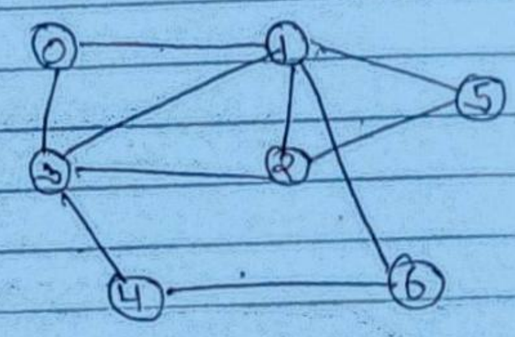
Graph

Date: / / Page no:

Representation of Graph

using matrix

	0	1	2	3	4	5	6
0	0	1	0	1	0	0	0
1	1	0	1	1	0	1	1
2	1	0	0	1	0	1	0
3	1	1	1	0	1	0	0
4	0	0	0	1	0	0	1
5	0	1	1	0	0	0	0
6	0	1	0	0	1	0	0

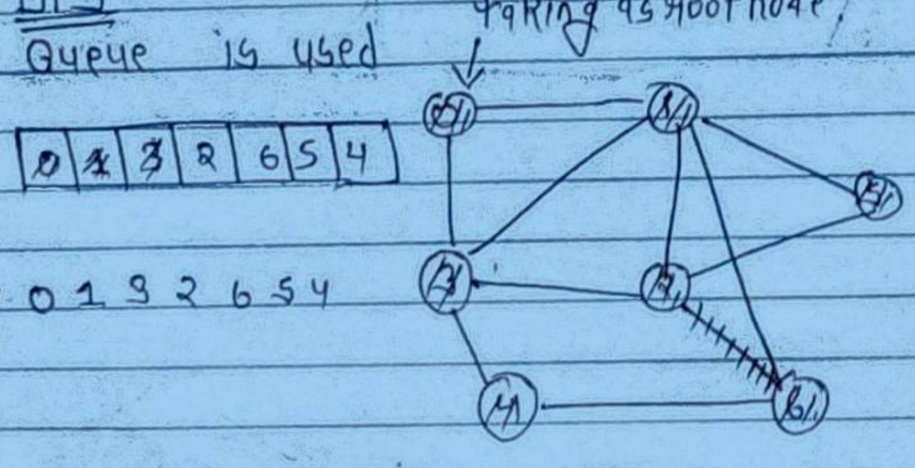


Graph Traversal

BFS

Queue is used

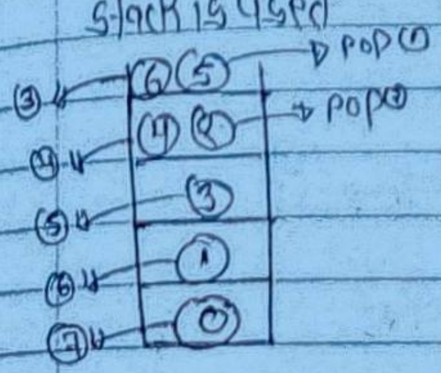
taking as root node

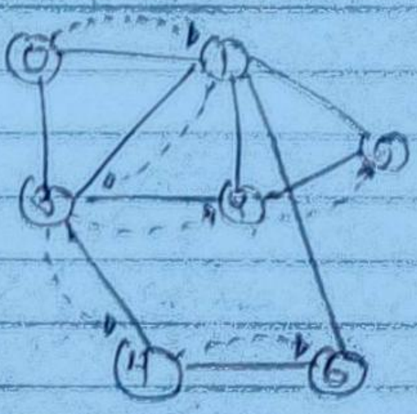


BFS → 0 1 3 2 6 5 4

DFS

stack is used





DFS → 0 1 3 2 5 4 6

Types of edges in DFS :

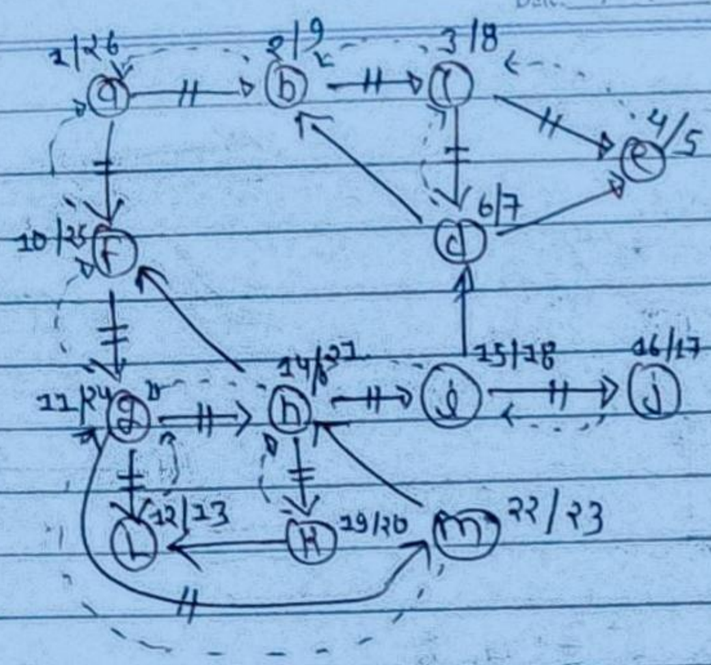
① tree → members of DFS

② forward → e(x,y) where y appears after x and there there is a path from x to y

imp

③ Backward → e(x,y) where y appears befor x and there is a a path from y to x

④ cross → e(x,y) where is no path from y to x.



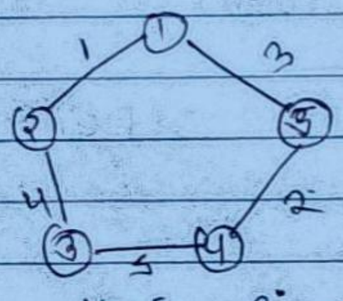
tree edge → (a,b) (b,c) etc (——→)

forward edg → (a,m)

Backward → (d,b)

cross → (d,e)

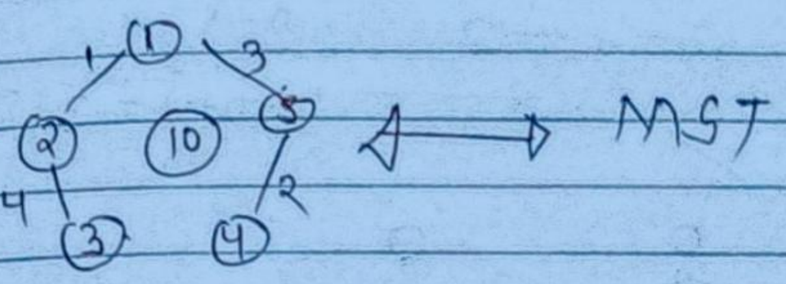
Min spanning tree.



$G(V,E)$
$G'(V',E')$
$V' = V$
$E' \subseteq E$
$E' = |V|-1$

$V=5$ $E=5$
$V'=5$ $E'=|V|-1=4$

MST has min sum of weight of edges min.



① Removing one edge from the ST will make it disconnected

② Adding one edge to ST will create a loop

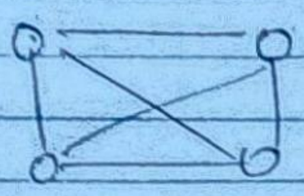
③ if each egde having distinct weight then there will be onlyone MST

④ Complete indirected graph can have n^{n-2} ST
n → no. of vertices.

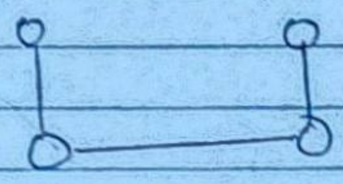
⑤ Every connected and undirected graph has atlest one ST

⑥ Disconnect grap does not have an ST

⑦ From complete graph by removing max $(e-n+1)$ edges, we can construct a ST



e = 6
n = 4
(e-n+1) = 3

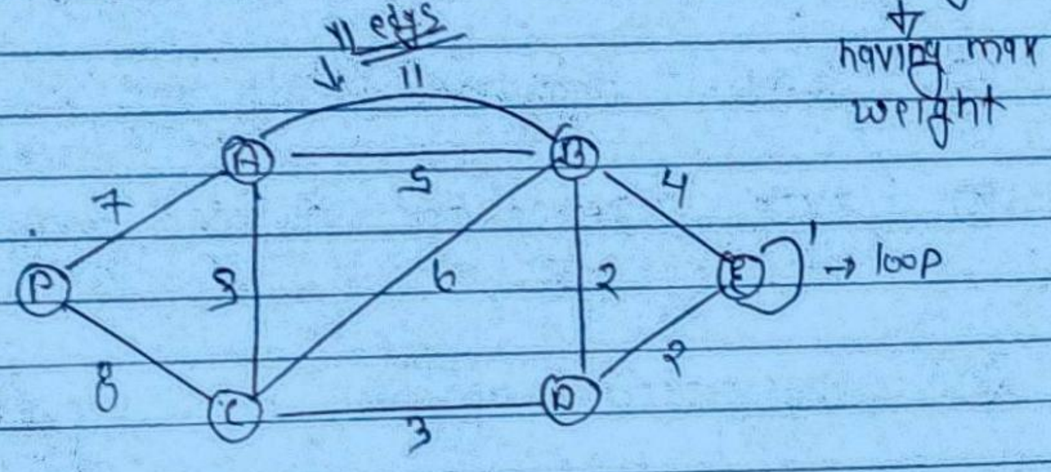


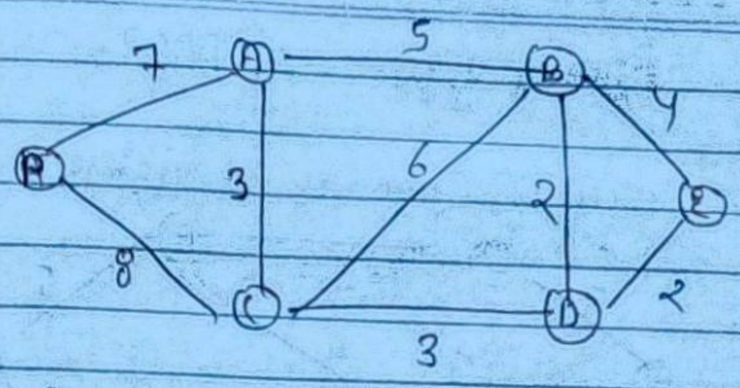
→ ST

Total no. of ST = $n^{n-2} = 4^2 = 16$

Kruskals Algo. of MST

step 1 → Remove all the loops and || edges ↓ having max weight



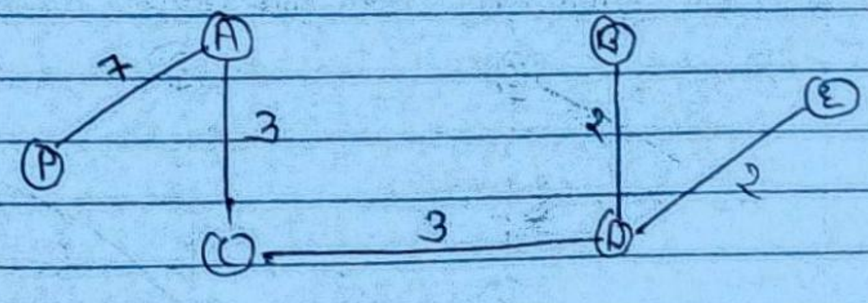


step 2 → Arrang all the edge in increasing order of edge weight

2 → BD, DE
3 → CD, AC
4 → BE
5 → AB
6 → BC
7 → DA
8 → DC

step 3 → choose edge having min weight

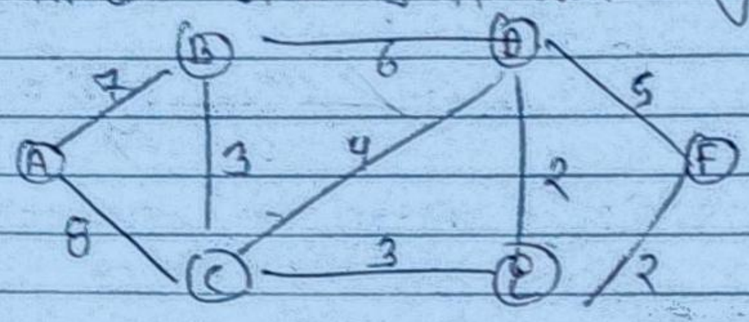
MST →



Note ⇒ we can not conned edge which form circle.

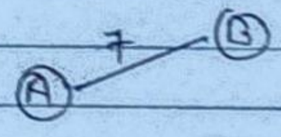
Prim's algo of MST

Step ① → Same as Kruskal's algo

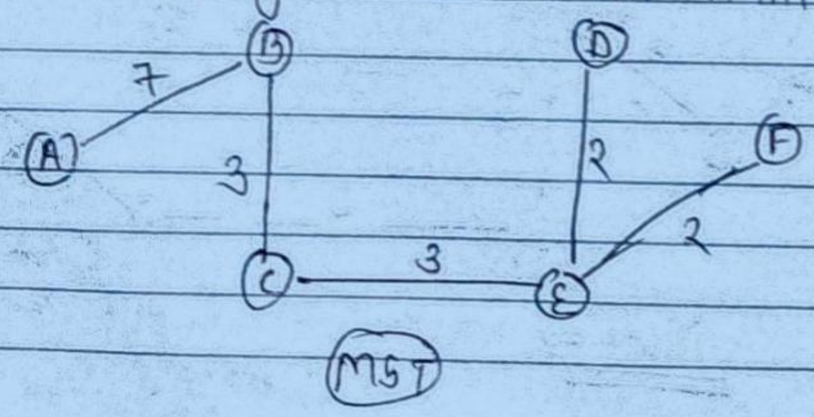


Step ② → choose any root node one as

Step ③ → check All edges connected to root and select which having min. weight



Step ④ check All edges connected to (A) and (B) and select which having min weight and not form circuit

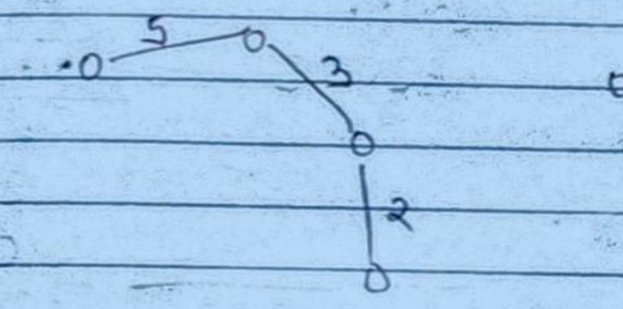


Q → x is cost of MST $G(V,E)$ & y is cost of MST $G(V,E^2)$ where E^2 means →

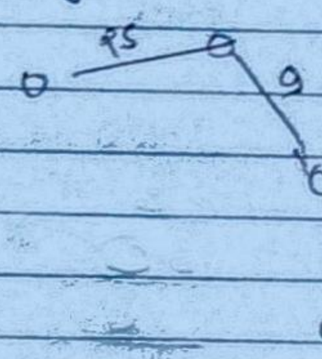
if $w(u,v) \in E$ then $w^2(u,v) \in E^2$

Find relationship b/w x & y $(w \in R)$

i) $x > y$ ii) $x < y$ iii) $x = y$ iv) None.

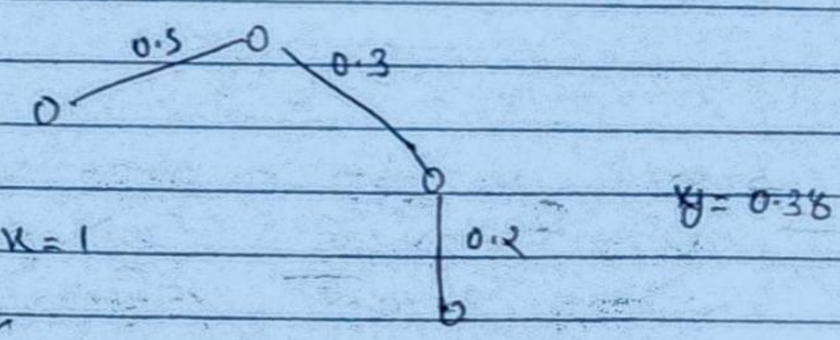


$x = 10$



$y = 38$

(ii) ✓



$x = 1$

$y = 0.36$

(i) ✓

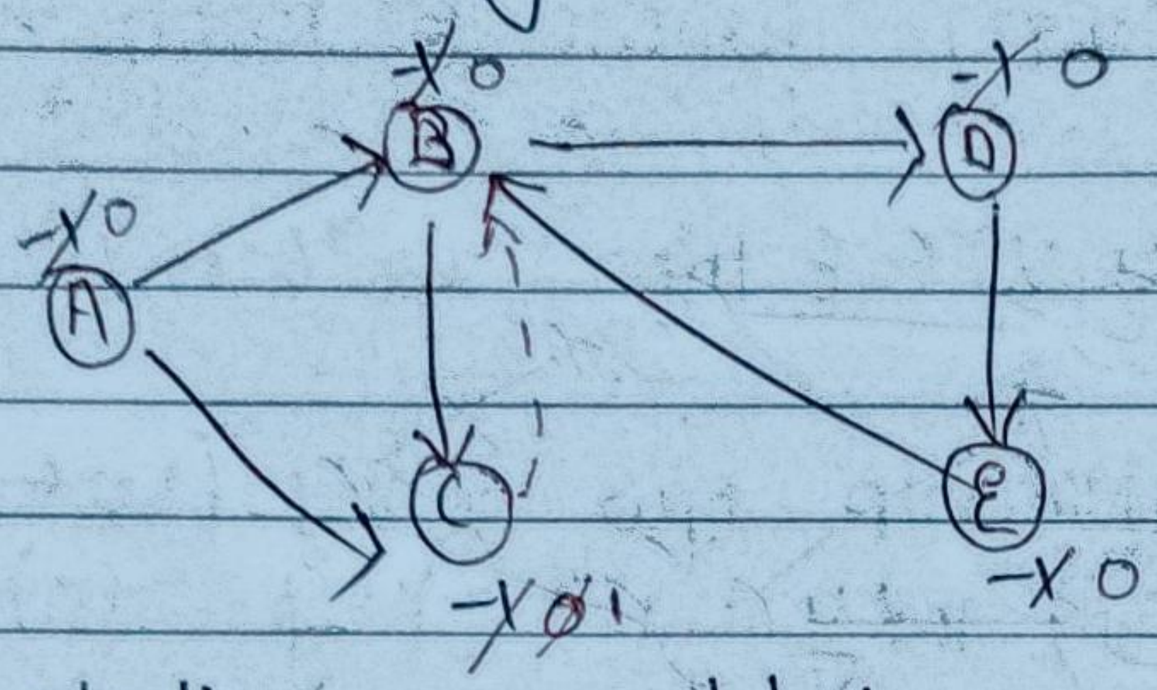
(iii) ✓

if all 1, 1, 1

Date: / / Page no:

Detect cycle in Directed graph.

Flag



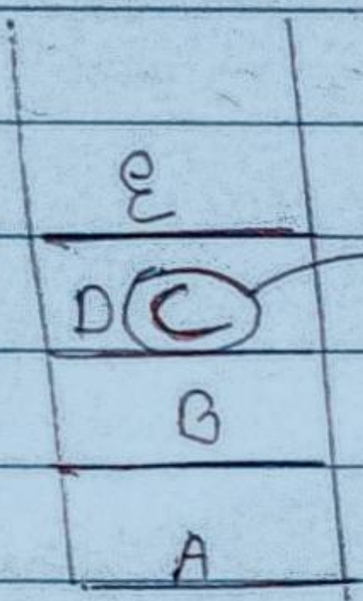
-1 : unvisited

0 : visited & in stack

1 : visited & popped out

DFS

stack



visited set

A B C D E

B → D → E → B

cycle

Parent Map

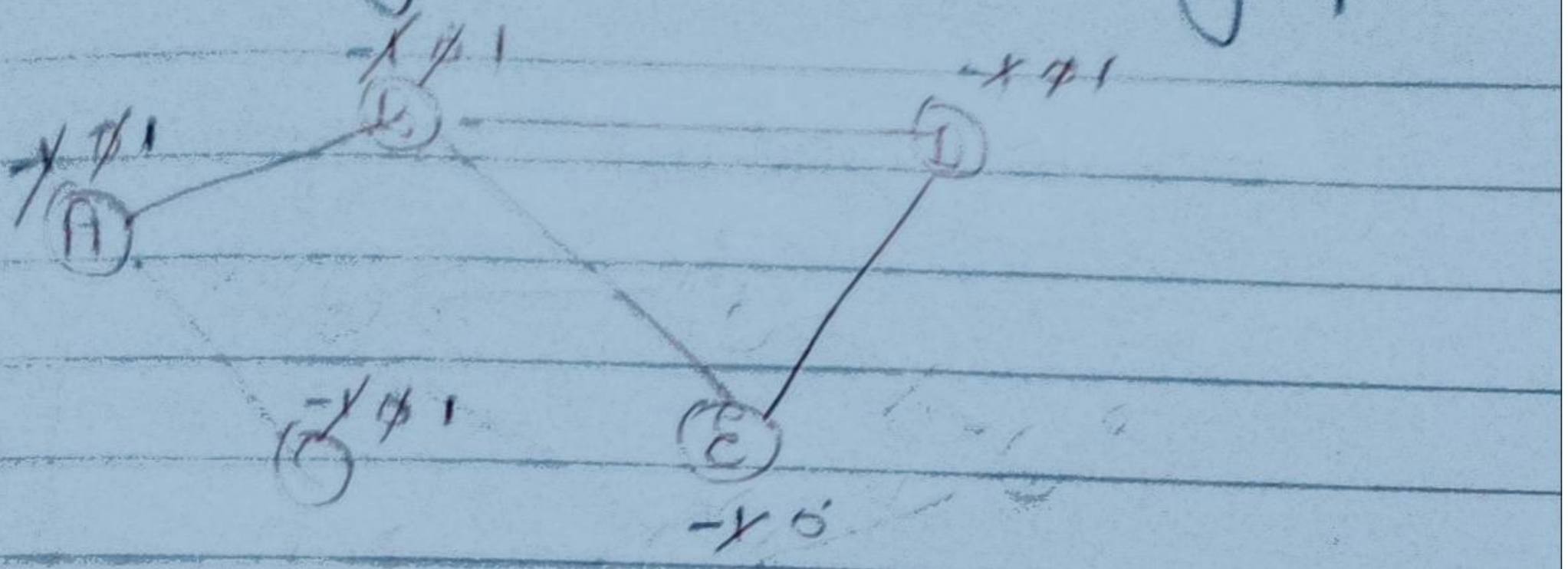
Vertex	parent
A	—
B	A
C	B
D	B
E	D

Note → when you get adjecent vetex of current vertex having flag 0 means graph contain cycle.

eg → (E to B)

Time complexcity O(V+E)

Detect cycle in Undirected graph



Queue

A	B	C	D	E	

visited set

A B C D

```
if (F[v] == 0)
    return 1
if (F[v] == -1)
    return
    insert
```

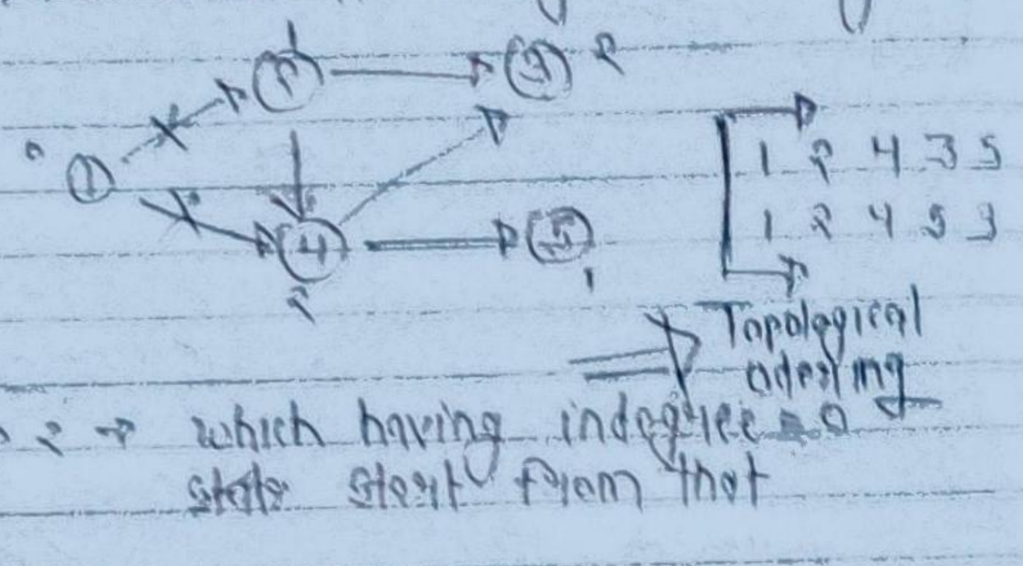
- if any vertex find its adjacent vertex with flag 0, then contain cycle

Topological Sort

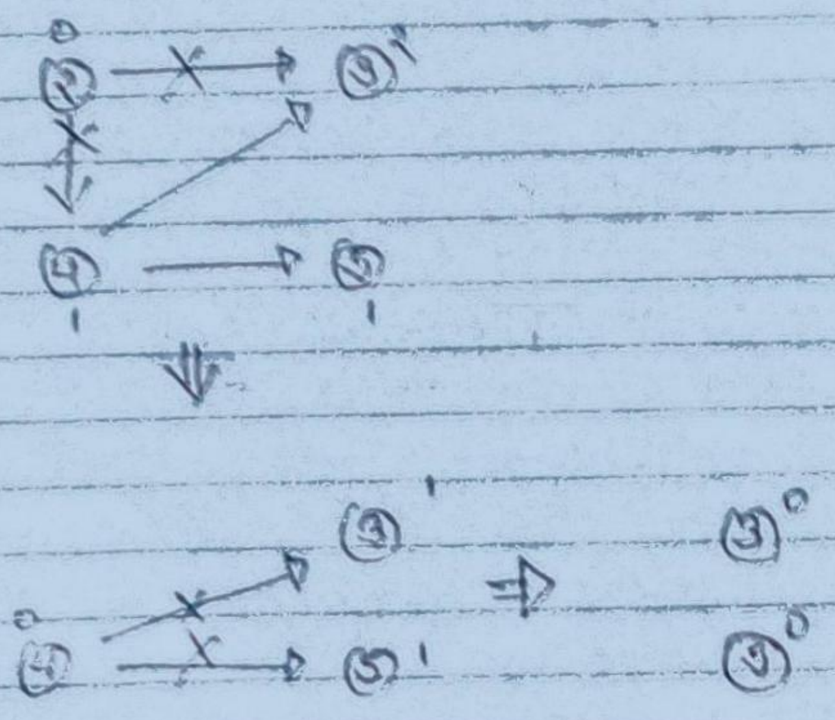
It is linear ordering of its vertices, such that for every directed edge uv For vertex u to v, u comes before vertex v in the ordering graph should be DAG (directed Acyclic graph) Every DAG will have atleast one topological ordering

u → v

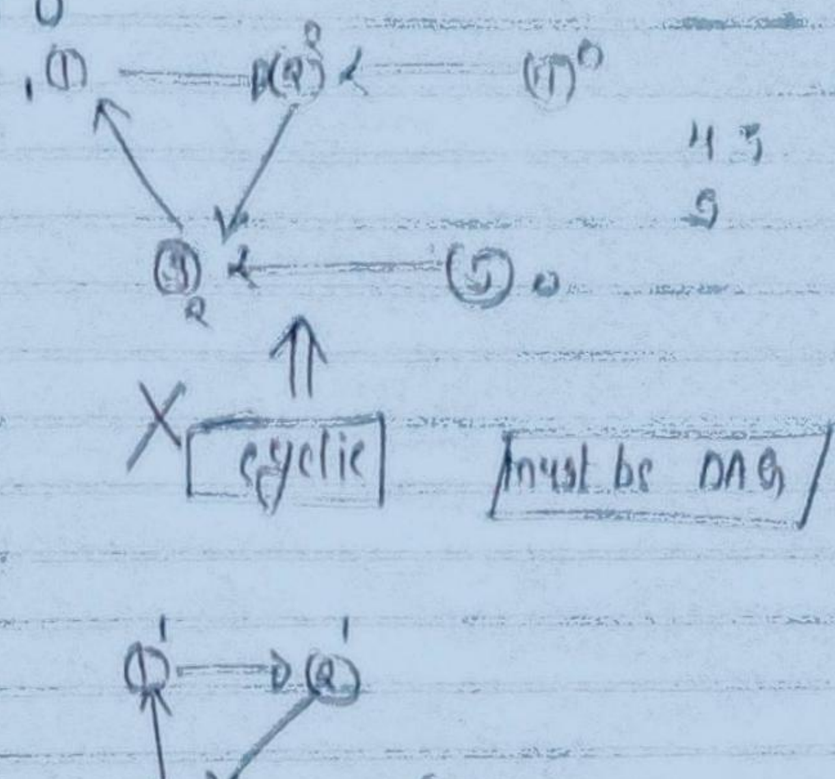
Step 1 → Find out indegree of every vertex

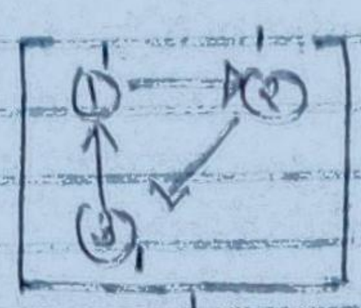


Step 2 → which having indegree = 0 state step from that

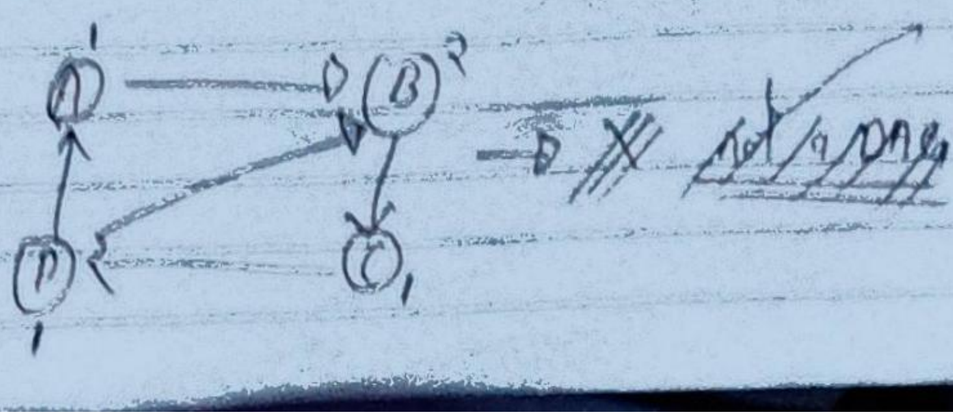


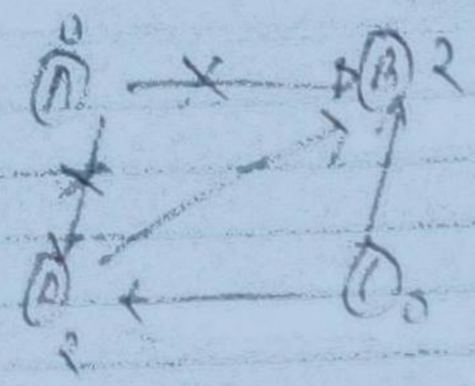
Q. Find T O



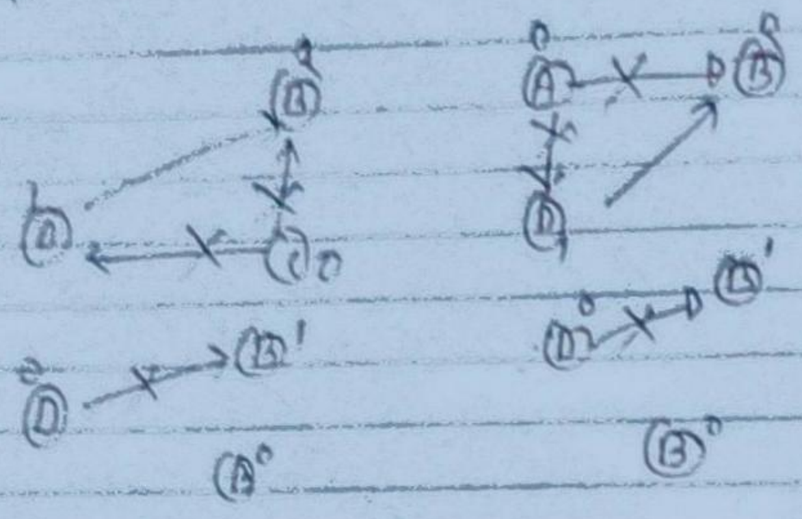


because must be DAG

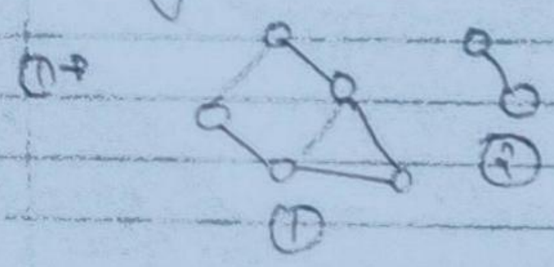


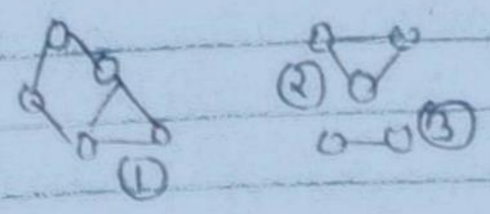


A C D B
C A D B



⇒ Connected components
* subgraph in which every any two vertex connected by path. And no vertex is connect with any other vertex. in supper graph.





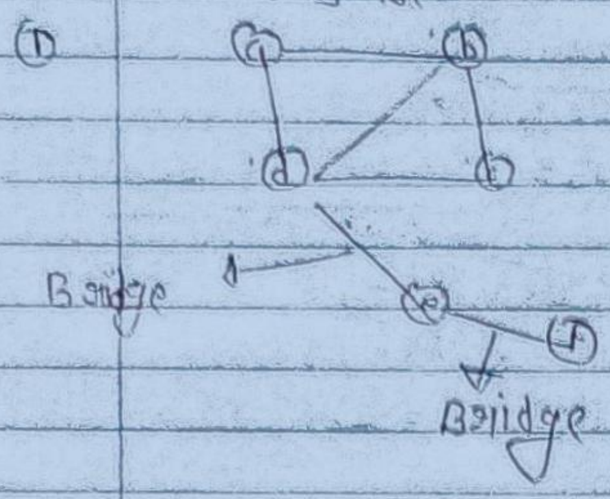
```
Connected_components (G) {

    for each vertex V ∈ N
        Flag [V] = -1
    count=0
    for (int V=0, V<N, V++)
        {
            if ( Flag [V] == -1)
            {
                DFS (V, Flag)
                count++;   cout << end;
            }
        }
    cout << count << endl;
}

DFS (int V, int flag ) {  flag[V] = 1, cout<<V;

    for each adjacent node u of V
        if (Flag [u] == -1)
            DFS (u, flag)
}
```

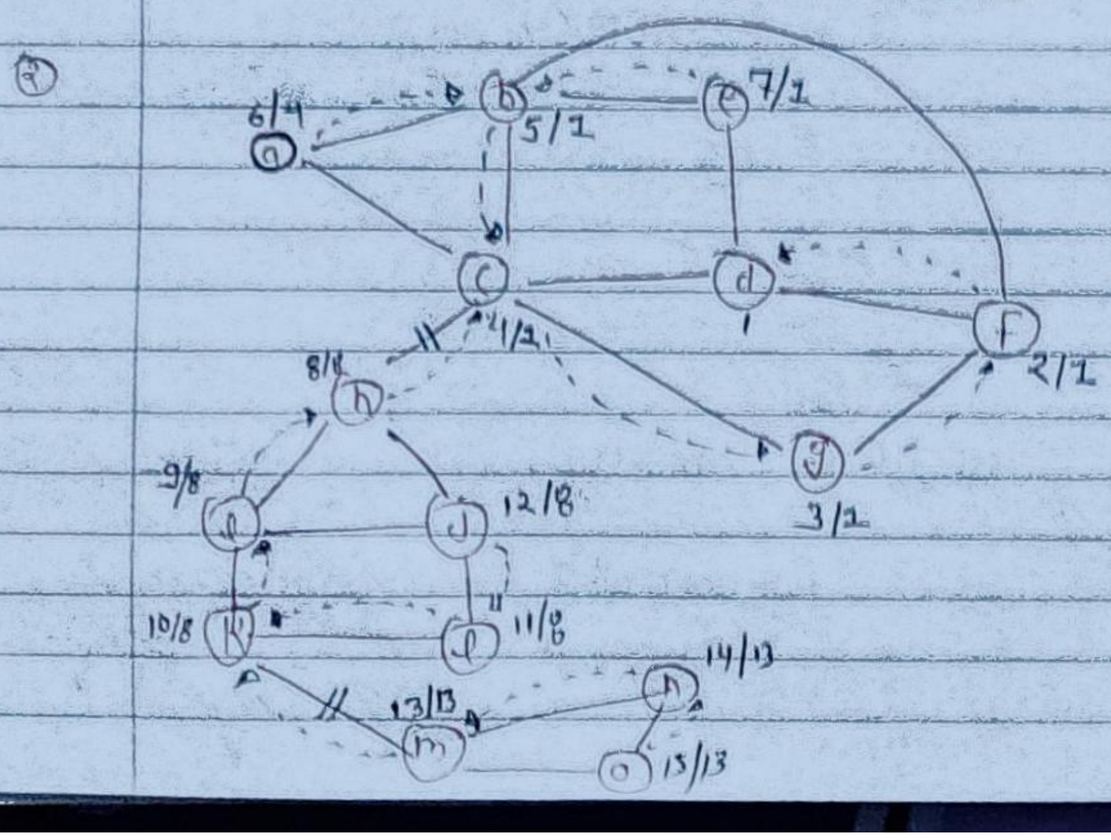
Date / / Page no

Finding all the Bridges of a graph.

→ is an edges whose removal res no. of connected components.

* DFS is use



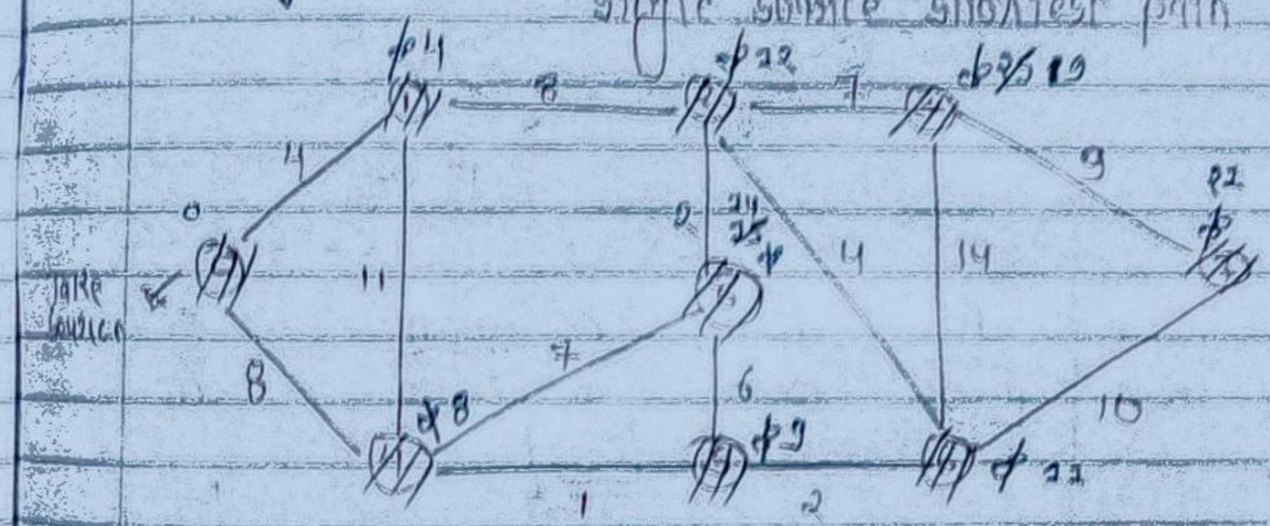


→ Time which a particular vertex can take to form and that min of all and so that we can check if that time is greater than or equal to its parents visit time then bridge is exists.

Time complexity = O(V+E)

Dijkstra Algo.

Single source shortest path



Step 1 → calculate distance of all vertex from source vertex that is 0 for source and infinity for all others.

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

update $d[v] = d(u) + c(u,v)$

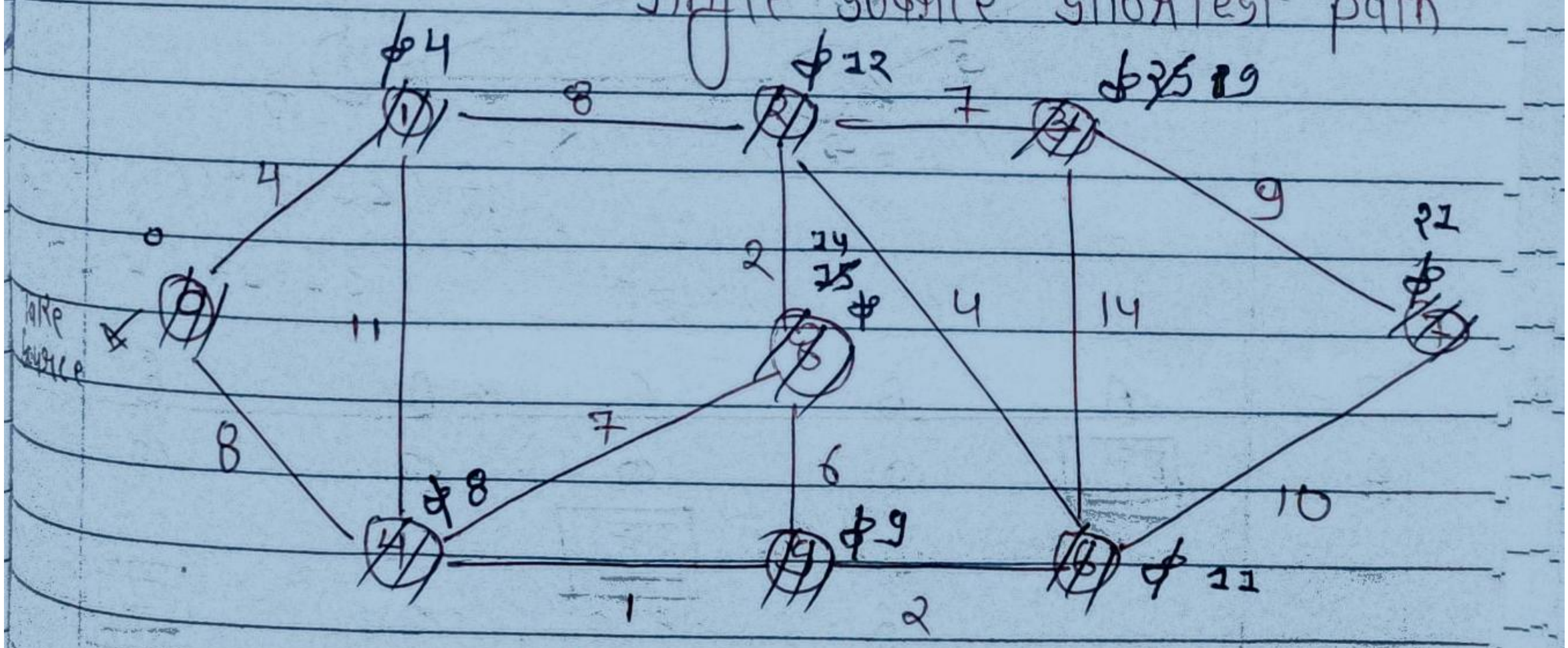
Date: / / Page no:

⇒ Time which a particular vertex can take from nbd. that min of all nbd. so that we can check if that time is greater than or equal to its parents start time then bridge is exists.

Time complexity $\Theta(V+E)$

Dijkstra Algo.

Single source shortest path



Step 1 → calculate distance of all vertex from source vertex that is 0 for source and infinity for all others

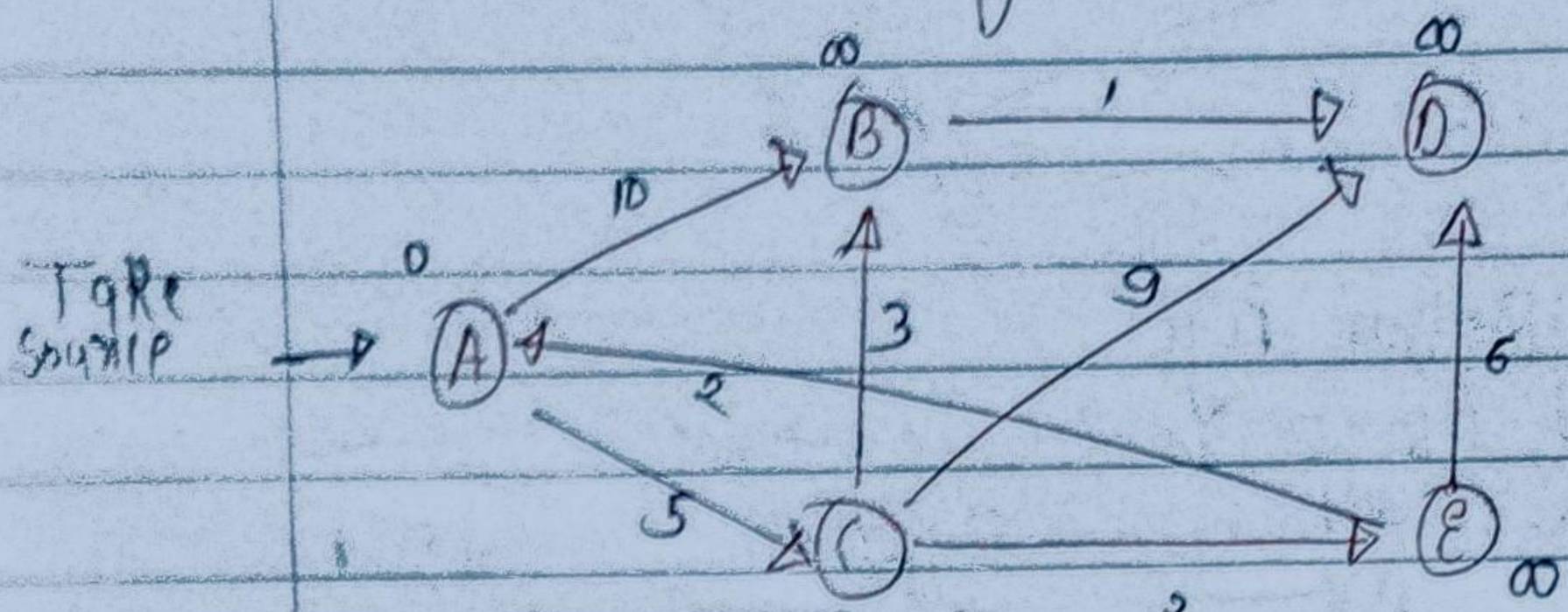
if $d(u) + c((u,v)) < d(v)$

update $d[v] = d(u) + c((u,v))$

Date: / / Page no:

Step 2 → ~~Next~~ select vertex having min dist^n

Note → Dijkstra Algo work on both directed and undirected graph.



Take source

if (d[u] + c(u,v) < d[v])

d[v] = d[u] + c(u,v)

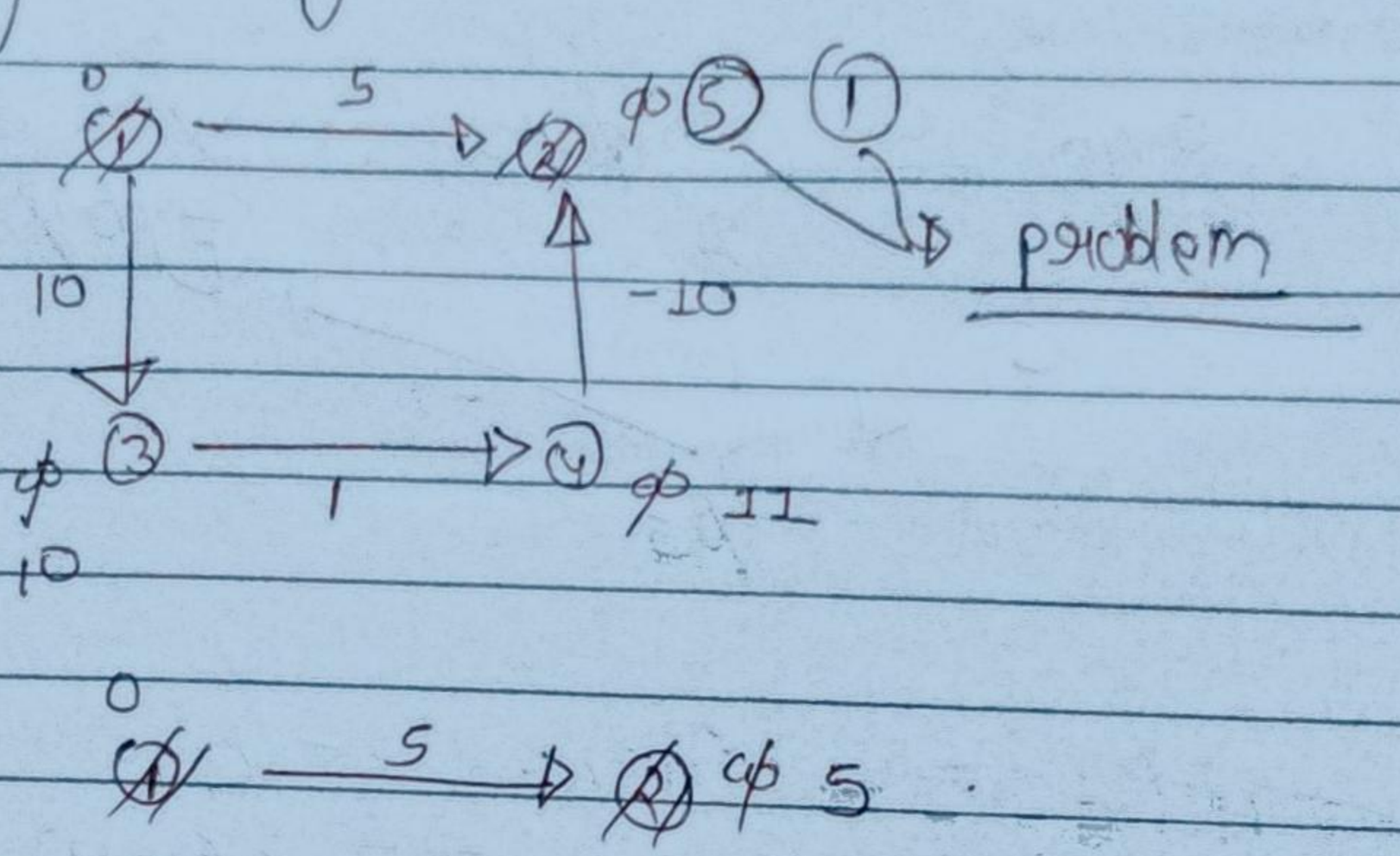
	A	B	C	D	E
A	[0]	∞	∞	∞	∞
		10	[5]		
		8		14	[7]
		[8]		13	
				[9]	

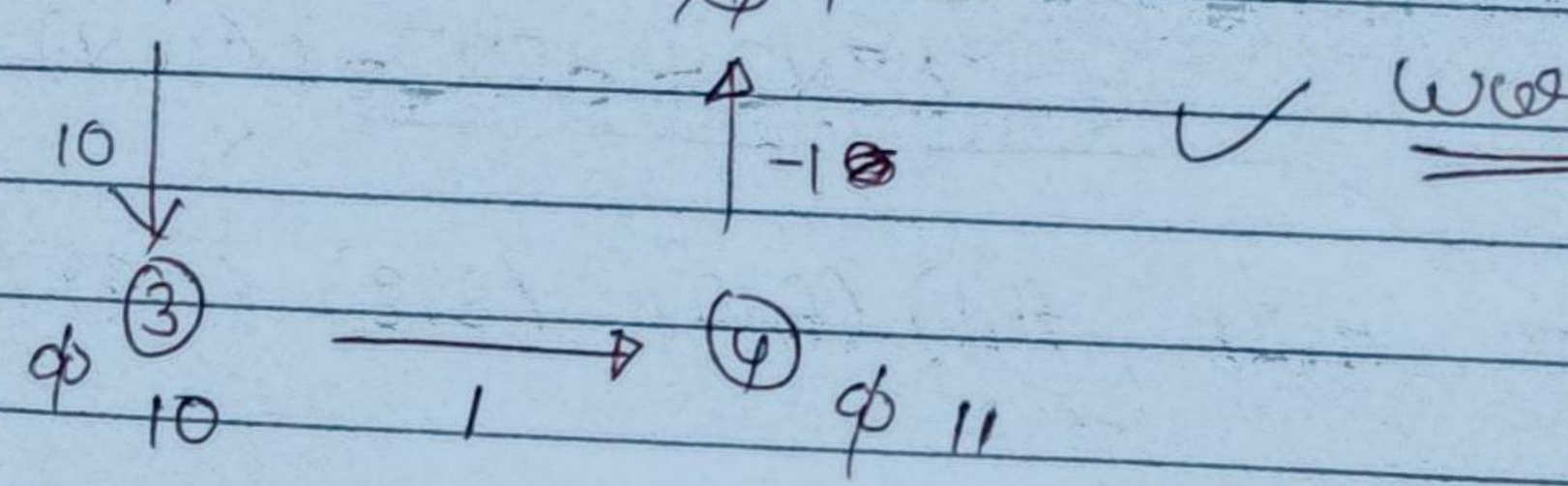
path

(A → D) ⟹ D B C A

(A → B) ⟹ B C A

Note → Dijkstra algo may or may not work when edge weight is -ve.





Bellman - Ford Algo.

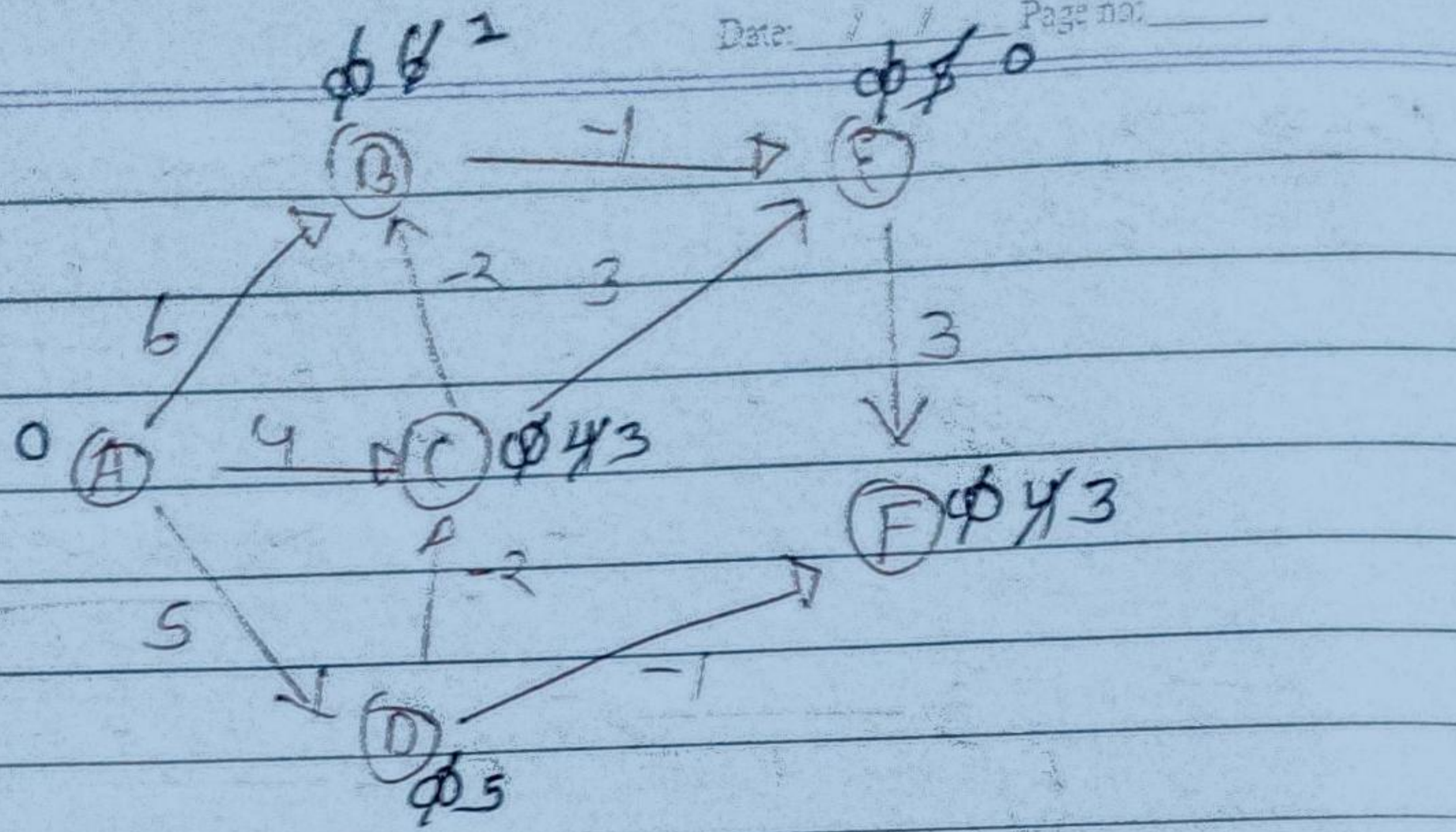
Single source shortest path

⇒ here is no problem when edge weight is -ve

It is slower than Dijkstra.

go on relaxing all the edges (n-1) times

n = no. of vertices



if (d[u] + ((u,v) < d[v])

d[v] = d[u] + ((u,v)

edges → (A,B), (A,C) (A,D) (B,E) (C,E) (D,E) (D,F)

Ist → ✓

IInd → ✓

IIIrd → ✓ * There is no change then over

IVth →

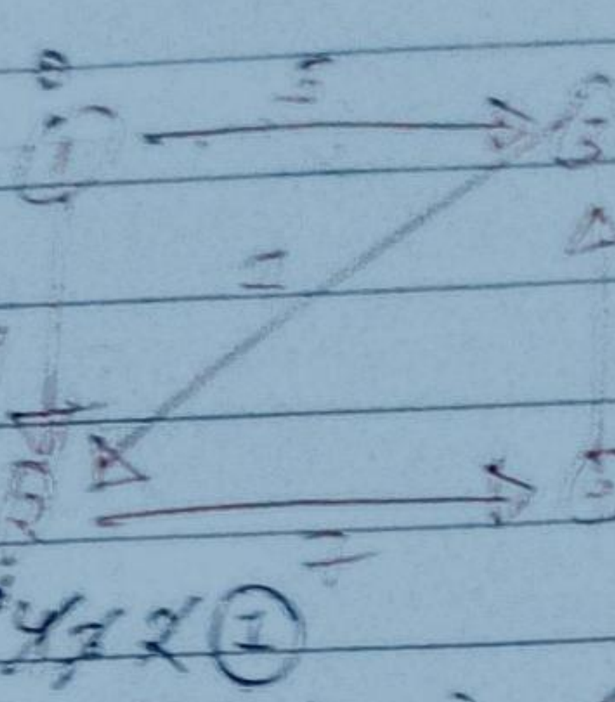
(n-1) times

5

Time complexity $O(E(|V|-1))$

if complete Graph $\theta\left(\frac{n(n-1)^2}{2}\right) \cong n^3$

if there is cycle and whose total sum is -ve then Bellman not work

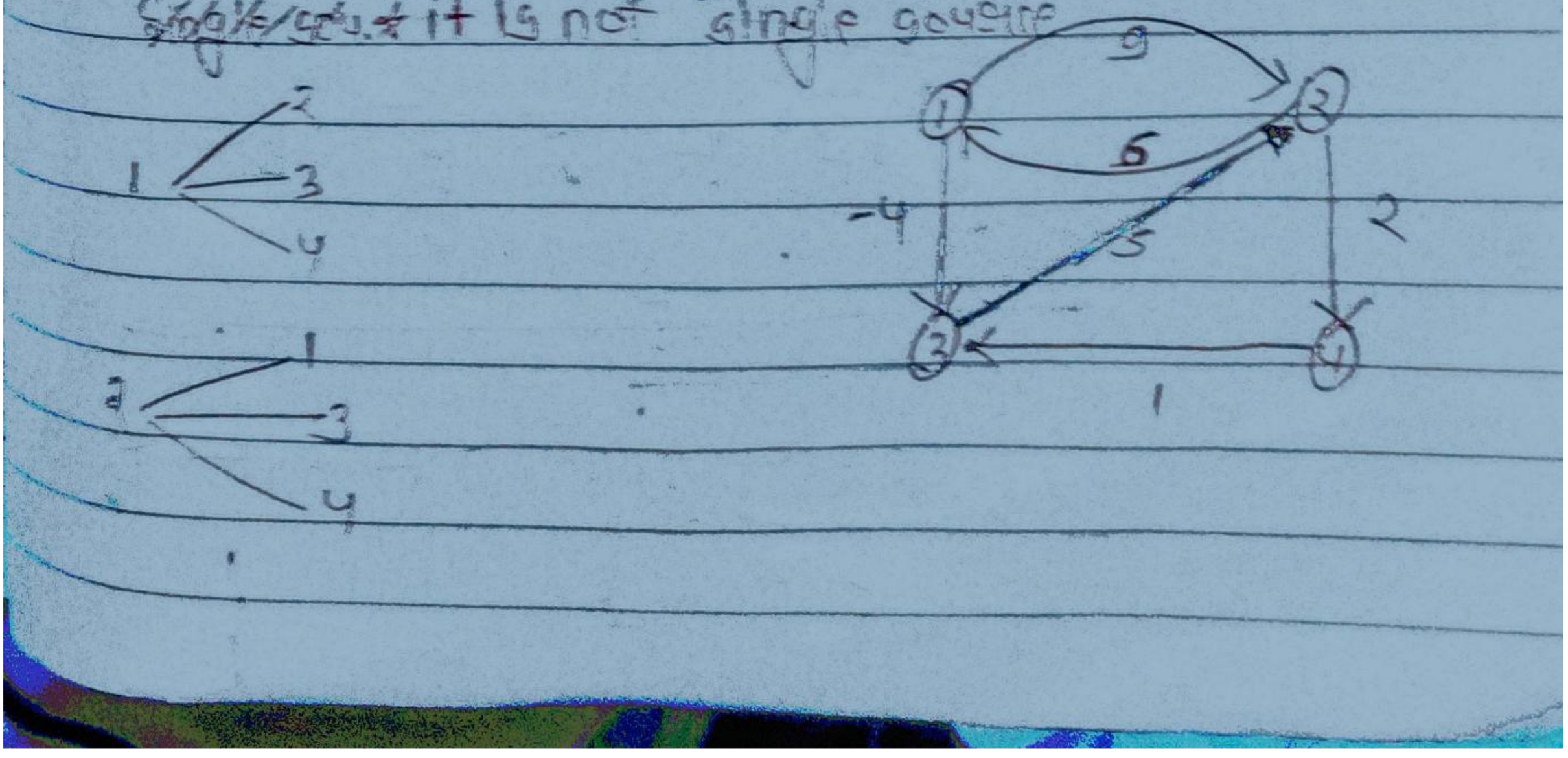


(1,3) (1,2) (3,2) (2,4) (4,3)

if in ith iteration dist change shows so not working because contain -ve cylce.

All pair shortest path (Floyd washall Algo)

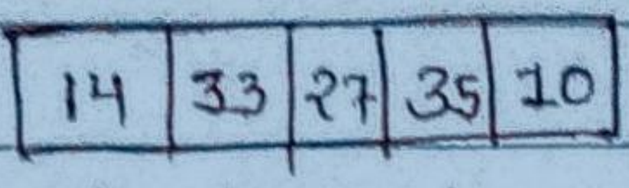
Single/src. it is not single source



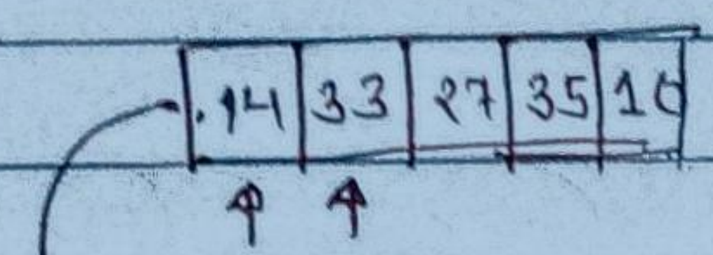
SORTING

1) ⇒ Bubble sort → each pairs of adjacent elemnts are compared and swapped if they are not in orders.

⇒ Working ⇒



→ start with first two elements and compare those



→ No swapping (14<33)

→ swapping (33>27)

→ Noswapping (35>33)

→ swapping (10<35)

→ Array after 1st iteration

repeat iteration for (size-1) time.

Note ⇒ if in any iteration their is no swapping then break. the whole process. because array become sorted.

HTML → Responsive

CSS

JS → Left

PYTHON → ✓

DJANGO → ✓

C++ → ✓

OPPS in C++ → Left

DS / ALGO → Left

CP → Left

JS

elements | Events

1) onclick
2) on hover
3) onmousehover
4) onmousedown
5) onmouseout
5) onsumit

eval (doc. exam.. calc. value)

Dialoge box ⇒ 1) prompt
2) alert
3) confirm

maths ⇒ 1) random
2) floor
3) clie
4) round
5) log
6) sqrt

Date() var dt = new Da
dt.getDay()
— / — fullyear()
— / — Hours()
minutes
months
seconds
milliseconds

getTime
↓
1/1/1970

get.date

JAVA SCRIPT Note

<script src="js/dom.js"></script>

DOM

Date / / Page no.

How to target DOM objects?

1) id → doc.getElementById("")
2) class name → ——|——|——|—|—getClassName("")
3) Tag name ——|——|——|—|—TagName()

1) doc.
2) doc.all
3) doc.documentElement
4) doc.head
5) doc.title
6) doc.body
7) doc.images
8) doc.anchors
9) doc.links
10) doc.forms
11) doc.doctype
12) doc.URL
13) doc.baseURL
14) doc.domain
15) doc.getElementsByClassName
16) doc.getElementsByTagName
17) doc.getElementById()

what we can get with DOM?

1) HTML
2) Text
3) Attribute

- innerText
- innerHTML
- getAttribute
- getAttributeNode
- Attributes
- .value
- .name

DOM Set Methods

1) innerText
2) innerHTML
3) getAttribute.
4) Attribute.
5) removeAttribute

DOM Get Methods

- doc.querySelector → select 1st one
- doc.querySelectorAll → select all elements

DOM css styling methods:

1) style
2) className
3) classList → return list (Array)

- click (onclick)
- Doubleclick (ondblclick)
- Right click (oncontextmenu)
- mouseHover
- MouseOut
- MouseDown
- mouse up
- KeyPress
- Key UP
- Load
- Unload
- Resize
- Scroll

classList

1) add(class1, class2, ...)
2) remove(class1, class2, ...)
3) toggle (class)
4) contains (class)
5) item (index)
6) length.

addEventListener().

doc.getElementById(id).addEventListener ("click", myfn)
("click", myfn
stat });

access children of body
doc.body.children;

add a new child in body

var newchild = doc.createEle('p');
var TextNode = doc.createTextNode("-")
para.appendchild.(TextNode);
doc.body.appendchild(para).

var list = doc.querySelector("#1")

console.log (list.FirstChild);
(list.ParentNode);

id.nextElementSibling.nextElementSibling

CSS styling using DOM

da.getEte.('').style.colour = "Red";

doc.getEle(').setAttribute("style", "colour:red;
font:italic;");

Window location;
window.location.Hostname
window.location.href
http://
window.loca = "www.youtube.com")

window.loca.assign("https://ww-1-");

Print the Page

For whole Function my( ){ window.print();}

For div Fun my(id){

```
var backup = doc.body.innerHTML;
var divcont = doc.getEle(id).innerHT
doc.body.innerHTML = divcont;
window.print();
doc.body.innerHTML = backup;
```

Date and Time

```
var d = new Date();
var n = d.getTime();

var hour = d.getHours;
           min;
           Second;
d.toLocalTimestring();
```

```
var clock = setInterval (clocktiming
                         1000)
fun clockTiming(){

    var d = new Date();
    var x = x.tolocalTimestring();
    doc.write(x)
doc.get(id).innerHTML = x }

.clearInterval(Clock);
```