Thursday, 19 July 2018

Launchpad

Lecture -22

Data Structures

Graphs

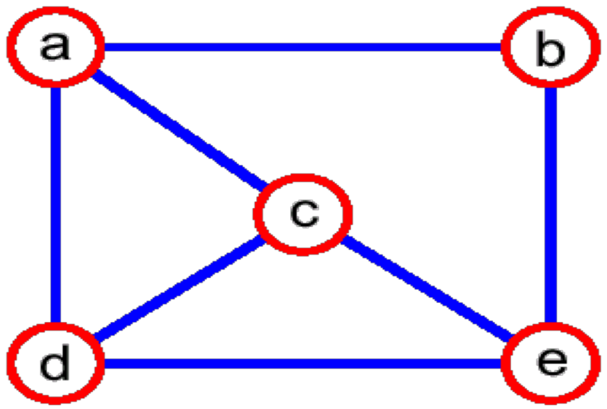
Kartik Mathur

CODING BLOCKS

Graphs

Graphs

Graph is a data structure that can be represented as a finite set of **nodes** and **edges**. **Nodes** are called as **vertices** and **edges** are used to connect two vertices.



V= {a,b,c,d,e}

E= {(a,b),(a,c),(a,d), (b,e),(c,d),(c,e), (d,e)}

Examples : Used in Google Maps, Social Networking sites, sending packets through various servers.

CODING BLOCKS

Terminology

1. Adjacent Vertices
2. Degree
3. Path
4. Connected Graph
5. Subgraph
6. Connected Components
7. Tree
8. Forest
9. Spanning Tree

Un-Weighted Graphs?

- Each edge may or may not carry some weight.

- Types of Edges :
  - Undirected Edges
  - Directed Edges

CODING BLOCKS

Weighted Graphs?

- ➢ Each edge carry some weight.

- ➢ Types of Edges :
- <u>Undirected Weighted edges</u>
  -A simple network of friends or bidirectional roads.

- Directed Weighted edges
  - One way road.

- <u>Bi-directional Weighted edges</u>
  - Flight Fares, Two way trips with different costs.

Number of edges

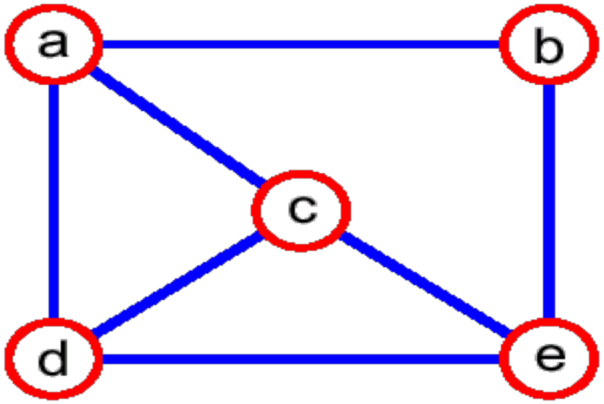
1. Complete Graph O(n^2)
2. Connected Graph
3. Tree

How to implement Graph?

1. Edge List
2. Adjacency matrix
3. Adjacency list

Edge List?

- List of edges and vertices.
- Generally used when we need sorted list of edges.
- Can be created using Linked List.
  list < pair<T,T>>
- Takes more time to find neighbours O(E).

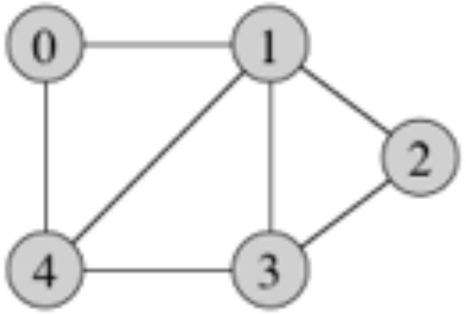


V= {a,b,c,d,e}

E= {(a,b),(a,c),(a,d),
(b,e),(c,d),(c,e),
(d,e)}

Adjacency Matrix?

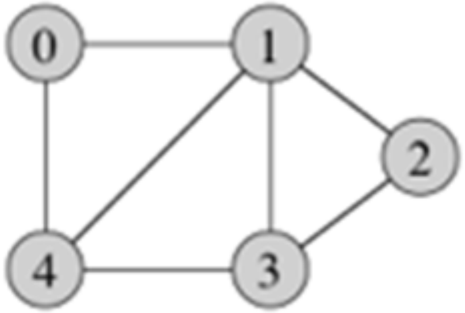
- Stored as VxV matrix, where V : number of vertices.
- Not memory efficient, useful for complete graph.
- Takes linear O(V) time to find neigbours of a given node.
- Useful when we directly want to look presence of edge/ look up for edge weight between two nodes.



$$\begin{array}{c|ccccc|}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 \\
2 & 0 & 1 & 0 & 1 & 0 \\
3 & 0 & 1 & 1 & 0 & 1 \\
4 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}$$

Adjacency List?

- Most widely used and useful.
- Efficiently computes neighbours of a given node.
- Memory Efficient.
- All neighbours are stored in a linked list.
- Can be created using array of linked list (for integer keys only) or hash map.



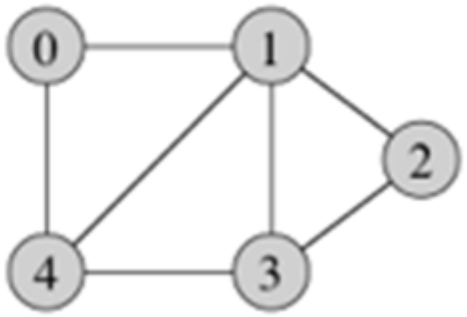
Keys		Values
0	->	1,4
1	->	2,3,4,0
2	->	1,3
3	->	2,1,4
4	->	0,1,3

Adjacency List?

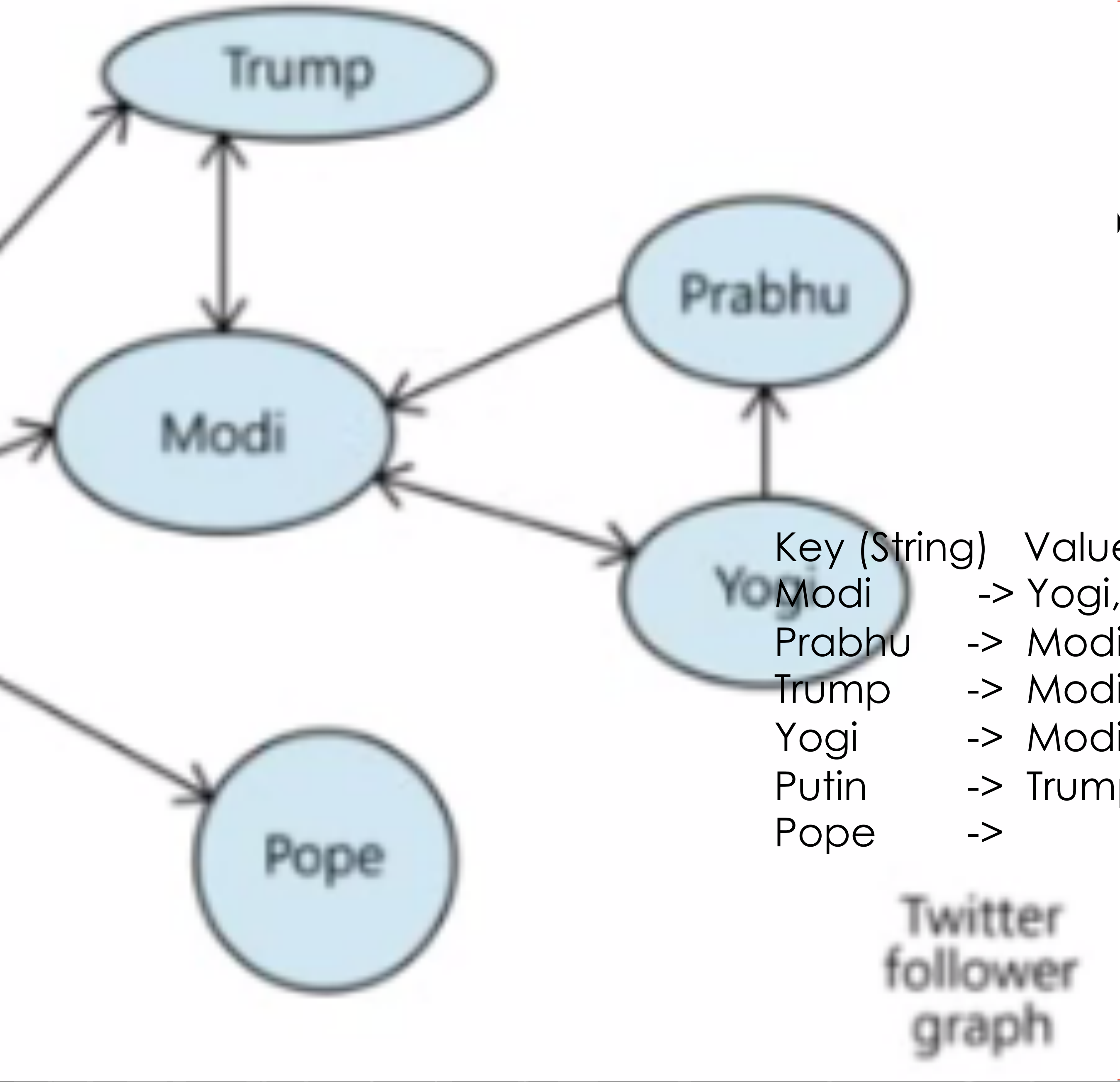
- **Implementation using array of linked list?**
  For Integer Nodes :

  int * a=new int[V]; // this will give array of integers

  list <int>* l=new list<int>[V]; // array of Linkedlist



?



Key (String)		Value(LinkedList<String>)
Modi	->	Yogi, Trump
Prabhu	->	Modi
Trump	->	Modi
Yogi	->	Modi, Prabhu
Putin	->	Trump, Modi, Pope
Pope	->	

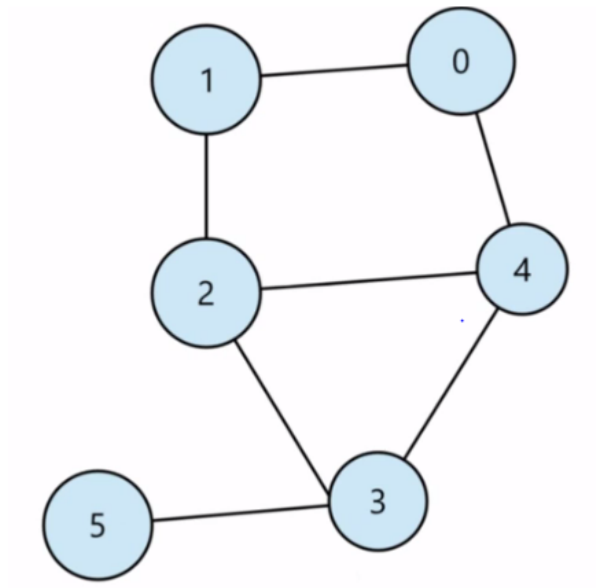
Searching in a Graph

How to Search through a Graph?

1. Breadth First Search
2. Depth First Search

BFS?

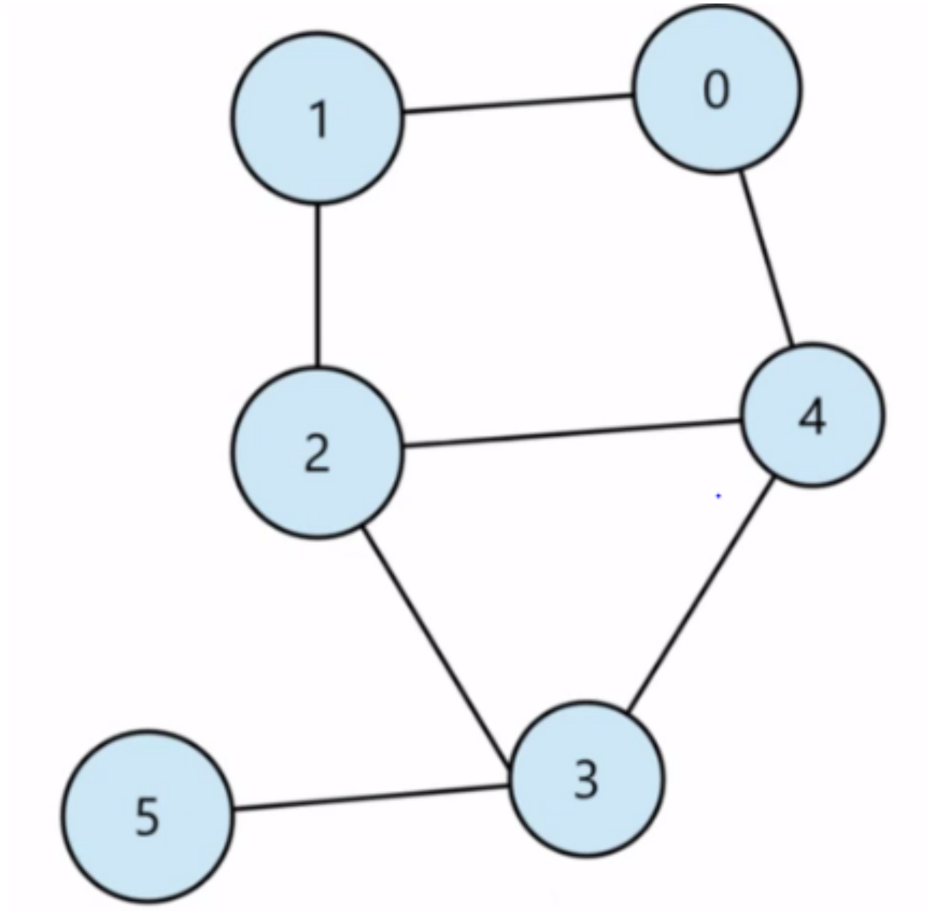
1. It traverses graph, starting from a source vertex, then moving on to the neighbours of the source, then moving on to the next layer and so on.
2. Iterative approach like level order tree traversal.
3. Uses a Queue to maintain **FIFO** ordering, another data-structure (like array/map) is used to maintain the list of vertices visited so far.

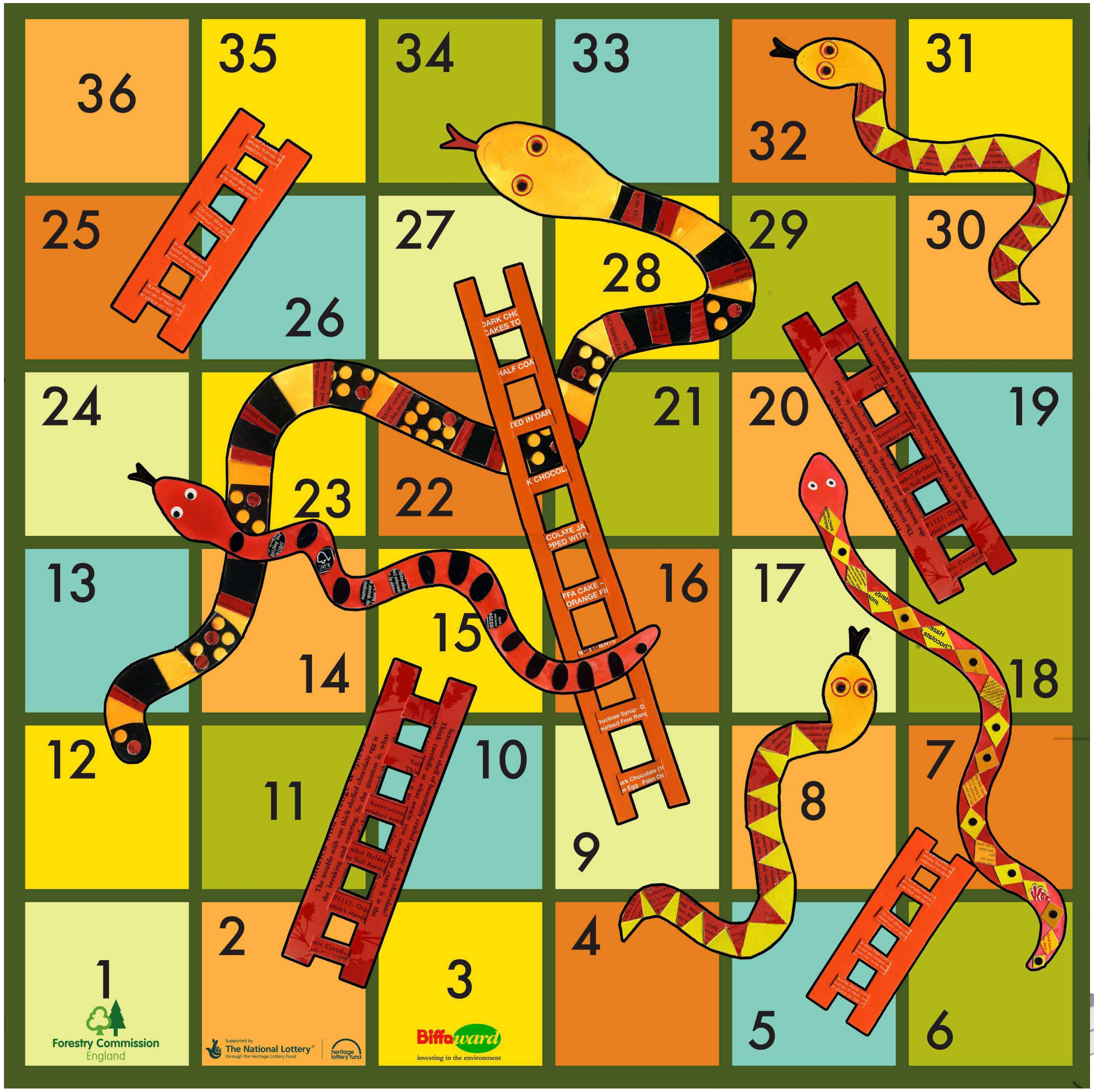


Time Complexity: O(V+E)

Single Source Shortest Path?

1. An application of BFS, to find the minimum distance of all the nodes from a source node.





We are standing at '0'

CODING BLOCKS

Sample Output :

Min moves: 4

One Possible Shortest Path :
1->15->29->30->36

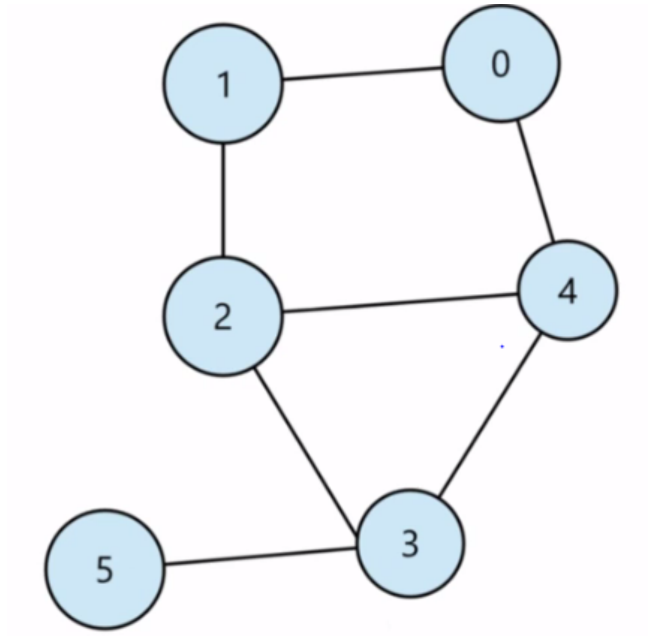
```
int board[50] = {0};
board[2] = 13;
board[5] = 2;
board[9] = 18;
board[18] = 11;
board[17]=-13;
board[20] = -14;
board[24] = -8;
board[25] = -10;
board[32] = -2;
board[34] = -22;
```

DEPTH FIRST SEARCH?

In this we pick one of the neighbors and visits it, and then we traverse further till we reach the vertex from we can't go any further. After that we visits the next unvisited neighbor.

Thus to implement it we need :

- Hash map to maintain visited nodes.
- Recursive call on unvisited nodes.

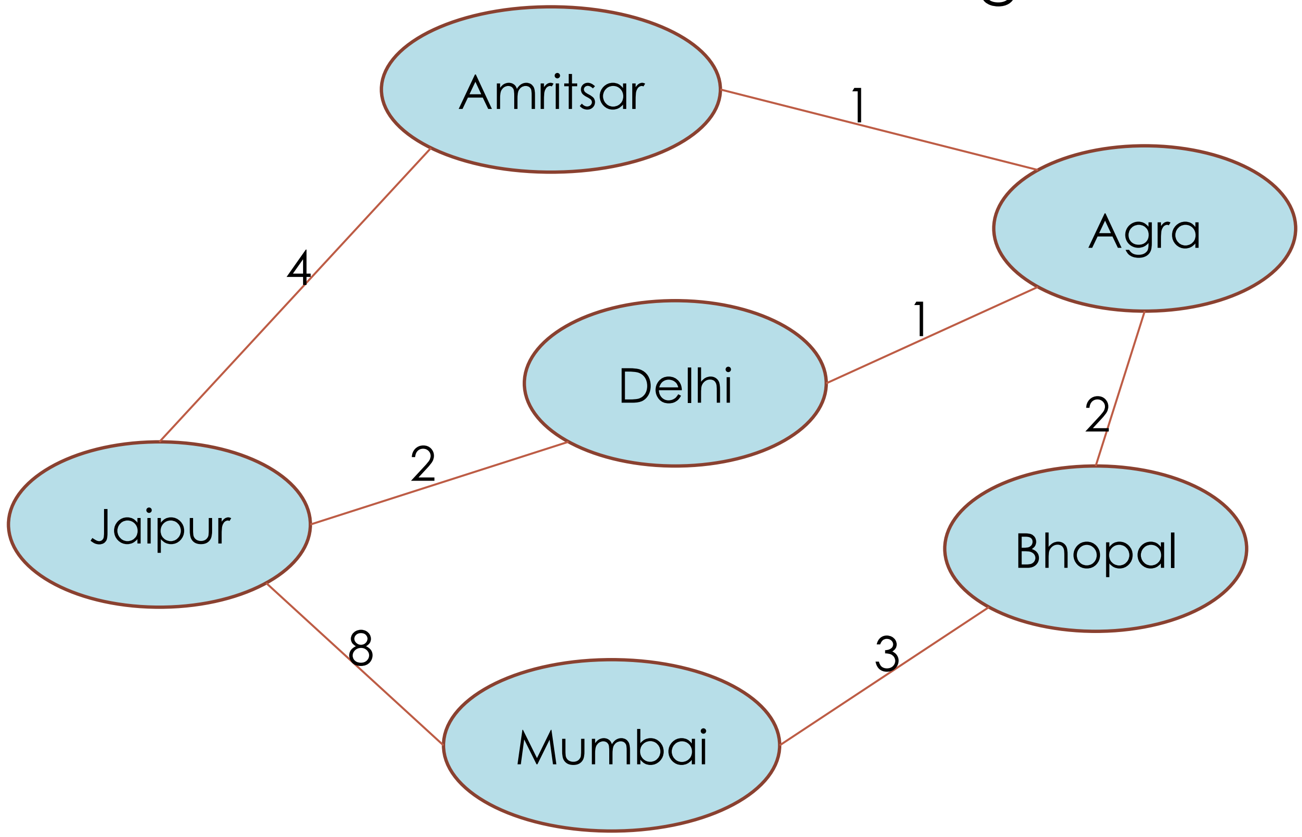


Problems

1. Return all the connected components of the graph
2. Snakes and Ladders Problem.

Dijkstra's Algorithm?

- To find Shortest Path on Weighted Graph.



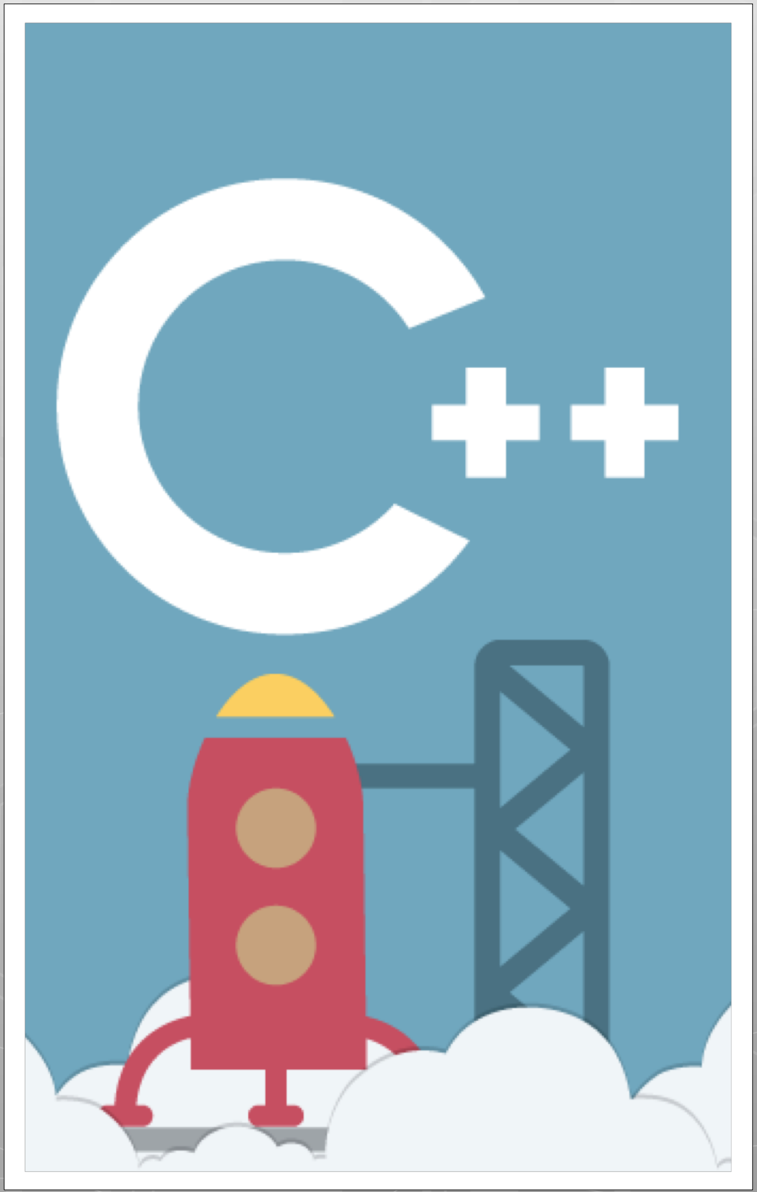
Dijkstra's Algorithm?

- Adjacency List

Keys	**Values**
Amritsar ->	(Delhi,1) (Jaipur,1)
Jaipur ->	(Delhi,2) (Mumbai,8)
Mumbai ->	(Bhopal,3) (Jaipur, 8)

Keys -> string

Values -> list of pair(String, distance)



Thank You!

Kartik Mathur