V_i, V_j)$  is present in  $G$ .
  - Adjacency Matrix –  $O(1)$
  - Adjacency List –  $O(\min(\deg(V_i), \deg(V_j)))$
- Iterating through the list of neighbours of  $V_i$ 
  - Adjacency Matrix –  $O(|V|)$
  - Adjacency List –  $O(\deg(V_i))$

# Representation - III

- Implicit graphs
  - Two squares on an 8x8 chessboard. Determine the shortest sequence of knight moves from one square to the other.
- Tricks:
  - Use  $Dx[]$  ,  $Dy[]$  for generating the neighbors of a position in grid problems.

# Depth First Search

- Finding Connected Components
- Implemented using
  - Stack
  - Recursion (Most Frequently used)
- Complexity
  - Time:  $O(|V| + |E|)$
  - Space:  $O(|V|)$  [to maintain the vertices visited till now]
- Google Code Jam Problem
  - <http://code.google.com/codejam/contest/dashboard?c=90101#s=p1>

# Breadth First Search

- Finding a Path with Minimum # of edges from starting vertex to any other vertex.
- Used to Solve Shortest Path problem in un weighted graphs
- Implemented using queue
- Same Time and Space Complexity as DFS.
- SPOJ Problem
  - <http://www.spoj.pl/problems/PPATH/>



# Dijkstra's Algorithm

- Used to solve Shortest Path problem in Weighted Graphs
- Only for Graphs with positive edge weights
- Greedy strategy
- Use `priority_queue<node>` for implementing Dijkstra's
- SPOJ Problem
  - <http://www.spoj.pl/problems/CHICAGO>

# Practice problems

- <http://www.spoj.pl/problems/PARADOX/>
- <http://www.spoj.pl/problems/HERDING/>
- <http://www.spoj.pl/problems/COMCB/>
- <http://www.spoj.pl/problems/PT07Y/>
- <http://www.spoj.pl/problems/PT07Z/>

# More Practice Problems

- SRM 453.5 Division 2 500 Pt
- <http://www.codechef.com/problems/N4>
- <http://www.spoj.pl/problems/ONEZERO>
- <http://www.spoj.pl/problems/CERC07K/>