2nd year and 1st sem of CSE

Graph

Note by

Moloy Ghosh

Wed, 17-May-2023

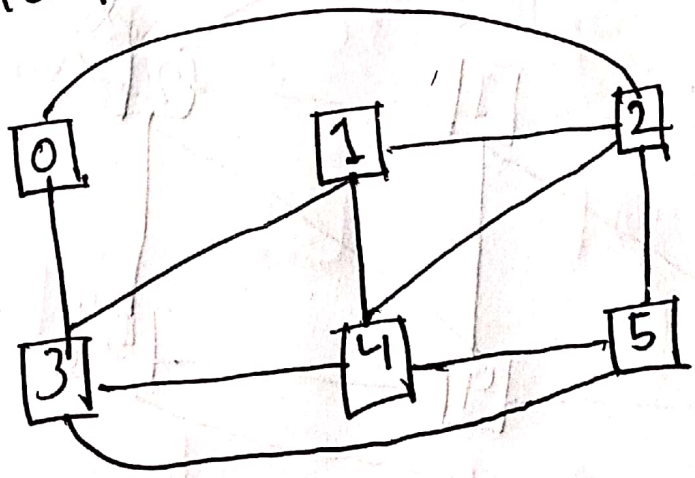
From Books and class note

Graph

A graph is a collection of interconnected nodes or vertices connected by edges or arcs. It is a non-linear data structure used to represent relationships or networks.

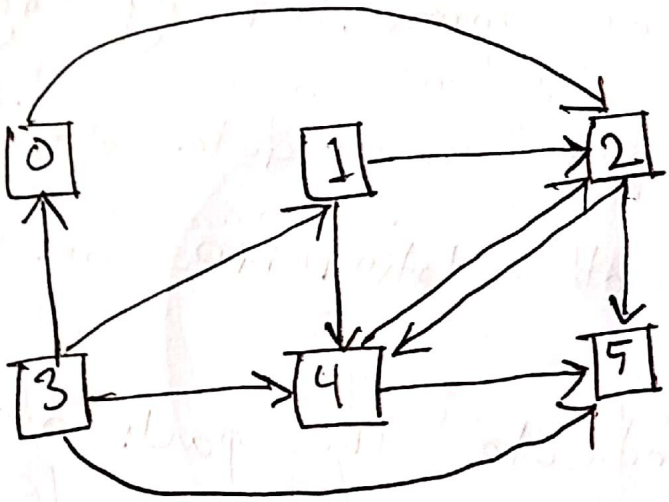
Sometimes we indicate the parts of a graph by writing $G = (V, E)$

Undirected Graph: An undirected graph is a type of graph in which the edges have no orientation or direction

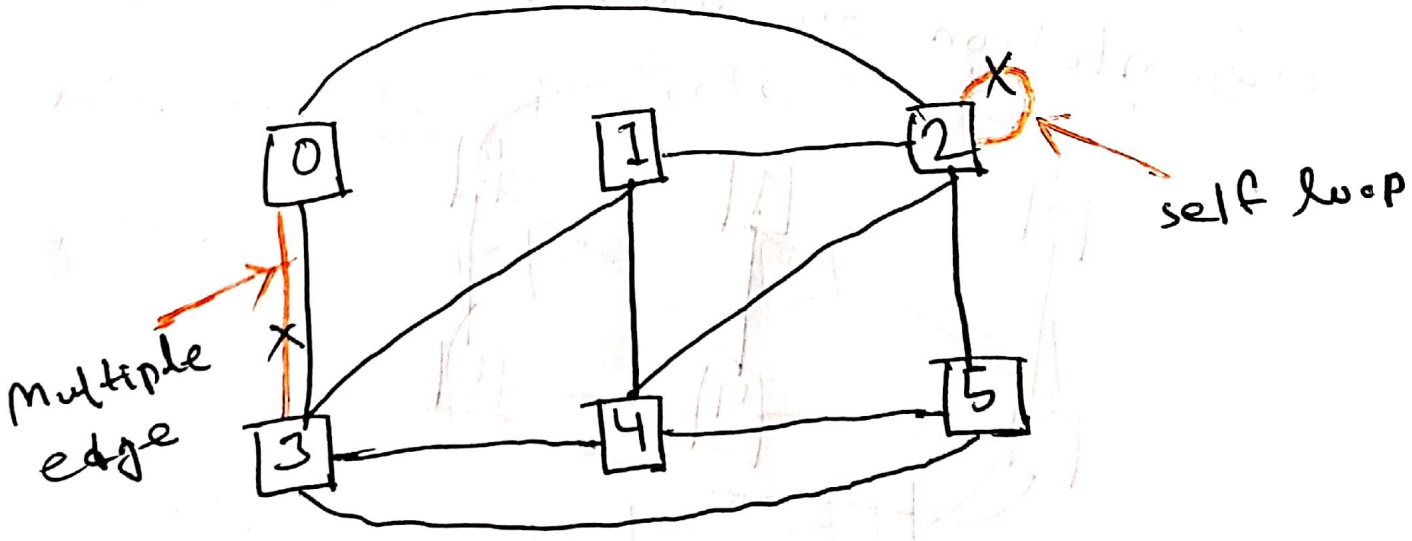


Sabitar™

Directed Graph: In which the edges have a specific direction, ~~inedt~~ including a one-way relationship between two vertices



Simple graph: Multiple edges between pair of nodes and self loops are not allowed in simple graphs.

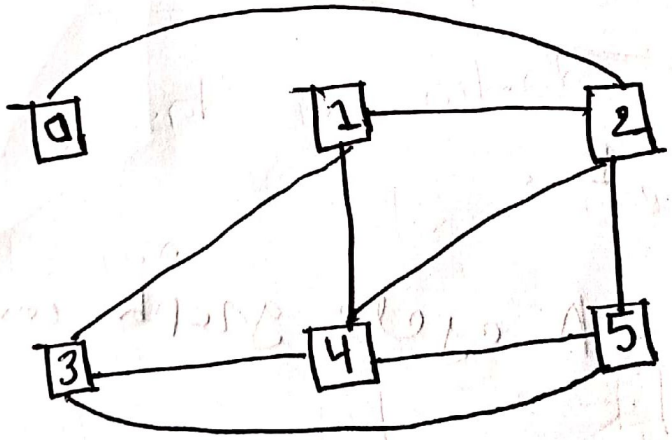


Path: The path of a graph is a sequence of connected vertices, where each vertex is directly linked to the next vertex in the sequence.

A path P of length n from a node u to a node v is defined as a sequence of n+1 nodes.

$$P = (V_0, V_1, V_2, \ldots, V_n)$$

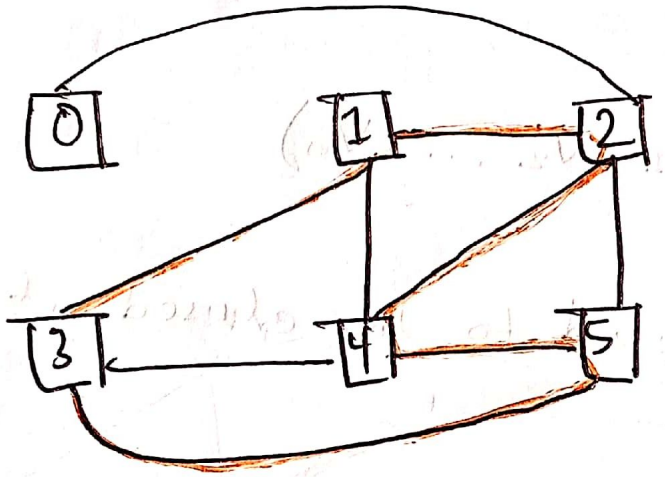
The path P is said to be closed if $V_0 = V_n$



0, 2, 5, 3 is a path. but,

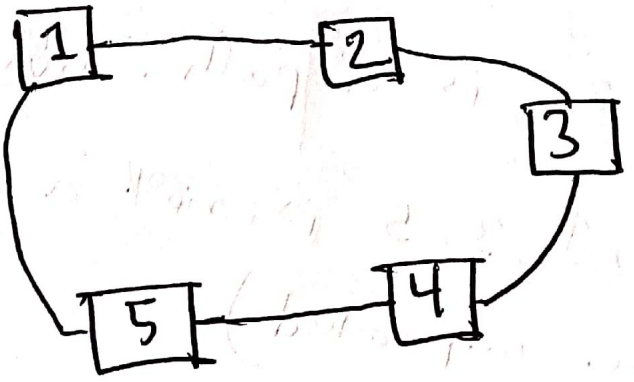
1, 2, 4, 1, 3, 5 is not a path. (some nodes are repeated)

Cycle: A cycle is a closed path in a graph meaning it starts and ends at the same vertex and does not repeat any other vertex (except for the starting and ending vertex). It's also known as circuit



1, 2, 4, 5, 3, 1.

Cycle Graph: A cycle graph consists of a single cycle.

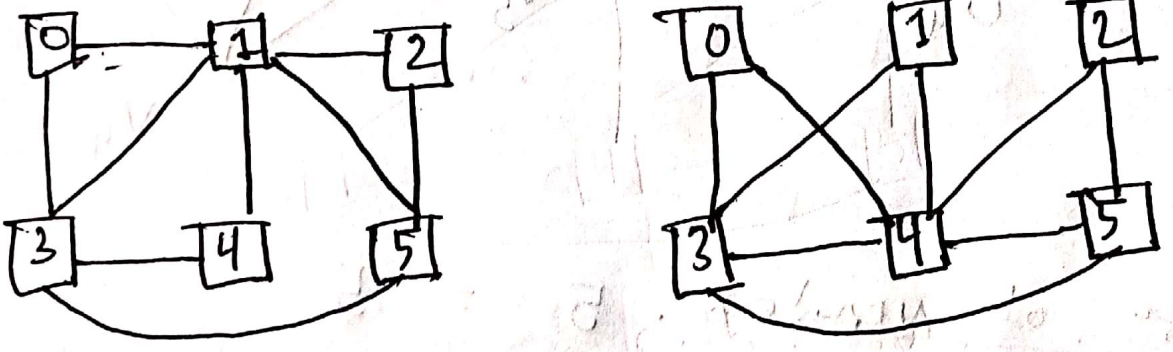


Tree: Trees are acyclic (no cycle) graphs that are connected (a path between every pair of edges)

In an N node ~~tree~~, how many edges are there? (N-1) ~~nodes~~ edges

Isomorphic Graph:

Two graphs are called isomorphic if they are the same if ~~same~~ renaming the nodes will result in indentical graphs.



Vertex 1 corresponds to vertex 4

দুইটি গ্রাফ Isomorphic কিনা নির্ণয়:

শর্ত:

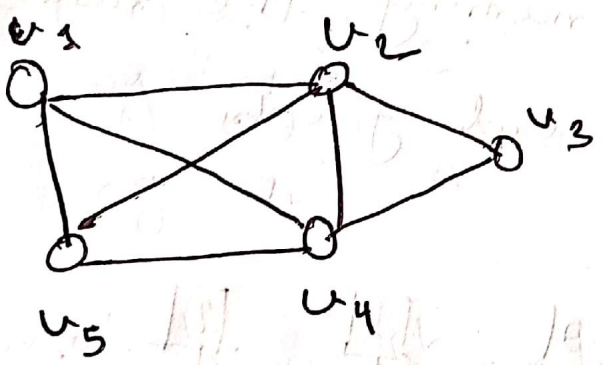
(i) Number of vertex সমান

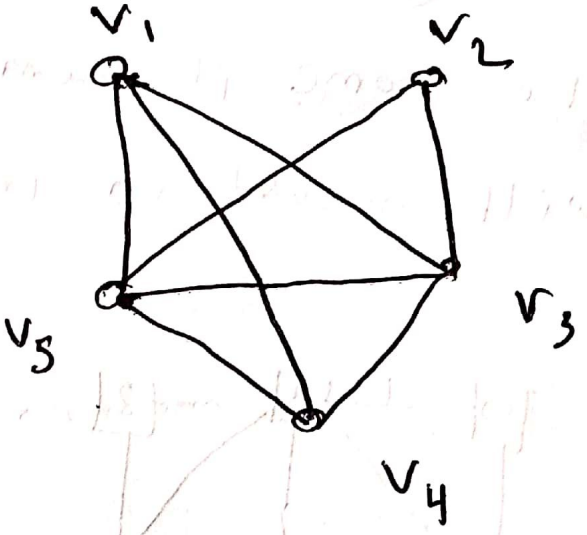
(ii) Number of edge সমান

(iii) Degree sequence সমান

(iv) Mapping -ও থাকতে। (কোথাও কোনো সাথে কোথাও কোনো কানেকশন)

সমাধান:





Number of vertex: 5 = 5

" " edeg: 8 = 8

Degree sequence:

$u_1 = 3$ (ডিগ্রি ক্রম লক্ষ)
$u_2 = 4$
$u_3 = 2$
$u_4 = 4$
$u_5 = 3$

$v_2 = 3$
$v_2 = 2$
$v_3 = 4$
$v_4 = 3$
$v_5 = 4$

sequence:
(4, 4, 3, 3, 2) = (4, 4, 3, 3, 2)

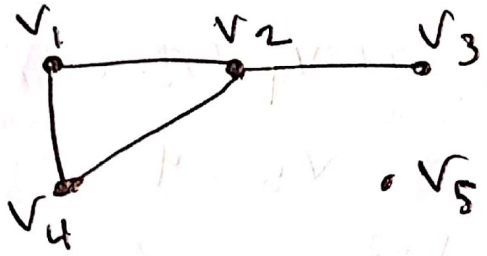
Mapping:

$u_1 = v_1$
$u_2 = v_3$
$u_3 = v_2$
$u_4 = v_5$
$u_5 = v_4$

এখানে, দেখতে হবে, u_1 ও u_2 edge এর ক্রমানুসারে মিল আছে কিনা। যদি থাকে তবে v_1 ও v_3 এরও মিল থাকতে হবে। এভাবে প্রতিটি চেক করতে হবে। না মিললে কিছু নোড পরিবর্তন করতে হবে। মিলে গেলে প্রদত্ত দুইটি Isomorphic.

তাহলে এটি ISOMORPHIC

Degree: The degree of a node in an undirected graph is the number of edges incident to the node.



$\deg(v_1) = 2$

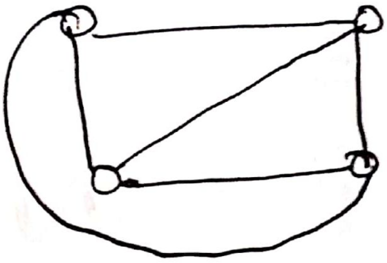
$\deg(v_2) = 3$

$\deg(v_3) = 3$

$\deg(v_4) = 2$

$\deg(v_5) = 0$

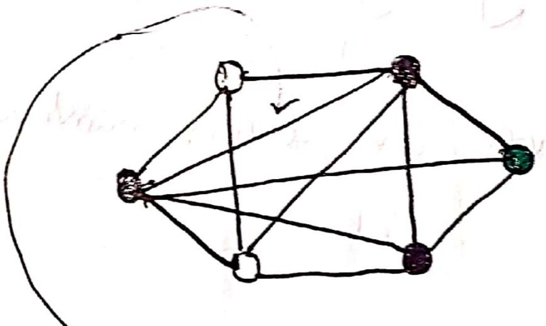
Complete graph: The minimum number of edges in a graph is 0. A graph with the maximum number of edges is called a complete graph. The graph below is a complete undirected graph with four nodes.



Bipartite Graph:

A bipartite graph (or bigraph) is a graph whose vertices can be devided into two disjoint sets and (this is, and are each independent sets) such that every edges connects a vetex in to one in. vertex set and are often denoted as partite set.

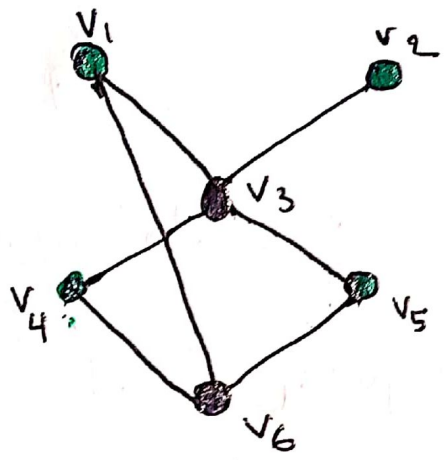
নির্ণয়:



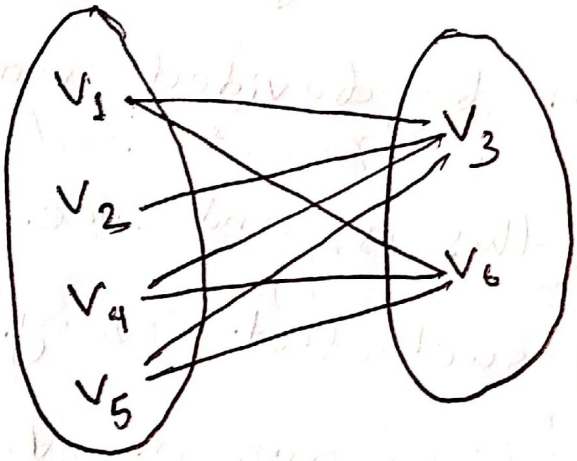
প্রথমে যেকোনো একটি নোড এক কালার করলাম। তার নেইবার নোড আছে অর্থাৎ এই নোডের সাথে ডাইরেক্ট কানেকশন যেগুলোর আছে নোডগুলোকে অন্য কালার করলাম। যদি একই কালারে দুটো কানেকশন থাকে তবে তা যেমনটাই হোক ও Bipartite graph না। তাই এটি

NOT BIPARTITE

Sabitar™



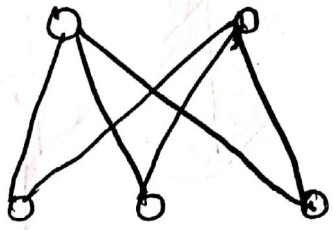
ए॰ Bipartite ग्राफ :

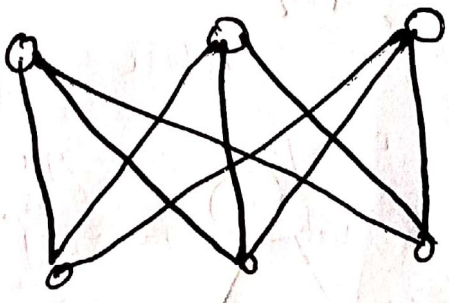


Bipartite graphs have a number of interesting properties. For example, any cycle in a bipartite graph must have an even number of edges.

Complete Bipartite graph: A complete bipartite graph or biclique is a special kind of ~~big~~ bipartite graph where every vertex of the first set is connected to every vertex of the second set.

A complete bipartite graph has all possible edges. so for example, if one partition has K nodes and the other has M nodes, nodes in the K node partition will have degree M and the nodes in the M node partition will have degree K.





Subgraphs:

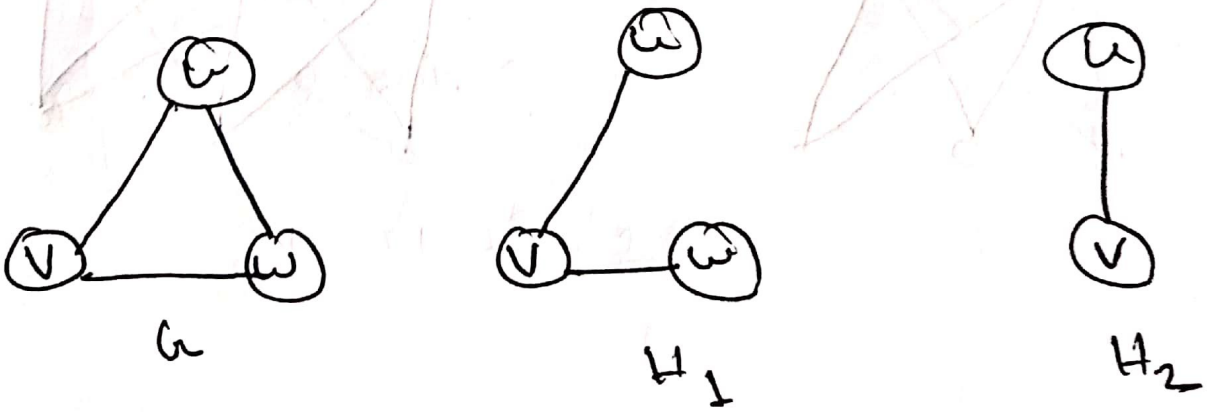
A subgraph of a graph $G = (V, E)$ is a graph $H = (V', E')$ where V' is a subset of V and E' is a subset of E

Application example: solving sub-problems within a graph

Representation example: $V = \{u, v, w\}$,

$E = (\{u, v\}, \{v, w\}, \{w, u\})$,

H_1, H_2



- $G = G1 \cup G2$ wherein $E = E1 \cup E2$ and $V = V1 \cup V2$, G, G_1 and $G2$ are simple graph of G

Definition: A subgraph is a graph that is formed by taking a subset of the vertices and edges of a larger graph. In other words, a subgraph is a smaller graph that is contained within a longer graph.

edges: An edge is a single connection between two vertices.

* How is the degree of a graph related to the number of edges?

sum of degrees of all vertices =

2* number of edges

* what is the degree of each node in an N node complete graph?

N-1

* what is the sum of the degree in an N node complete graph?

N*(N-1)

* How many edges are there in an N node complete graph?

N(N-1)/2

Graph Representation

1. Adjacency Matrix
2. Incidence Matrix
3. Adjacency List

~~Adjacent~~ Adjacency Matrix

~~For undirected graph~~

(i) No of vertices (n) কে row করে।

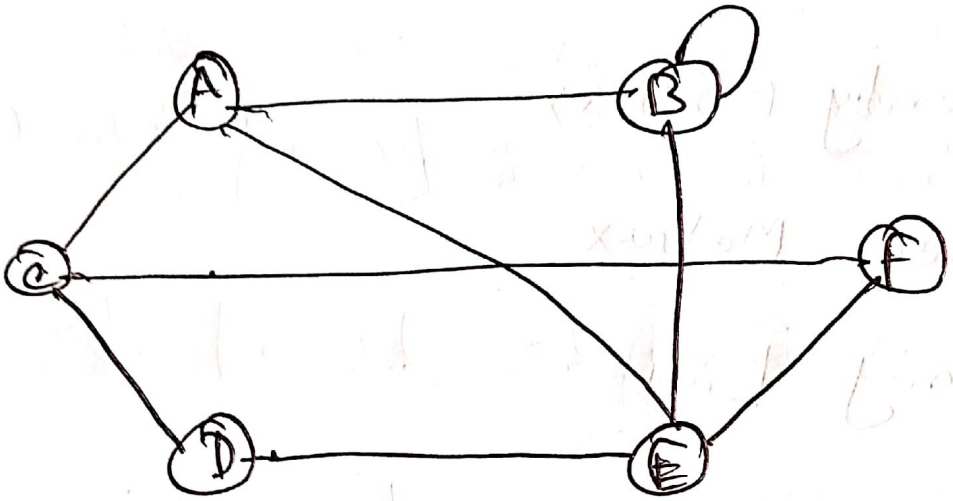
(ii) square matrix (n×n) represent করে।

(iii) if adj [i,j] = 1 (loop, i,j adjacent হয়)

(iv) if adj [i,j] = 0 (i,j adjacent না হয়)

adjacent মানে হলো সংলগ্ন। অর্থাৎ যেমন A হলো, B, C এর সংলগ্ন

For undirected graph:

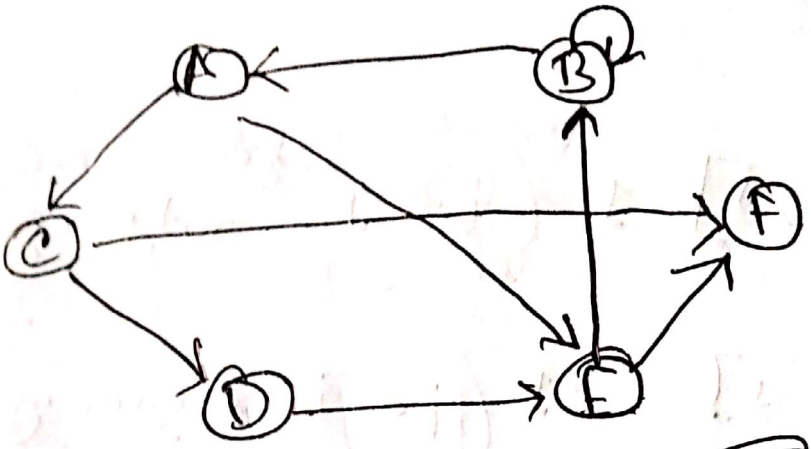


এখন এনুযায়ী, এখানে vertices 6

square matrix represent এ 6×6 matrix

(I) ↓ \ (J) →	A	B	C	D	E	F
A	0	1	1	0	1	0
B	1	1	0	0	1	0
C	1	0	0	1	0	1
D	0	0	1	0	1	0
E	1	1	0	1	0	1
F	0	0	1	0	1	0

For Directed graph:



B ka output

D ka output

	A	B	C	D	E	F
A	0	0	1	0	0	0
B	1	1	0	0	0	0
C	0	0	0	1	0	1
D	0	0	0	0	1	0
E	0	1	0	0	0	1
F	0	0	0	0	0	0

B ka input

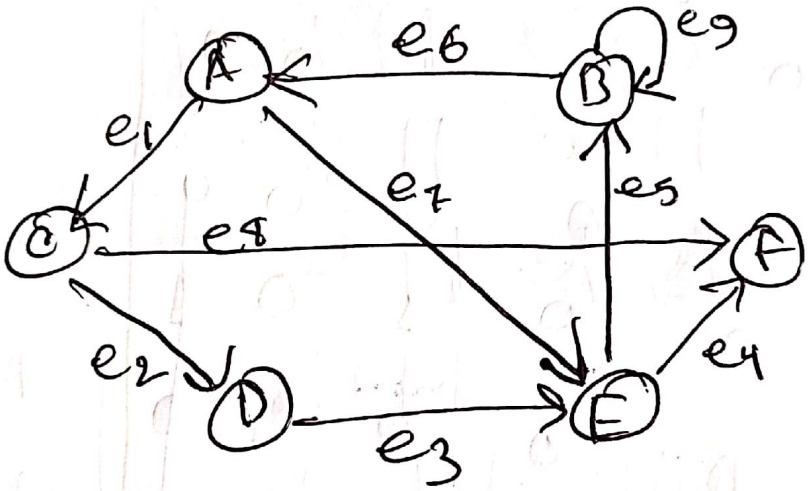
D ka input

Incidence Matrix

(i) adj [i, j] = 1 ($i \to j$ edge outgoing)

(ii) adj [i, j] = -1 ($i \leftarrow j$ incoming)

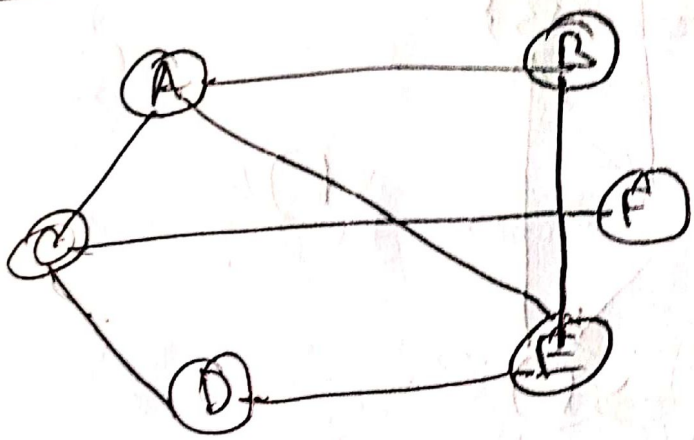
(iii) adj [i, j] = 0 (no connection)

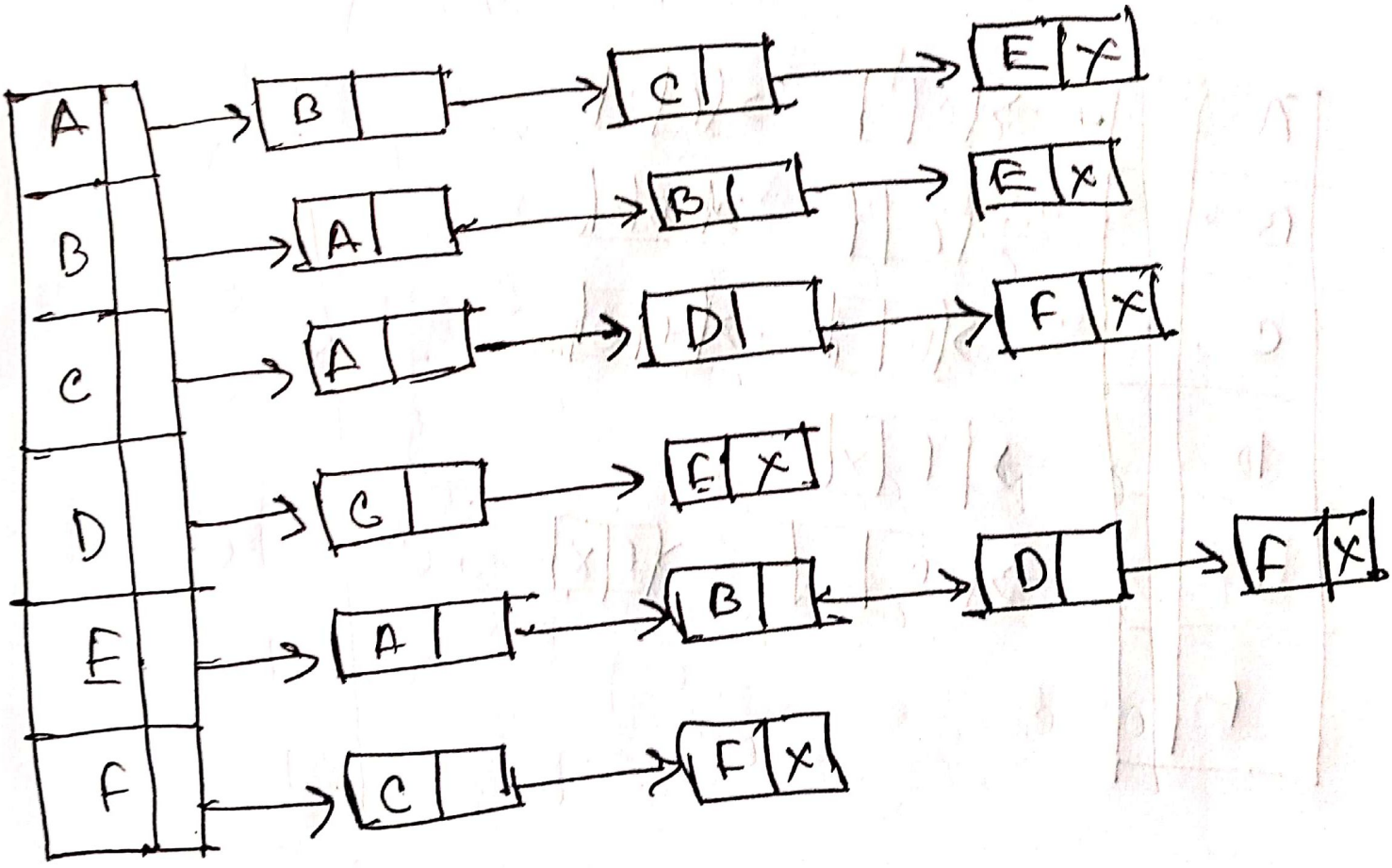


	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
A	1	0	0	0	0	-1	1	0	0
B	0	0	0	0	-1	1	0	0	1
C	-1	1	0	0	0	0	0	1	0
D	0	-1	1	0	0	0	0	0	0
E	0	0	-1	1	1	0	-1	0	0
F	0	0	0	-1	0	0	0	-1	0

Adjacency List

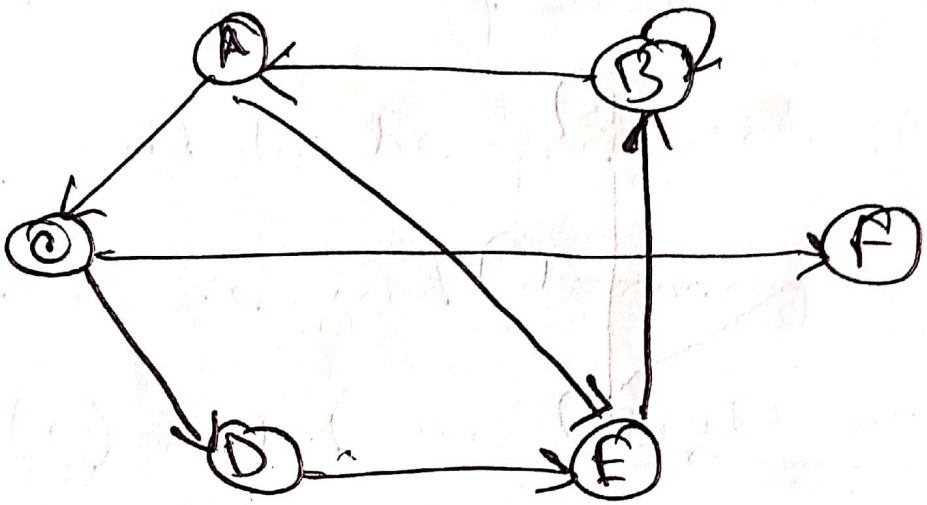
For undirected graph:

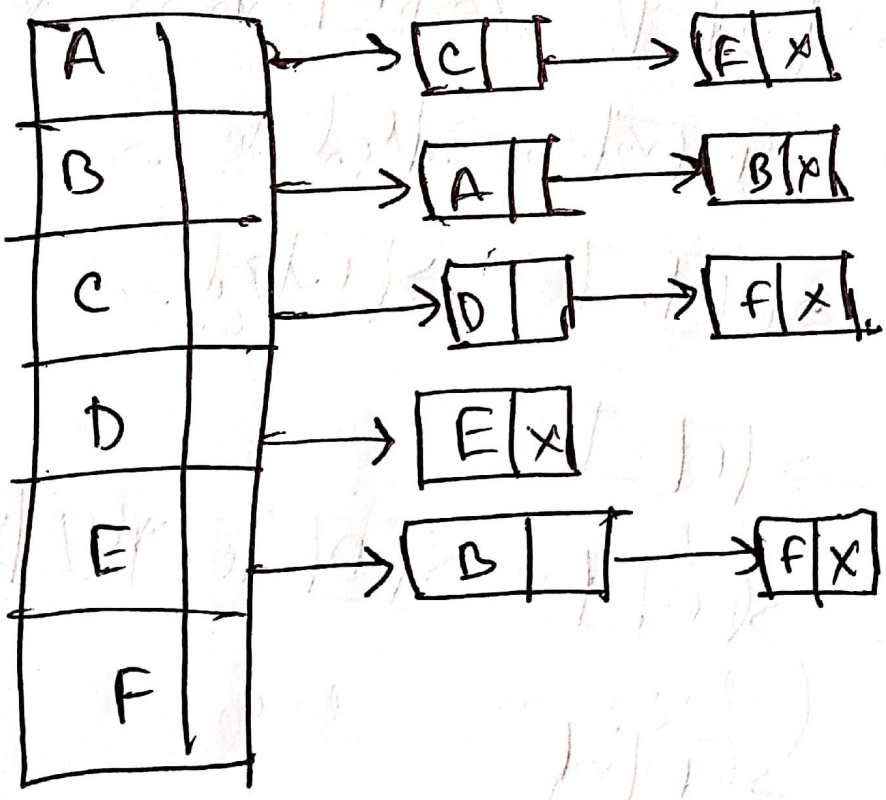




A

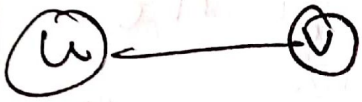
Directed Graph.





Adjacency (matrix / list): Most useful when information about the vertices is more desirable than information about the edges

Incidence (matrix): Most useful when information about edges is more desirable than information about vertices



~~Vert~~ u and v are adjacent if {u,v} is an edge, e is called incident with u and v. u and v are called end points of {u,v}

A Graph G is said to be labeled if its edges are assigned data. In particular, G is said to be weighted if each edge e in G is assigned a nonnegative numerical value w(e) called the weight or length of e.

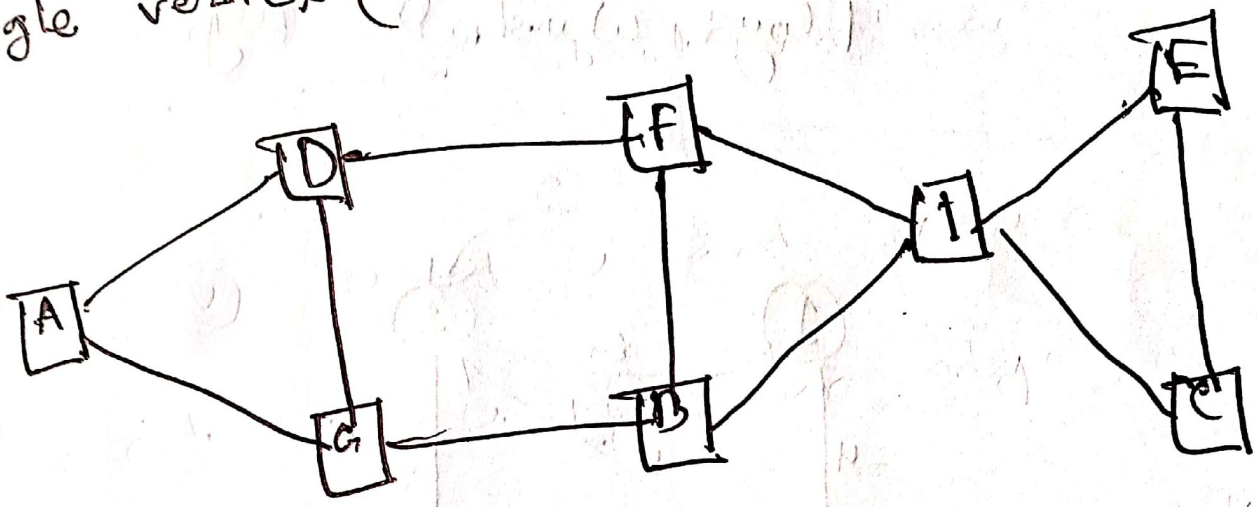
* each path P in G is assigned a weight or length which is the sum of the weights of the edges along the path P.

DAG (Directed Acyclic Graph)

Acycle (no cycle)

Bi-connected Graph:

A bi-connected graph is a connected graph which cannot be broken down into any further pieces by deletion of any single vertex (and incident edges).



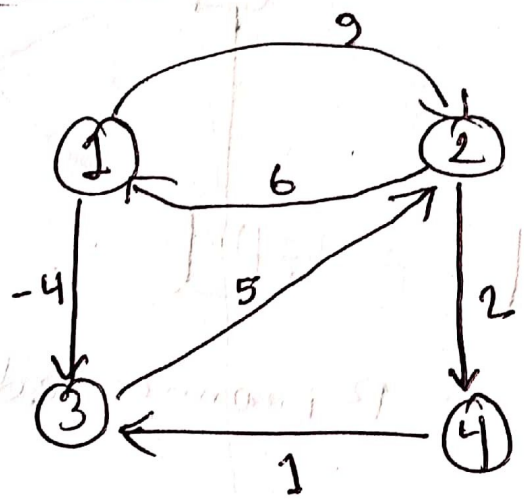
Biconnected graph.

There are two standard ways of maintaining a graph G in the memory of a computer. One way, called the sequential representation of G, is by means of its adjacency matrix A.

The other way, called the link representation of G, is by means of linked lists of neighbors.

All Pair shortest Path

Floyd Warshall's Algo



$$D^{\circ} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 9 & -4 & \infty \\ 6 & 0 & \infty & 2 \\ \infty & 5 & 0 & \infty \\ \infty & \infty & 1 & 0 \end{bmatrix} \end{array}$$

Path Matrix

Now: $D^1 = $

	1	2	3	4
1	0	9	-4	∞
2	6	0		
3	∞		0	
4	∞			0

1 এর জন্য D^0 তে থাকা 1 এর কলাম ও রো বরাবর যা আছে তা এখানে ও থাকবে O। তার পর ওইগুলি দিয়ে বাকিগুলো:

$$D^0[2,3] \qquad D^0[2,1] + D^0[1,3]$$

$$\infty \quad > \quad 6 + (-4)$$

$$= 2$$

তাহলে,

$D^1 = $

	1	2	3	4
1	0	9	-4	∞
2	6	0	2	2
3	∞	5	0	∞
4	∞	∞	1	0

এভাবে উপরের এখানেও হয়। তাও [illegible] [illegible]

$D^0[2,4] \; D^0[2,1] + D^0[1,4]$

$2 < 6 + \infty = \infty$

$D^0[3,2] \quad D^0[3,2] + D^0[1,2]$

$5 < \infty + 9 = \infty$

<u>Now:</u>

$D^2 =$

	1	2	3	4
1	0	9	-4	11
2	6	0	2	2
3	11	5	0	7
4	∞	∞	1	0

এইখানে
আমরা D1
থেকে 2
এর রো ও
কলাম
বসাবো
ও বাকিগুলি

$D^1[1,3] = D^1[1,2] + D^1[2,3]$

$-4 < 9 + 2 = 11$

$D^1[1,4] = D^1[1,2] + D^1[2,4]$

$\infty > 9 + 2 = 11$

Scanned with CamScanner

$D^1[3,1] = D^1[3,2] + D^1[2,1]$

$\infty > 5 + 6 = 11$

Now:

$$D^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & -4 & 3 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & 12 & 6 & 1 & 0 \end{array}$$

তাবত D^3
থেকে 3
এর কো
ফলায়
বগালো।
ও কোনা
কুমি

$D^2[1,2] = D^2[1,3] + D^2[1,2]$

$9 > -4 + 5 = 1$

$D^2[1,4] = D^2[1,3] + D^2[3,4]$

$11 > -4 + 7 = 3$

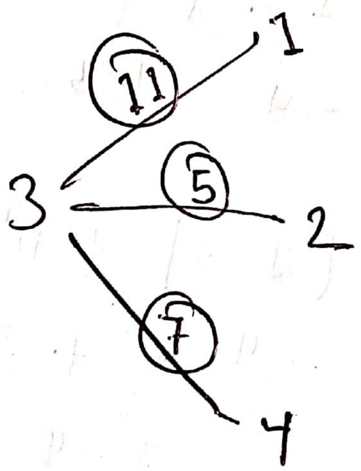
Now:

$$D^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & -4 & 3 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & 12 & 6 & 1 & 0 \end{array}$$

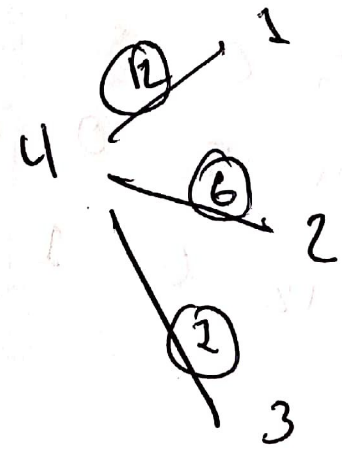
এবার আমরা লাস্ট স্টেপ থেকে পাই D^4 দিয়ে, শর্টেস্ট পাথ বের করবো।

$$D^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & -4 & 3 \\ 6 & 0 & 2 & 2 \\ 11 & 5 & 0 & 7 \\ 12 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

এখন 3 থেকে অন্যান্যগুলোর শর্টেস্ট পাথ



যদি 4 started vertex.



Actual formula

$$D^{k}[i,j] = \min\{D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j]\}$$

এখানেই হাতে সুযোগ নেই বলেছি।

previous problem এর জন্য লেখা pseudocode

```
for (k = 1 to 4){
    for (i = 1 to 4){
        for (j = 1 to 4){
            D^k[i,j] =
        }
    }
}
```

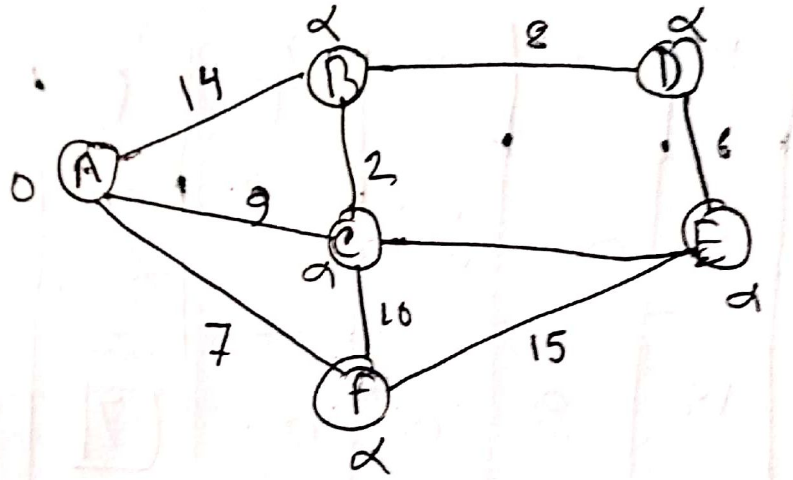
যদি -ve weight ও কাজেগুলোর output ঠিক

অবশ্যই হবে থাকলো।

Dijkstra's Algorithm

$\Rightarrow$ if $(d(u) + c(u,v) < d(v))$

$$d(v) = d(u) + c(u,v)$$



প্রথমে যে গ্রাফটি দেওয়া থাকবে তা আমরা নিবো যে-
ধরলাম সোর্সে 0 ও অন্যগুলোতে α ধরবো। সূত্র অনুযায়ী
ধরলাম A হলো u এবং B হলো v ~~নিয়েছি এবং u হবে~~

Distance of u $d(u) = 0$

cost of u to v $c(u,v) = 14$

সূত্র অনুযায়ী, $0 + 14 < \alpha$

$\therefore d(v) = 14$

তাহলে B এর ওখানে α এর বদলে এখন 14 হবে।

A থেকে B, A থেকে C, A থেকে F এর থেকে বের করতে হবে। তুলনা করে যা কম তখন A বিবেচিত হবে।

Visited vertex	A	B	C	D	E	F
A	**0**	α	α	α	α	α
F		14	9	α	α	**7**
C		14	**9**	α	22	
B		**11**		α	20	
D				**19**	20	
E					**20**	

যেটি কম সেটিতে visited node-এ যাবে

এখন A to E shortest path বের করতে হবে। ধরা যাক E তে পৌঁছাতে হবে। তারপর ব্যাকট্র্যাক করবো। যাকে উল্টো উঠতে থাকবো। যতক্ষণ না পরিবর্তন (E এর মান) আসবে। পরিবর্তন আসলে সেই row তে

যে সিরিজটি হবে, তখন সেটা হবে সবচেয়ে।
তাহলে সবগুলো A থেকে যাবে।

⇒ E, C, A

∴ A, C, E

9 + 11 = 20

A to D = 11

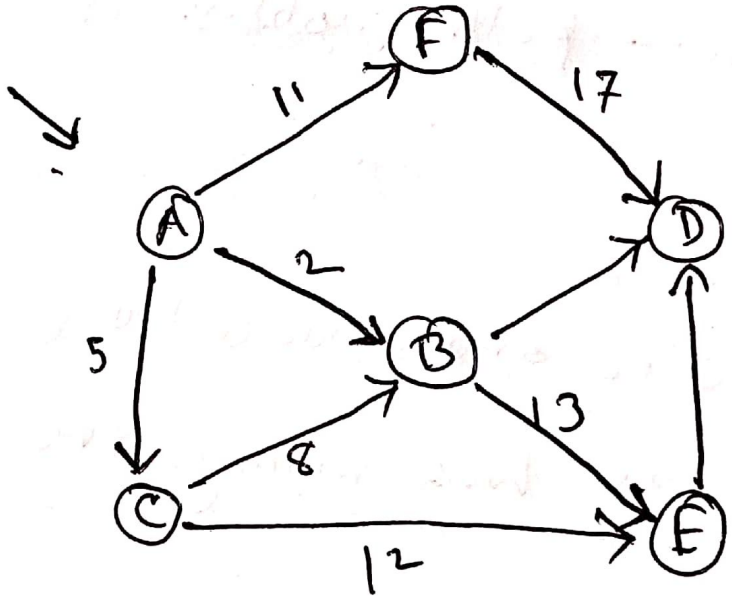
A to C = 9

A to D = 19

A to E = 20

A to F = 7

Directed Graph



Visited Node	A	B	C	D	E	F
A	[0]	∞	∞	∞	∞	∞
B		[2]	5	∞	∞	11
C			[5]	7	15	11
D				[7]	15	11
F					15	[11]
E					[15]	

Dijkstra's algorithm: is a solution to the single-source shortest path problem in graph theory

works on both directed and undirected graphs. However, all edges must have nonnegative weight

single-source shortest path problem - the problem of finding shortest paths from a source vertex to all other vertices in the graph.

shortest path: a path of the minimum weight

- Application:
  - static / dynamic network routing
  - robot motion planning
  - map / route generation in traffic
- Negative edge will be considered as minimum cost always
- shortest-paths can have no cycles
  - Any shortest-path in graph G can be no longer then n-1 edges, where n is the number of vertices.

Depth First Search (DFS) and Breadth First Search (BFS) Algorithms

- DFS and BFS are common methods of graph traversal, which is the process of visiting every vertex of a graph.
- Stacks and queues are two additional concepts used in the DFS and BFS algo-

Stack:

- A stack is a type of data storage in which only the last element added to the stack can be retrived.
- It is like a steck of plates only the top plate can be taken from the steck.

- The three stacks operations are:

- push - put an element on the stack
- peek - look at the top element on the stack, but do not remove it.
- pop - take the top element off the stack

Queue:

- A queue is a type of data storage in which the elements are accessed in the order they were ~~deleted~~ added.
- It is like a cafeteria line where the person at the front of the line is next
- Two queues operation are:
  - Enqueue - add an element to the end of the queue
  - Dequeue - remove an element from the start of the queue

Depth First Search (DFS) Algorithm:

Mark the starting node of the graph as visited and push it onto the stack

while the stack is not empty

peek at top node on the stack

if there is an unvisited child of that node

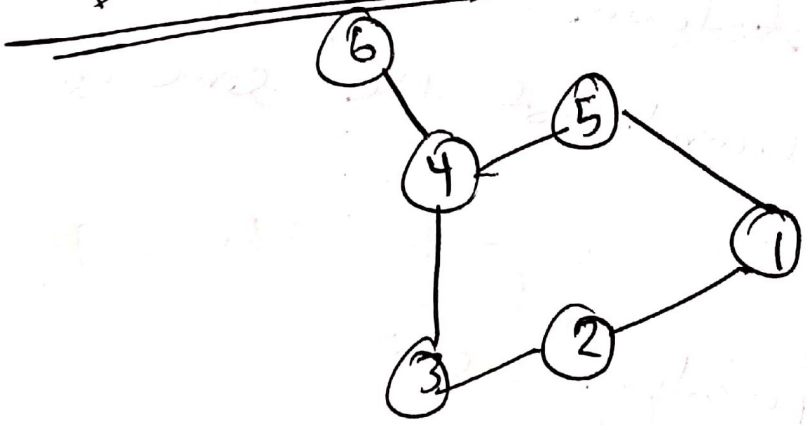
mark the child as visited and

push the child node onto the stack

Else

pop the top node off the stack.

DFS Example:



Action	stack	Unvisited nodes	visited node
Start with node 1	1	2, 3, 4, 5, 6	1
Peek at the stack Node 1 has unvisited child nodes 2 & 5	1	2, 3, 4, 5, 6	1
Mark node 2 visited	1, 2	3, 4, 5, 6	1, 2
Peek at the stack Node 2 has unvisited child 3 & 5	1, 2	3, 4, 5, 6	1, 2
Mark node 3 visited	1, 2, 3	4, 5, 6	1, 2, 3
Peek at the stack Node 3 has unvisited child node 4	1, 2, 3	4, 5, 6	1, 2, 3
Mark node 4 visited	1, 2, 3, 4	5, 6	1, 2, 3, 4
peek at the stack Node 4 has unvisited child node 5	1, 2, 3, 4	5, 6	1, 2, 3, 4
Mark node 5 visited	1, 2, 3, 4, 5	6	1, 2, 3, 4, 5
peek at the stack Node 5 has no unvisited children	1, 2, 3, 4, 5	6	1, 2, 3, 4, 5
pop node 5 off stack	1, 2, 3, 4	6	1, 2, 3, 4, 5
Peek at the stack Node 4 has unvisited child node 6	1, 2, 3, 4	6	1, 2, 3, 4, 5
Mark node 6 visited	1, 2, 3, 4, 6		1, 2, 3, 4, 5, 6

Breadth First Search (BFS) Algorithm

Mark the starting node of the graph as visited and enqueue it into the queue

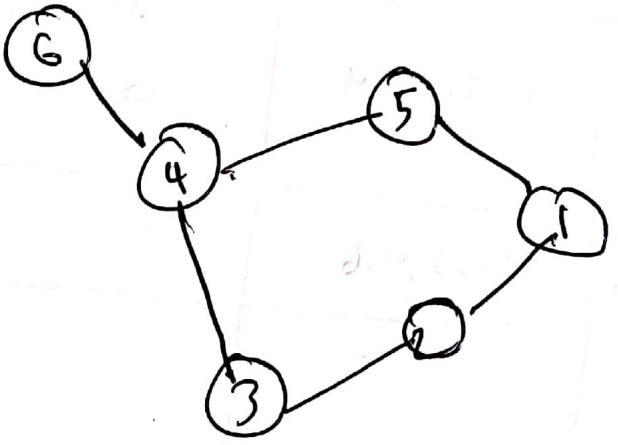
while the queue is not empty

Dequeue the next node from the queue to become the current node

while there is an unvisited child of the current node

Mark the child as visited and enqueue the child node into the queue

Example:



Action	current Node	queue	unvisited Node	visited node
start with node 1		1	2, 3, 4, 5, 6	1
Dequeue Node 1	1		2, 3, 4, 5, 6	1
Node 1 has unvisited child node 2 & 5	1		2, 3, 4, 5, 6	1
Mark 2 as visited and enqueue into queue	1	2	3, 4, 5, 6	1, 2
Mark 5 as visited node enqueue into queue	1	5, 2	3, 4, 6	1, 2, 5
Node 1 has no more unvisited child, dequeue a new current node 2	2	5	3, 4, 6	1, 2, 5
Mark 3 as visited and enqueue into queue	2	3, 5	4, 6	1, 2, 5, 3
Node 2 has no more unvisited child, dequeue a new current node 5	5	3	4, 6	1, 2, 5, 3
Mark 4 as visited and enqueue into queue	5	4, 3	6	1, 2, 5, 3, 4
Node 5 has no more unvisited child, dequeue a new current node 3	3	4	6	1, 2, 5, 3, 4
Node 3 has no more unvisited children, dequeue a new current node 4	4		6	1, 2, 5, 3, 4
Mark 6 as visited and enqueue into queue	4	6		1, 2, 5, 3, 4, 6

Topological Sort

⇒ (i) Graph directed হবে।

(ii) Graph এ কোনো cycle থাকবে না।

• Topological sort of a directed graph is a linear ordering of its vertices such that every directed edge uv from vertex u to v, where u comes before v

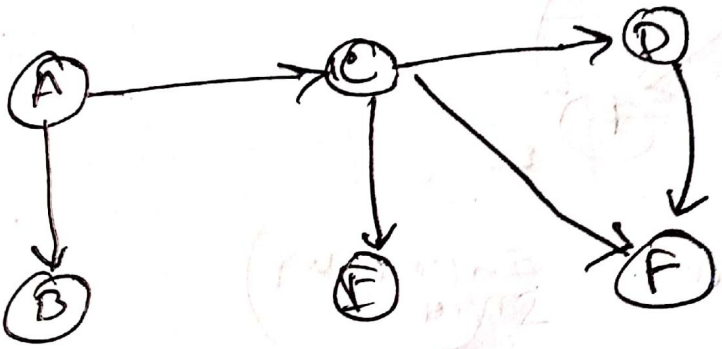
---

Method:

(i) প্রথমে vertices এর indegree (no of incoming edge) বের করতে হবে।

(ii) যে vertices পাওয়া যাবে যার indegree = 0

Example:



indegree এর মান:

A এর 0

B এর 1

C এর 1

D এর 1

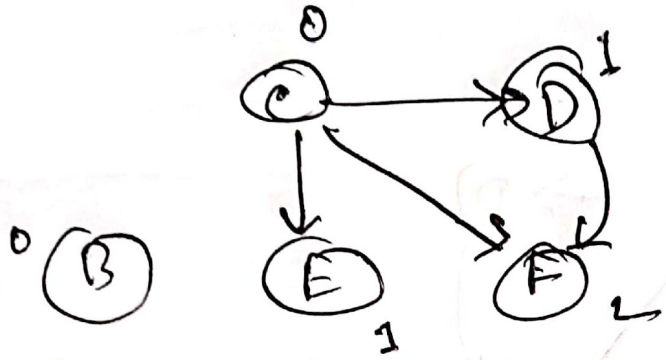
E এর 1

F এর 2

এখানে আমরা যার indegree 0 সেই প্রথম যে যাবে সে প্রিন্ট করবে।

Print: A B C D E F

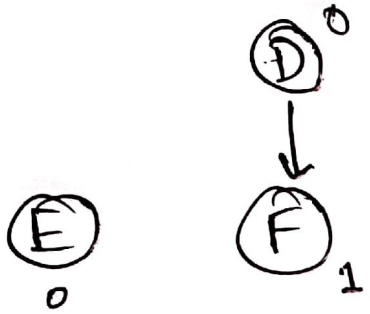
এভাবে পর পর A node graph থেকে বাদ যাবে এবং edge ও বাদ যাবে মনে, যার ফলে আবার নতুন করে যার indegree 0 সে প্রিন্ট হবে।



(এখানে indegree বের করে লেখা আছে)

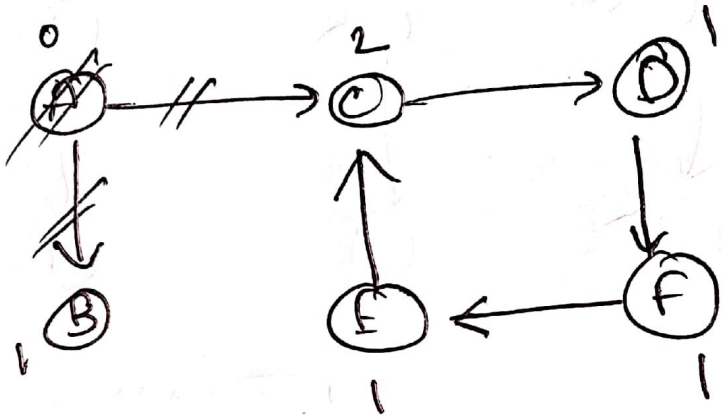
Next:

এখানে যেহেতু B ও E লেখার পর আর কোন edge নেই

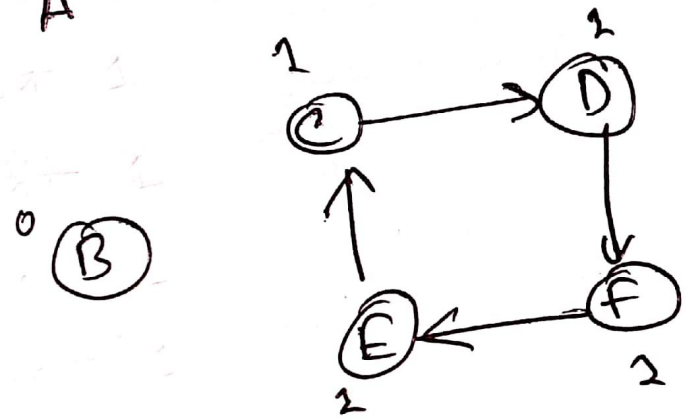


Next:

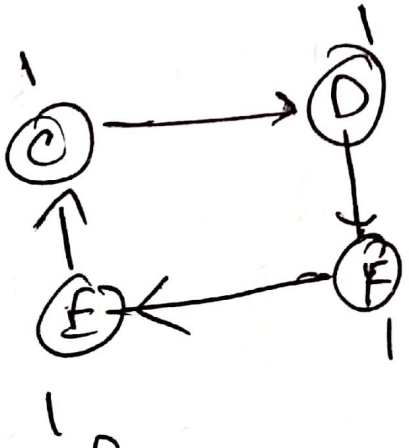
Cycle হলে যেভাবে আসে:



Print: A



print: A B



যেখানে কোনটির indegree 0 হয় সেটা cycle হলে হবে না।