Competitive Programming

Lec 16, 17, 18

Graphs + Heap

Bhavik Dhandhalya

Why Graphs ?

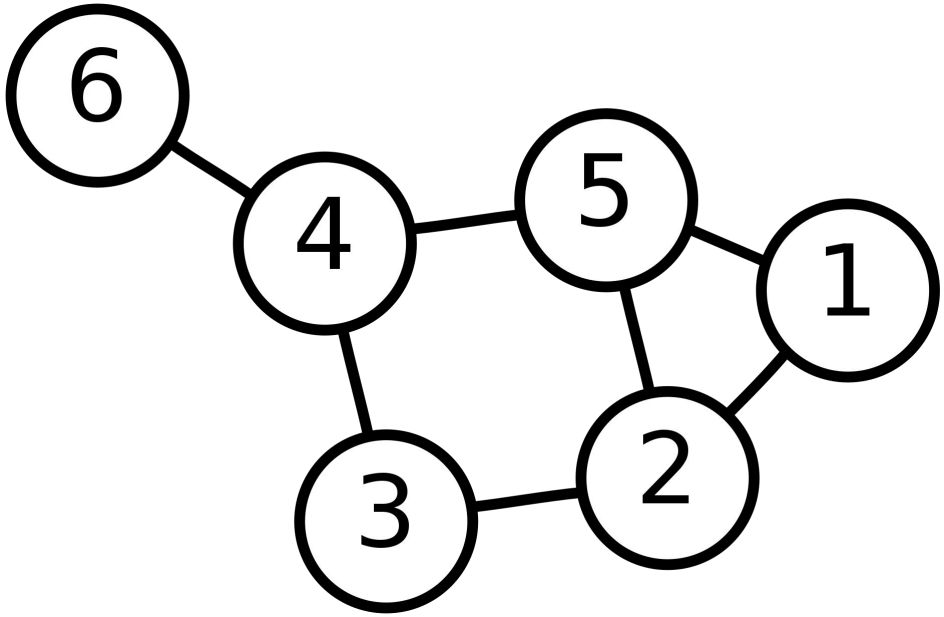
- Facebook
- Linkedin
- Shortest path
- Networking
- Ola, Uber
- Machine learning models
- Shaadi.com
- For placements/interview (very important)

Graph

G = (V, E)
- V = vertices
- E = edges

V = {1, 2, …, 6}

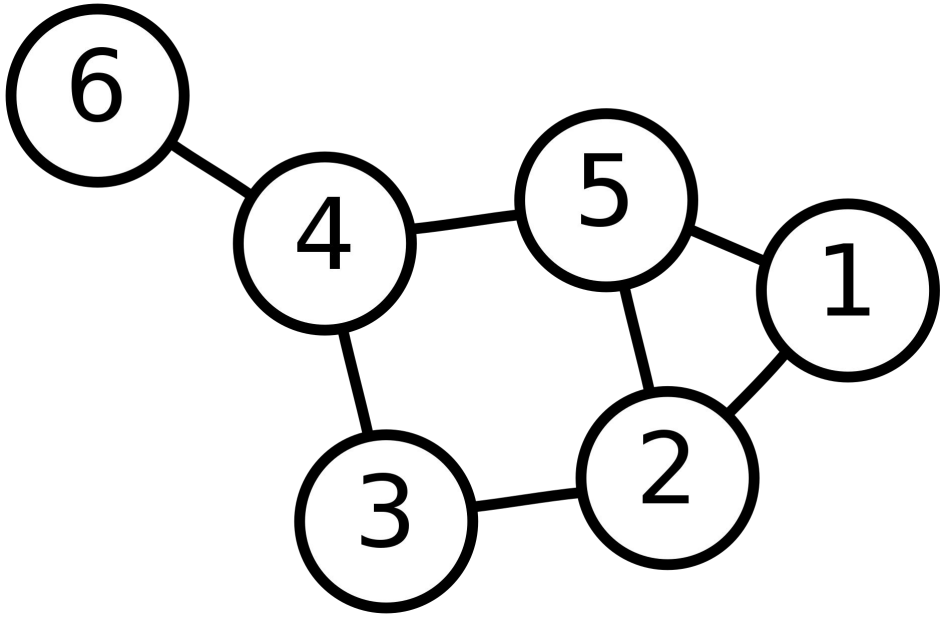
E = {{1,2},{2,3},...}

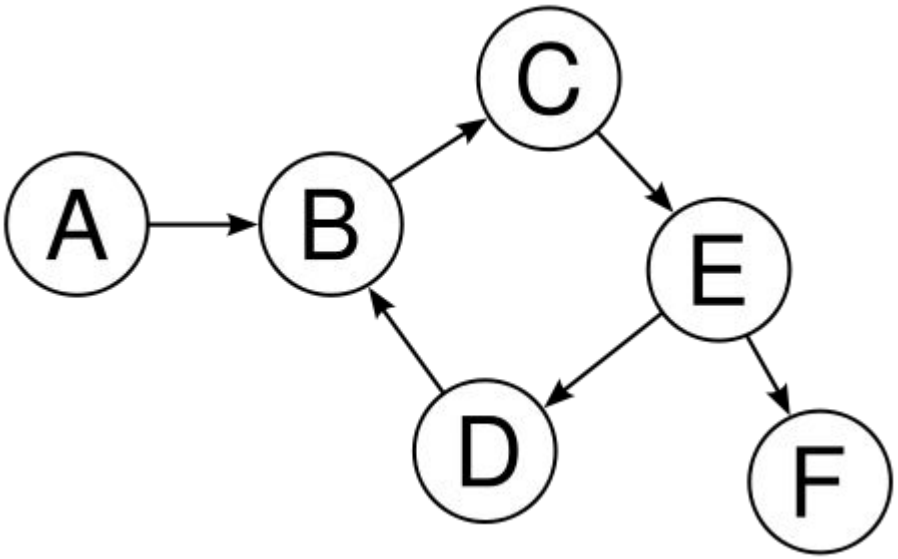


- A graph is a mathematical structure for representing relationships.

Types of Graph

1. Undirected Graph
2. Directed Graph

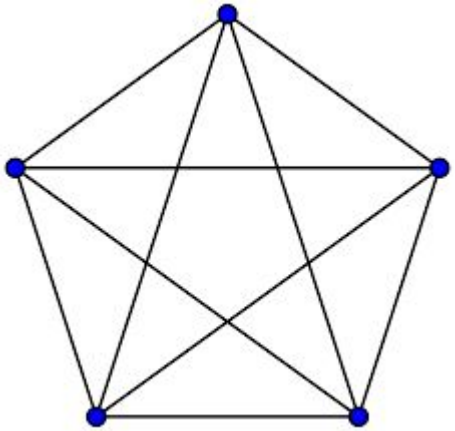


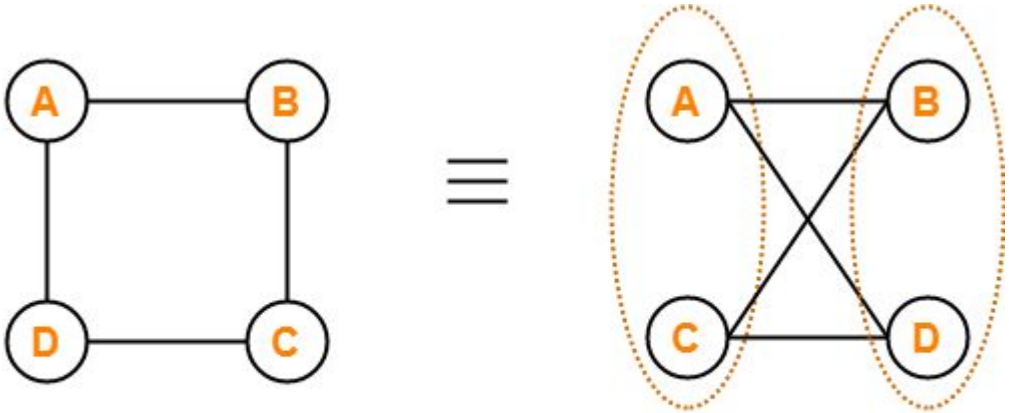


Bhavik Dhandhalya

Types of Graph

3. Complete Graph
4. Bipartite Graph

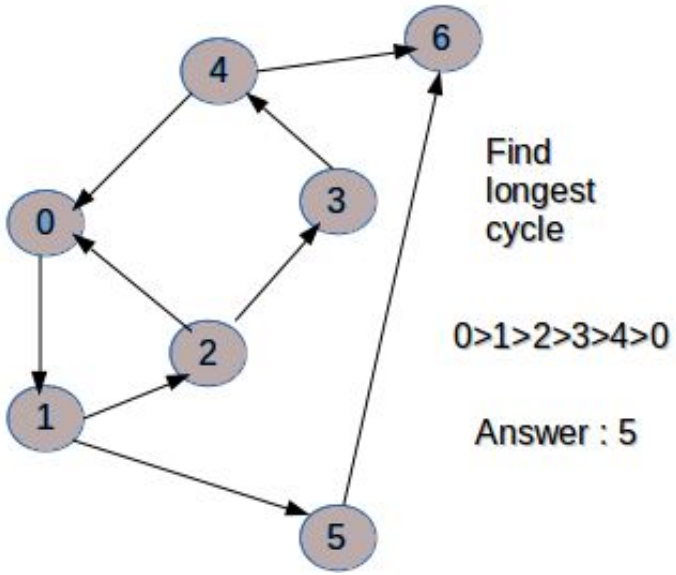


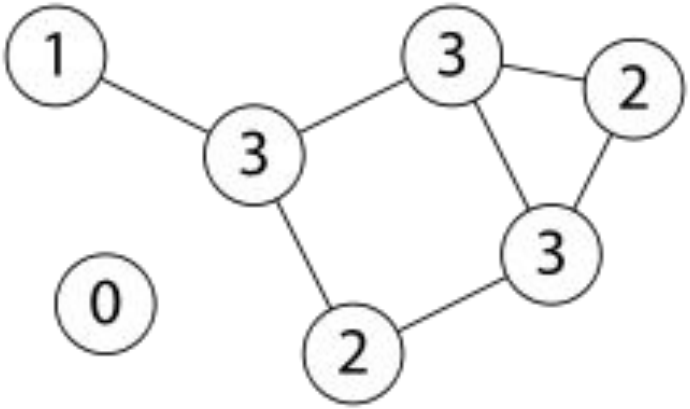


Example of Bipartite Graph

Types of Graph

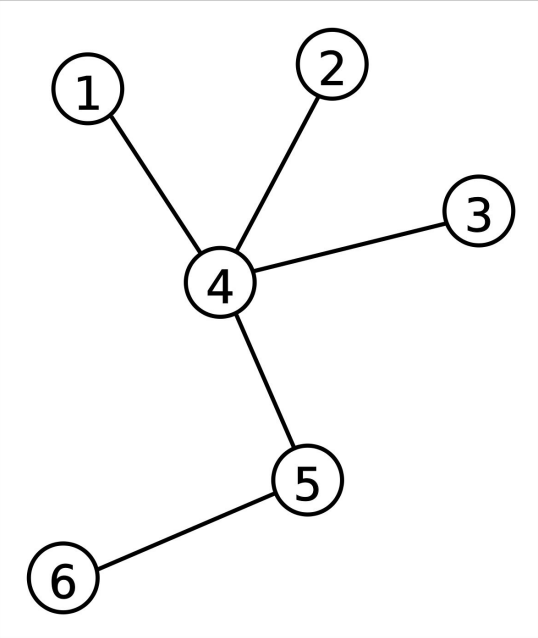
5. Cycle Graph
6. Connected Graph

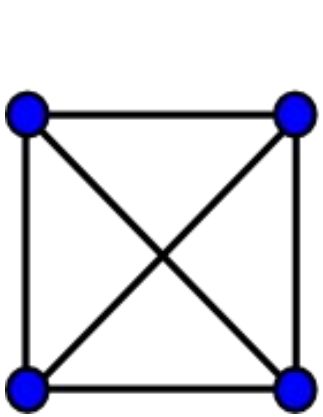


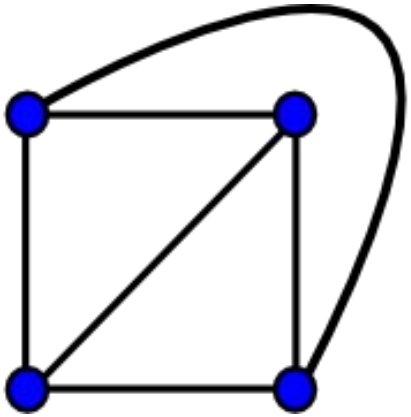


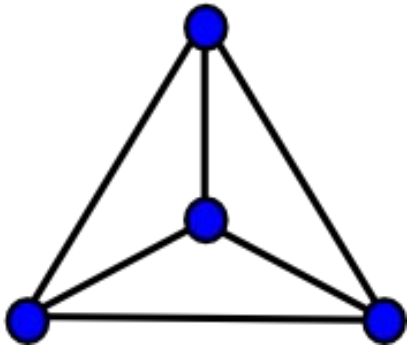
Types of Graph

7. Planar Graph
8. Tree



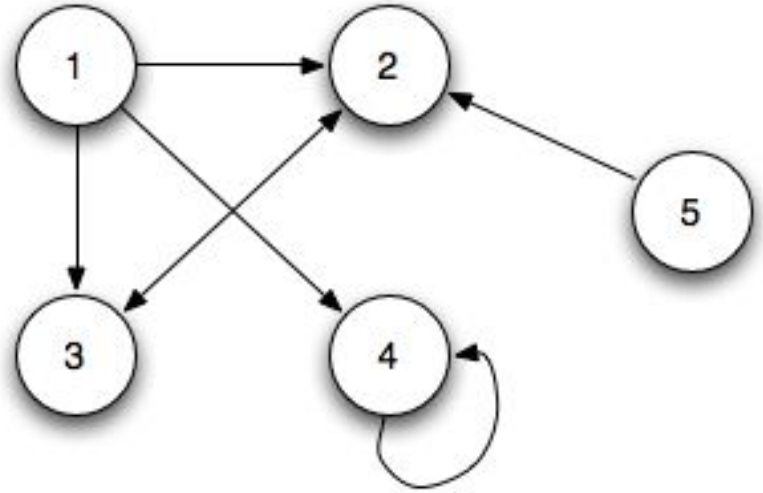


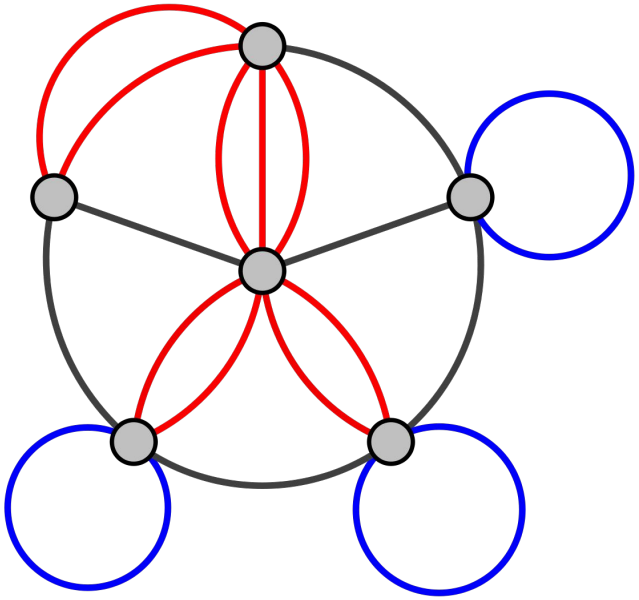




Types of Graph

7. Self-Loop Graph
8. Multigraph





Degree of Node

1. In-degree
2. Out-degree

Graph Representation

1. Adjacency Matrix
   - `Graph[][]`
   - If Node x have edge to Node y,
     Then, for undirected graph `Graph[x][y] = 1`
     `Graph[y][x] = 1`
     For directed graph `Graph[x][y] = 1`

2. Adjacency List
   - `map < int, vector < int > > graph;`
     Or
     `vector < vector < int > > graph;`

   - `graph[x].push_back(y);`

Weighted Graph Representation

1. Adjacency Matrix
   - `Graph[][]`
   - If Node x have edge to Node y,
     Then, for undirected graph `Graph[x][y] = w`
     `Graph[y][x] = w`
     For directed graph `Graph[x][y] = w`

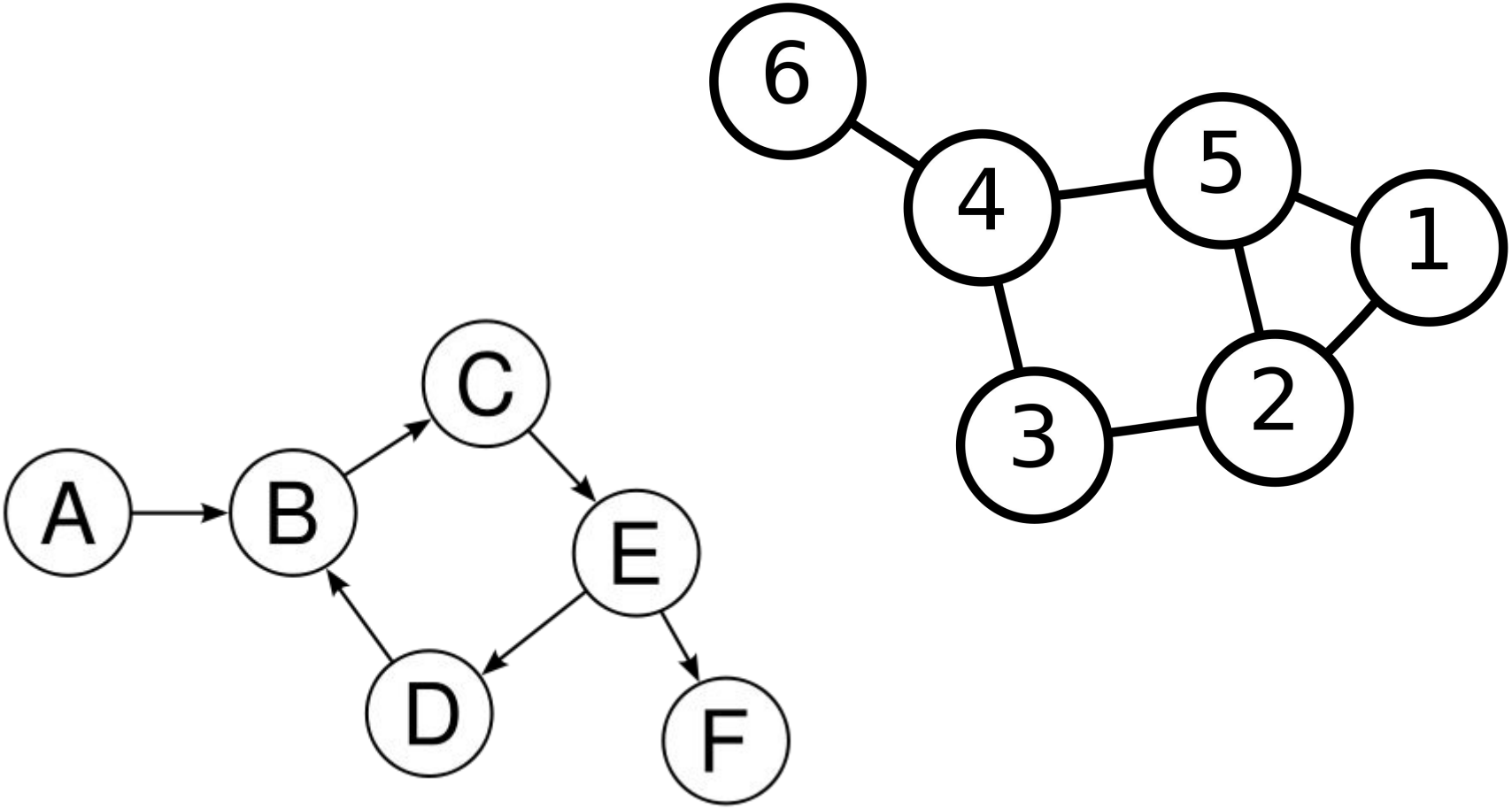
2. Adjacency List
   - `map < int, vector < pair <int, int> > > graph;`
     Or
     `vector < vector < pair <int, int> > > graph;`

   - `graph[x].push_back(y);`

Graph Traversal

Two Approach

1. DFS (Depth first search)
2. BFS (Breadth first search)

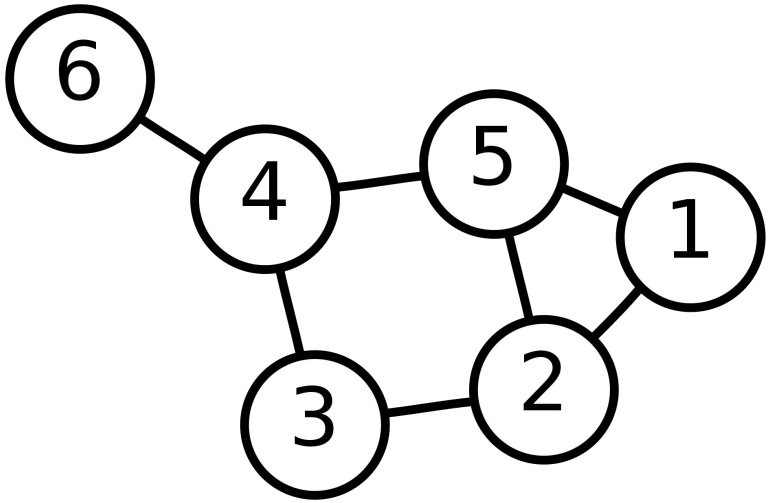


Bhavik Dhandhalya

DFS

```
DFS (currentNode) {
    currentNode = mark visited;

    For (x = all neighbour of currentNode) {
        If (x is not visited)
            DFS(x);
    }
}
```



DFS(starting_Node)

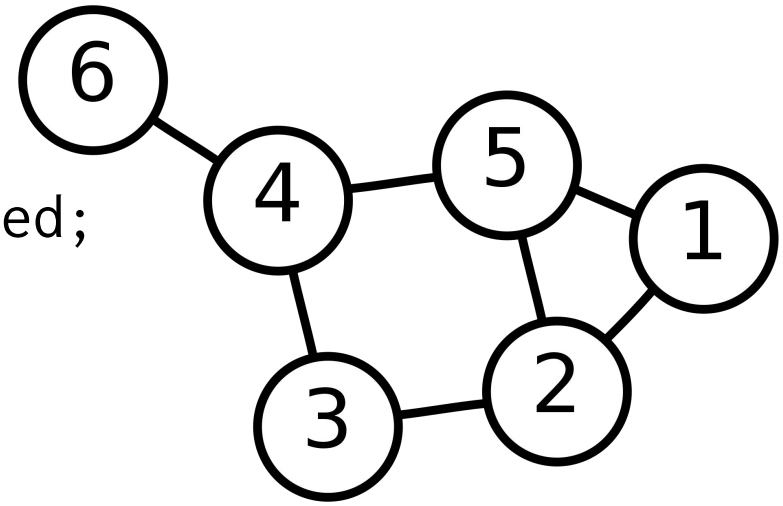
demo

BFS

```
Set All node to “not visited”
Q = new Queue();
Q.insert(starting_Node);
starting_Node = mark as visited;

While (Q is not empty) {
   currentNode = Q.pop();
   print currentNode;

   For (x = all neighbour of currentNode) {
      If (x is not visited)
         Q.push(x);
         x = mark as visited;
   }
}
```



BFS vs DFS

demo

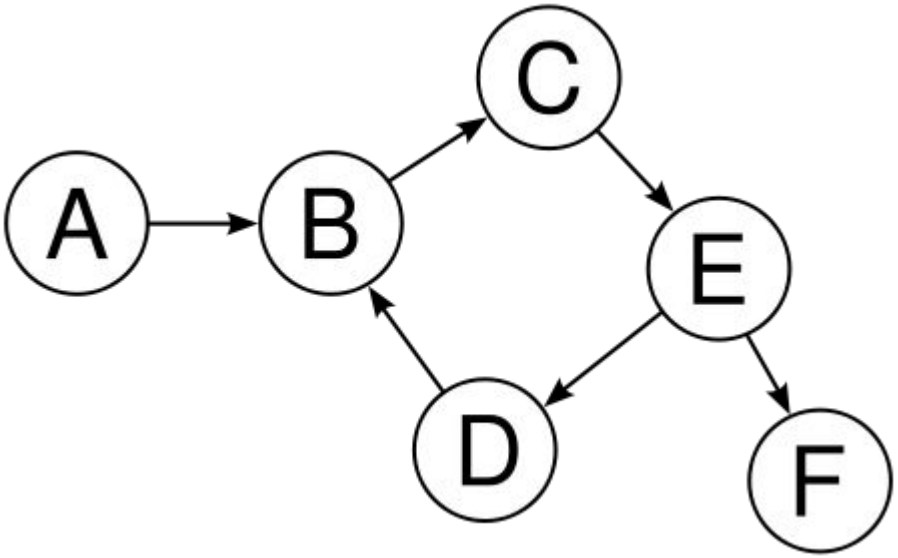
Graph Connectivity

Given a graph, how you will find whether the graph is connected or not ?

Ans - Run DFS from any Node.
Count All not visited Nodes.
If Any Node is not visited,
Then, graph is not connected.

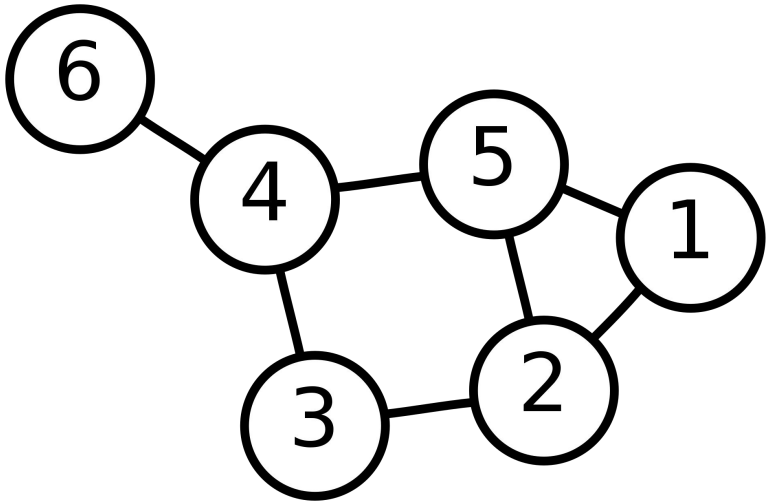
Detect Cycle in directed Graph

Code



Detect Cycle in undirected Graph

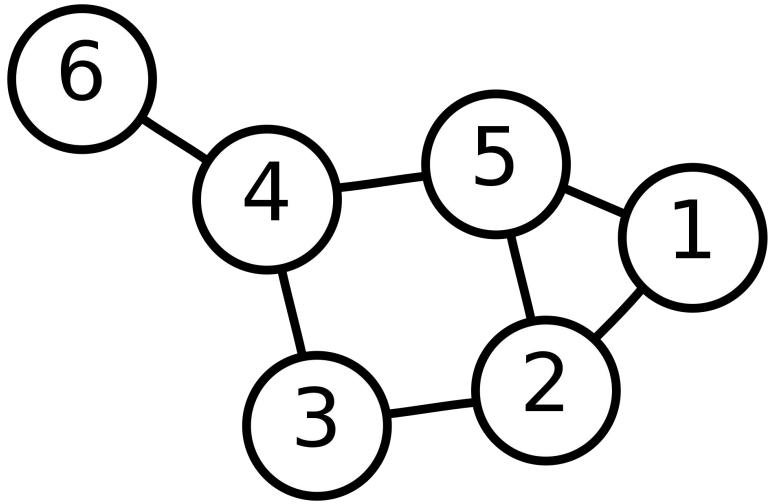
Code



Graph Coloring

Applications,

1. Cycle Detection
2. Bipartite graph



BFS for shortest path

- For unweighted graph, BFS is the fastest algorithm to calculate shortest path
- How ?
- Take example and run BFS on Unweighted or equally weighted graph

Find the number of islands

A group of connected 1's forms an island. The task is to complete the method findIslands() which returns the number of islands present. The function takes three arguments the first is the boolean matrix A and then the next two arguments are N and M denoting the size(N*M) of the matrix A.

```
Input
2
3 3
1 1 0 0 0 1 1 0 1
4 4
1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0

Output
2
2
```

Explanation:
Testcase 1: The graph will look like

```
1 1 0
0 0 1
1 0 1
```

Here, two islands will be formed
First island will be formed by elements {A[0][0] , A[0][1], A[1][2], A[2][2]}
Second island will be formed by {A[2][0]}.

Bhavik Dhandhalya

Find whether path exist

Given a N X N matrix (M) filled with 1, 0, 2, 3. The task is to find whether there is a path possible from source to destination, while traversing through blank cells only. You can traverse up, down, right and left.

A value of cell 1 means Source.
A value of cell 2 means Destination.
A value of cell 3 means Blank cell.
A value of cell 0 means Blank Wall.
Note: there is only single source and single destination.

Solution

Bhavik Dhandhalya

Find whether path exist

Input:
2
4
3 0 0 0 0 3 3 0 0 1 0 3 0 2 3 3
3
0 3 2 3 0 0 1 0 0

Output:
1
0

Testcase 1: matrix is
3 0 0 0
0 3 3 0
0 1 0 3
0 2 3 3

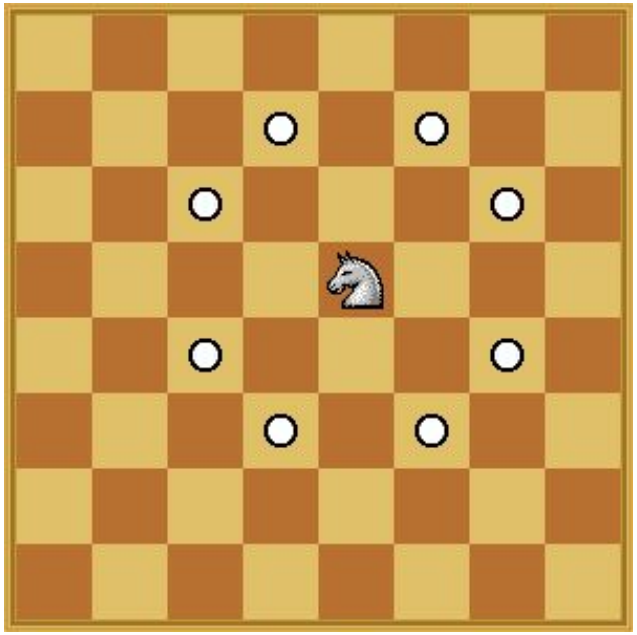
Testcase 2: matrix is
0 3 2
3 0 0
1 0 0
From the matrix we can see that there does not exists any path to reach destination 2 from source 1.

Bhavik Dhandhalya

Knight On Chess Board

Amazon, Goldman Sachs, Google(variation)

Given any source point, (C, D) and destination point, (E, F) on a chessboard, we need to find whether Knight can move to the destination or not.



The above figure details the movements for a knight (8 possibilities).

If yes, then what would be the minimum number of steps for the knight to move to the said point.
If knight can not move from the source point to the destination point, then return -1.
Note: A knight cannot go out of the board.

Solution

Bhavik Dhandhalya

Capture Regions on Board, Google

Given a 2D board containing 'X' and 'O', capture all regions surrounded by 'X'.
A region is captured by flipping all 'O's into 'X's in that surrounded region.

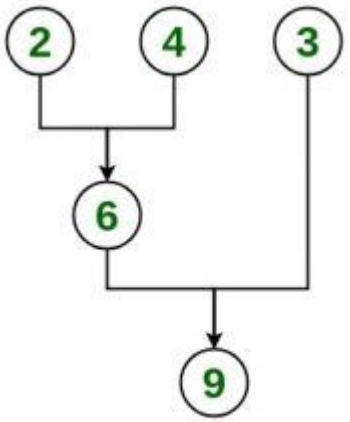
```
Input 1:
    A = [ [X, X, X, X],
          [X, O, O, X],
          [X, X, O, X],
          [X, O, X, X] ]
Output 1:
    After running your function, the board should be:
    A = [ [X, X, X, X],
          [X, X, X, X],
          [X, X, X, X],
          [X, O, X, X] ]
Explanation:
    O in (4,2) is not surrounded by X from below.
```

Solution

Bhavik Dhandhalya

Optimal File Merge Patterns

Given n number of sorted files, the task is to find the minimum computations done to reach Optimal Merge Pattern.
When two or more sorted files are to be merged all together to form a single file, the minimum computations done to reach this file are known as Optimal Merge Pattern.
If more than 2 files need to be merged then it can be done in pairs. For example, if need to merge 4 files A, B, C, D. First Merge A with B to get X1, merge X1 with C to get X2, merge X2 with D to get X3 as the output file.
If we have two files of sizes m and n,
the total computation time will be m+n.
Here, we use greedy strategy by merging two
smallest size files among all the files present.



Input: n = 3, size = {2, 3, 4}
Output: 14

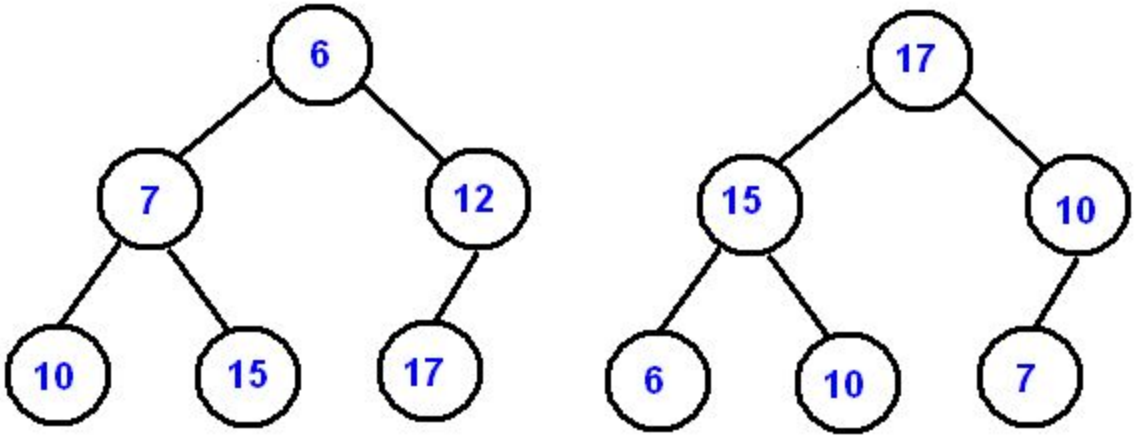
Cost = 6 + 9 = 15

Heap

Two type of heaps,

1. Min heap
2. Max heap

HeapSort complexity O(NlogN)



Applications of Heaps

1. Priority based scheduling in Operating System
2. Dijkstra’s Shortest path algorithm
3. Huffman coding

Heap in C++

Max heap

```
priority_queue <int> pq;
```

Min heap

```
priority_queue <int, vector<int>, greater<int> > pq;
```

Magician and Chocolates

Given N bags, each bag contains Ai chocolates. There is a kid and a magician. In one unit of time, kid chooses a random bag i, eats Ai chocolates, then the magician fills the ith bag with floor(Ai/2) chocolates.Given Ai for 1 <= i <= N, find the maximum number of chocolates kid can eat in K units of time.

K = 3

N = 2

A = 6 5

Return: 14

At t = 1 kid eats 6 chocolates from bag 0, and the bag gets filled by 3 chocolates

At t = 2 kid eats 5 chocolates from bag 1, and the bag gets filled by 2 chocolates

At t = 3 kid eats 3 chocolates from bag 0, and the bag gets filled by 1 chocolate

so, total number of chocolates eaten: 6 + 5 + 3 = 14

Note: Return your answer modulo 10^9+7

Solution

Bhavik Dhandhalya

Dijkstra’s Shortest path Algo

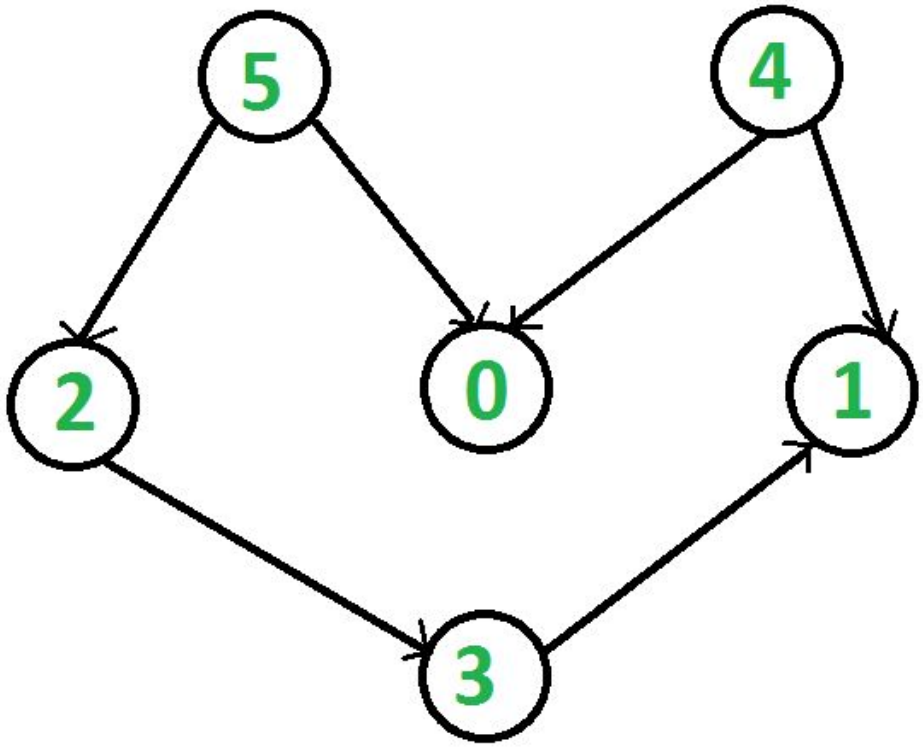
Code using Priority_queue

Code using matrix

Topological Sorting

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv, vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

Possible order
4 5 2 3 1 0



Solution

Strongly Connected Components

A directed graph is strongly connected if there is a path between all pairs of vertices.

Go for Tarjans's SCC.

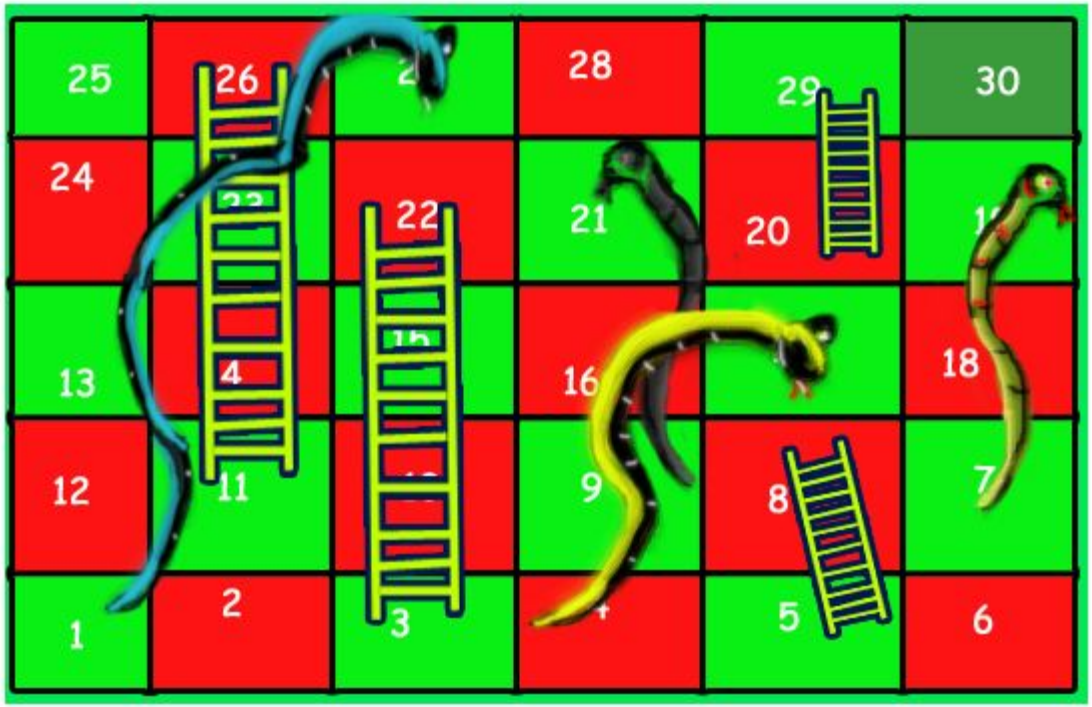
Follow this awesome video.

Snake and Ladder Problem

Given a snake and ladder board of order 5x6, find the minimum number of dice throws required to reach the destination or last cell (30th cell) from source (1st cell) .

For the above board output will be 3
For 1st throw get a 2
For 2nd throw get a 6
For 3rd throw get a 2

Solution



Circle of strings

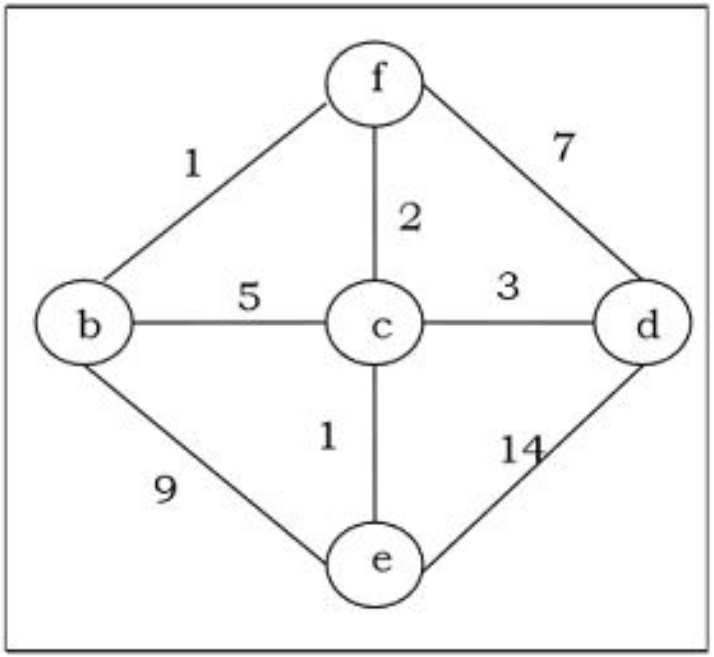
Given an array of strings A[], determine if the strings can be chained together to form a circle. A
string X can be chained together with another string Y if the last character of X is same as first
character of Y. If every string of the array can be chained, it will form a circle.
For eg for the array arr[] = {"for", "geek", "rig", "kaf"} the answer will be Yes as the given strings can be chained as "for", "rig", "geek" and "kaf".
Input
2
3
abc bcd cdf
4
ab bc cd da
Output
0
1

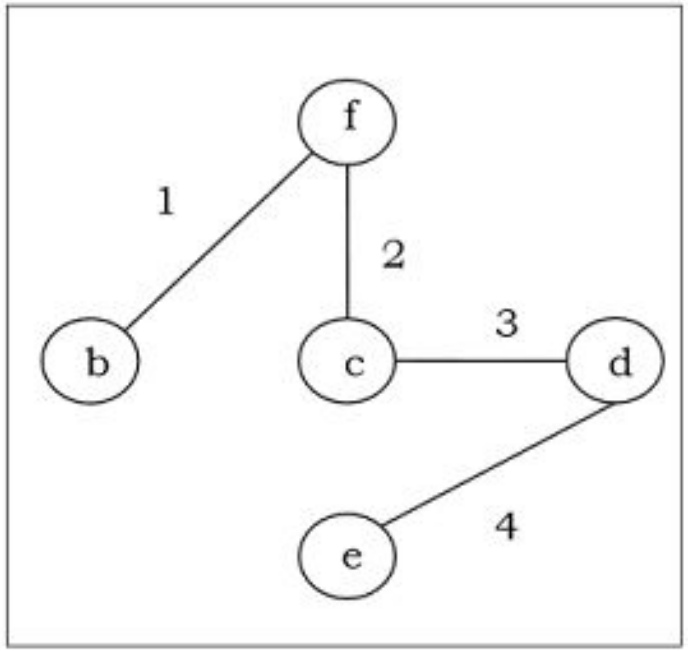
Solution

Bhavik Dhandhalya

Minimum Spanning Tree

1. Prim's Algo
2. Krushkal's Algo





Homework Algos

Union Find (Disjoint Sets)
Bellman ford
Floyd Warshall

Bhavik Dhandhalya

Homework

Level Order	Easy
Black Shapes	Easy, similar to Find no. of island
LRU Cache	Medium, Implementation is hard
Course Schedule	Medium, Topological sort
Word Ladder I	Google
Alien Dictionary	HARD, Microsoft still ask this question

References

- https://cs.stanford.edu/people/abisee/gs.pdf
- https://web.stanford.edu/class/cs97si/06-basic-graph-algorithms.pdf
- All demos - https://cs.stanford.edu/people/abisee/tutorial/