	18/06/2023 ADMPD - 12911 ALTON
	Cycle Detection in undirected graphs
	SYC) (2)
	(4) (3)
	Here we can see the cycle 2-3-4.
4	is a produce north both it comes a great
<u>(1)</u>	Using BFS
	l vioil a l
	<u>visited</u> <u>parent</u>
à (Jac	0-18 Ton who regions (0-) - 12 days of the
	1-87 1-0
	277 271
	3→ FT 3→ 2
	4(+ FT) +21001 - 4+120 00000000000000000000000000000000000
	Charletter & south 1991
	que - fo3 (mork O - T)
	Store front node, pop it and insert neighbours
1)	
,	queue \rightarrow {13 (Set parent of $1 \rightarrow 0$) $0/p \rightarrow 0$ (mark $1 \rightarrow T$)
	C digital and a
- 2.)	queue - {23 (Set parent of 2 - 1)
hari	queue \rightarrow {23 (Set parent of $2 \rightarrow 1$) 0/ $\beta \rightarrow 0$ 1 (mark $2 \rightarrow 7$)
3)	queue - 13,49 (Set parent of 3 -2 and 4-2)
	que → {3,43 (Set povent of 3 → 2 and 4 → 2) 0/p → 0 1 2 (mark 3 → T and 4 → T)
	Scarineu wini cail

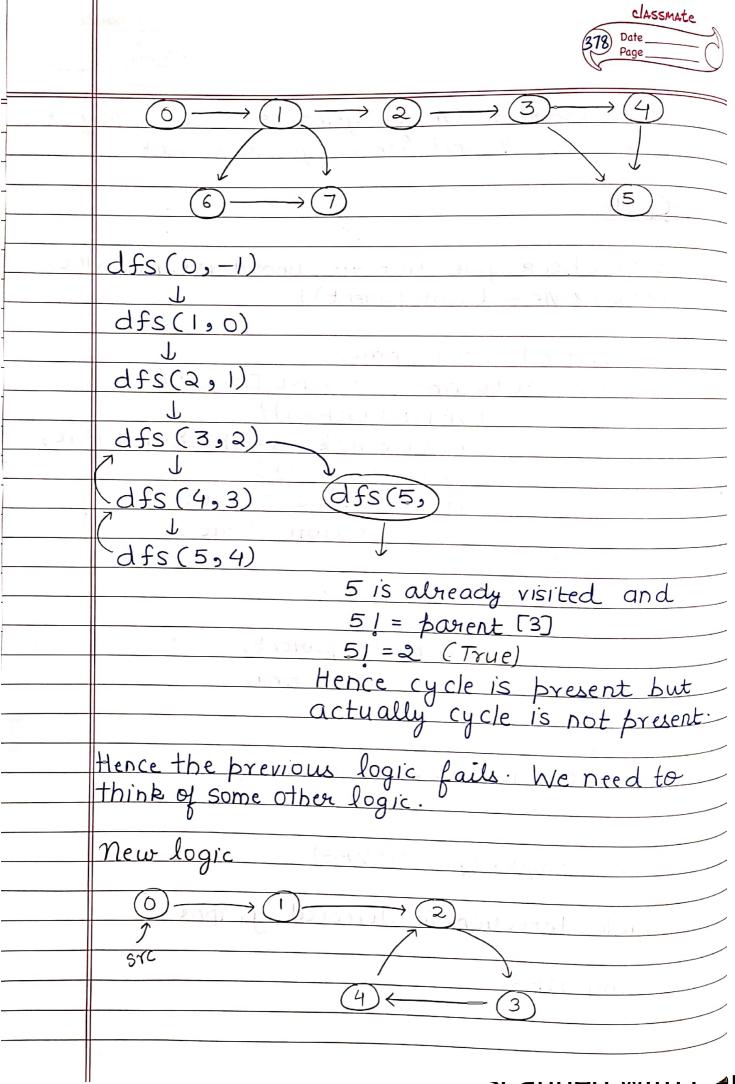
Scarineu with Cail

	Now front node is 3, it is now trying to go to 4.4 is already visited but is not farient of and hence cycle is present.
	4.4 is already visited but is not bayent of
	and hence cycle is present.
ote-	Parient will be set only once . If we are trying to set parient of node again, this means cycle is
	to set parient of node again, this means cucle is
	present.
	i return falle i
	2 Conditions -> Already visited & is not parent,
	cycle is present.
	The state of the s
	Code English Call Shippens 19 that I have the
	all the draw in) old in the draw of the draw in its d
	bool checkCycle (int src, unordered-map (int, bool)
	Avisited) {
	quiue (int)q;
	unordered_map <int, int)="" parent;<="" th=""></int,>
	q. push (src);
	visited [src] = true;
	parent [syc] = -1;
	(3+1) O = phixaldum = of f = e
	While (1q.empty())
	int front Node = q front ();
	g.bob(); for (auto nbr: adylist [front Node]) {
/	if (I visited [nbr]) {// push in queue
/	q. bush (nbx)
/	visited (nbr) = true;
/	barent (nbr) = front Node;
	100 (2) 2+103- (6) 2-10 (0) 2-05/
/	else { //already visited
NEW YORK	公里 3.

Scarnieu Willi Carn

	if (nbr. ! = parient (frontNode)){
	// cycle Bousent
	retuin true;
	}
TU'aa	A 340 AC DA LESON WORLD BE DE MONTH THOUSE IN THE
0 0 82 3	3 Ranger sett disposed on the thereof to
	3
	return false;
1 + 13	3d too a botton whise IA is madilla is
	// In main ()
	for (int i = 0) i(n) i++) {
	if (I visited (i)) {
•	ans = g.check(ycle(i, visited))
(kood e	talled on bif (ansi = true) shubshind inno
	break j
	2
	ans + true (Cycle present)
-	ans + false (Cycle not present)
	,
*	Time complexity = $O(V+\epsilon)$
	The complete of the second of
(ii)	Using DFS
C	
7 12 5	0 2
- 4, 1	1
	SYC
	4) (3)
, S) ,	b=3
	$dfs(0) \rightarrow dfs(1) \rightarrow dfs(2) \rightarrow dfs(3) \rightarrow dfs(4)$
	b=-1
'	Scarnicu With Carll

Here from 4, we are going to 2 which is not barrent of 4 and hence cycle is present. Code bool check Cycle (int src, unordered-map (int, bool) & visited, int farent)? Visited [src] = true; for (auto nbr: adjlist [src])? If (! visited [nbr])? bool check Ans = check Cycle (src, visited, src); if (check Ans = = true) //cycle present return true; 3 else? // already visited // cycle present if (nbr! = parent) > same condition of return true; 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs i) Using DFS		
bool check (ycle (int src, unordered map (int, bool > & visited, int parent) { Visited [src] = true; for (auto nbr: adjlist [src]) { if (! visited [nbr]) { bool check Ans = check (ycle (src, visited, src); if (check Ans = = true) // cycle prusen return true; } else { // already visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		Here from 4, we are aging to 2 miles
bool check (ycle (int src, unordered map (int, bool > & visited, int parent) { Visited [src] = true; for (auto nbr: adjlist [src]) { if (! visited [nbr]) { bool check Ans = check (ycle (src, visited, src); if (check Ans = = true) // cycle prusen return true; } else { // already visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		hagent of 4 and here a such is
bool check (ycle (int src, unordered map (int, bool > & visited, int parent) { Visited [src] = true; for (auto nbr: adjlist [src]) { if (! visited [nbr]) { bool check Ans = check (ycle (src, visited, src); if (check Ans = = true) // cycle prusen return true; } else { // already visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		parent of and nence cycle is present.
bool check Cycle (int src, unordered_map <int, bool=""> & Visited, int parent) { Visited [src] = true; for (auto nbr: adjlist [src]) { if (! visited [nbr]) { bool check Ans = check Cycle (src, visited, src); if (check Ans = = true) //cycle prusen return true; } else { // already visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs</int,>		
bool > & visited, int farent) { Visited [src] = true; for (auto nbr: adjlist [src]) { If (I visited [nbr]) { bool check Ans = check (ycle (src, Visited, Src); If (check Ans = = true) //cycle frusen return true; 3 else { // already visited // Cycle present If (nbr! = parent) > same condition of return true; BFS: 3 3 Teturn false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		Code
bool > & visited, int parent) { Visited [src] = true; for (auto nbr: adjlist [src]) { if (I visited [nbr]) { bool check Ans = check (ycle (src, Visited, Src); if (check Ans = = true) //cycle prusen return true; 3 else { // abready visited // Cycle present if (nbr! = parent) > same condition of return true; BFS: 3 3 Teturn false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
bool > & visited, int farent) { Visited [src] = true; for (auto nbr: adjlist [src]) { If (I visited [nbr]) { bool check Ans = check (ycle (src, Visited, Src); If (check Ans = = true) //cycle frusen return true; 3 else { // already visited // Cycle present If (nbr! = parent) > same condition of return true; BFS: 3 3 Teturn false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		bool check Cucle (int sxc uppordered make lint.
Visited [src] = true; for (auto nbr: adjlist [src]) { If (I visited [nbr]) { bool check Ans = check (ycle (src, Visited, Src); If (check Ans = = true) //cycle frusen return true; 3 else { //already visited // Cycle present If (nbr! = parent) > same condition of return true; BFS: 3 3 Teturn false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		bool > e vici od ist desiret)
for (auto nbr: adjlist [src]) { If (!visited [nbr]) { bool check Ans = check (ycle (src, visited, src); If (check Ans = true) //cycle frusen return true; 3 else { // already visited // cycle frusent If (nbr! = farent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		Dock / & VISIERCE ; ITTE Parience) ?
for (auto nbr: adjlist [src]) { If (!visited [nbr]) { bool check Ans = check (ycle (src, visited, src); If (check Ans = = true) //cycle frusen return true; 3 else { // already visited // cycle frusent If (nbr! = frusent) > same condition of return true; BFS. 3 3 return false: //Cycle npt present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
if (I visited [nbr]) { bool check Ans = check (ycle (src,	1	
if (I visited [nbr]) { bool check Ans = check (ycle (src,		for (auto nbr: adjlist [src]) {
bool check Ans = check (ycle (src, visited, src); if (check Ans = true) //cycle prusen return true; 3 else { //already visited //cycle present if (nbr! = parent) > same condition of the second true; 3 3 return false; //cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs	,	
Visited, SYC); If (check Ans = = true) //cycle prusen return true; 3 else & //already visited //cycle present If (nbr! = parent) > same condition of return true; BFS. 3 3 return false; //cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
If (check Ans = = true) //cycle frusen return true; 3 else { // abready visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 return false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
else { // abready visited // cycle present if (nbr! = parent) > same condition of return true; BFS. 3 3 Yetwin false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
else & // almeady visited // Cycle present if (nbr! = panent) > same condition of return true; BFS. 3 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		II
else { // already visited // Cycle present if (nbr! = panent) > same condition of return true; BFS. 3 return false; // Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs	-	return true;
if (nbr! = parent) > same condition of yeturn true; BFS. 3 3 3 return false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		3
if (nbr! = parent) > same condition of yeturn true; BFS. 3 3 3 return false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		else & //already visited
if (nbr!= parient) same condition of yeturn true; BFS. 3 return false; //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs		
3 Time complexity = O(V+E) Cycle detection in directed graphs		if (Dbx 1 = Bayent) > same condition of
3 Yetwin false: //Cycle not present 3 Time complexity = O(V+E) Cycle detection in directed graphs	7	BFS.
return false: //Cycle not present Time complexity = O(V+E) Cycle detection in directed graphs		return true
Time complexity = O(V+E) Cycle detection in directed graphs		
Time complexity = $O(V+E)$ Cycle detection in directed graphs		
Time complexity = $O(V+E)$ Cycle detection in directed graphs		return false: // Cycle not present
Cycle detection in directed graphs		3
Cycle detection in directed graphs		
Cycle detection in directed graphs		T. 0(1/46)
		Time complexity = O(V+E)
		Cycle detection in directed graphs
Using DFS		
JING DES	W	Using DEC
		THY DES
	1	



Scarnieu with Cam

	sand GRC classmate
	Recursive call _ 379 Date
	stacks
	Adjacency list visited dfs Visited
	U STATISTICAL
Mira	0-1- 0-1- 0-1-
	1 = 2 1 - 1 - 1 - 1 - 1 - 7
	2-13 2-18T 2-18T
	374 37 77 37 07
	4-2 4-PT 4-PT
	· · · · · · · · · · · · · · · · · · ·
	dfs (0, -1)
	J
35	dfs (1,0)
Malo	
	dfs(2,1)
	J - 001 4 (200 - 11)
	dfs(3,2)
Wit C	<u> </u>
- W	dfs(4,3)
	A Land Carlo
	Trying to go to 2 but dfs Visited has entry
	for 2 → true and hence cycle is present.
M	
Wote.	Visited - once mayked true, can't become false
	dfsVisited Jonce marked true, can become false.
	Code
	bool check Cycle (int src, unordered-map <int,< th=""></int,<>
	bool > 2 visited, unordered map (int 3 Book)
	dfsVisited) {

Visited (src) = true;

dfs Visited [src] = true;

Scarineu with Caril

	for (auto nbr: ady'List [src]) {
	1 (I vici bod I phy I) ?
	bool check Ans = check (ycle (nbr, visited, dfs Visited);
	visited, dfsVisited);
	if (check Ans)
	return true;
	else {
	if (dfsVisited[nbr])
	return true;
	3
	3
	// Backtrocking (Here 9 can do mistake) dfs Visited [src] = false;
_	dfs Visited [src] = false;
	retwin false;
_	3
- Wote-	Rotten oranges + no of islands are very very important questions (Solved easily with the help of BFS and DFS)
7	important questions. (Solved easily with the
- Asse	help of BFS and DFS)
	they is every and hope your springer
	a way of transe markets by a day to be a second
1.5,200	16 /12/10/10 10 10/10/10 (23/10) 10/10/10 (1/10/10/10/10/10/10/10/10/10/10/10/10/10
:	
	and the description of the state of the stat
	account of the formal state of the state of