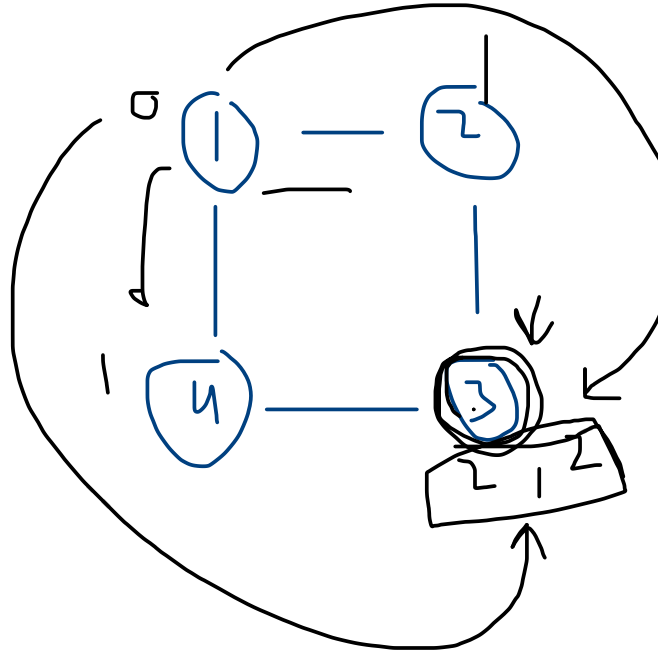
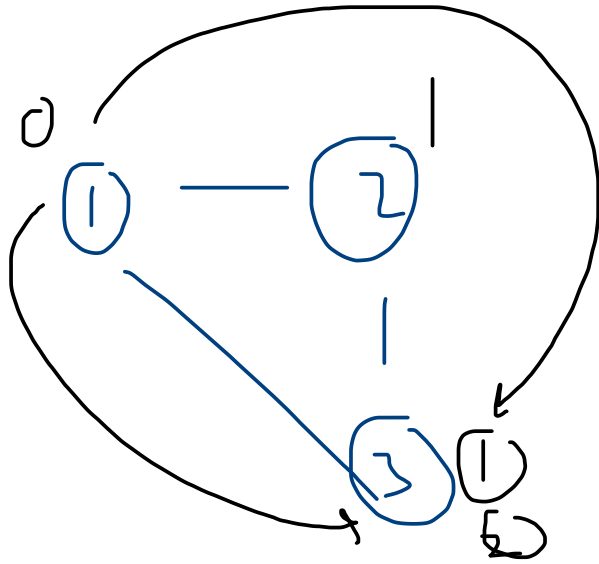
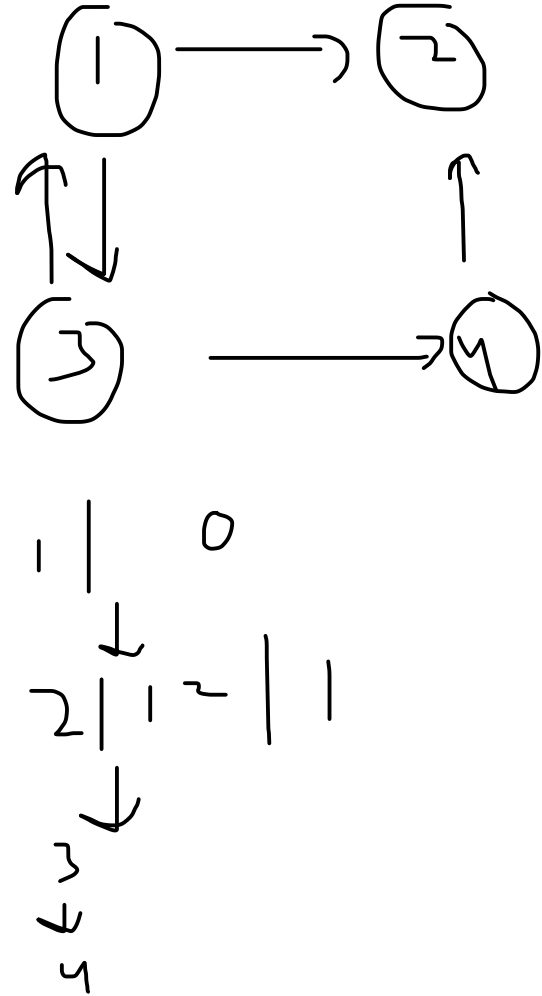
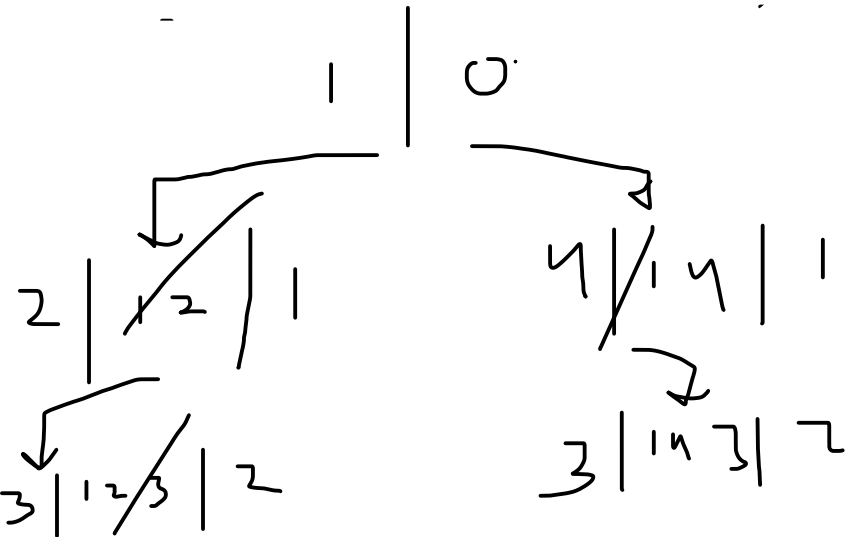
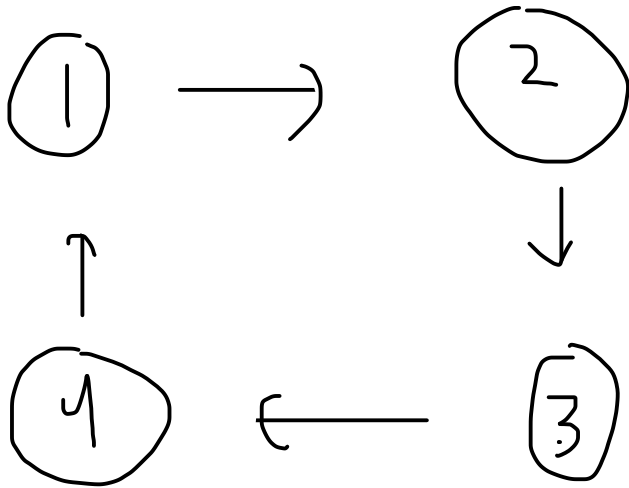


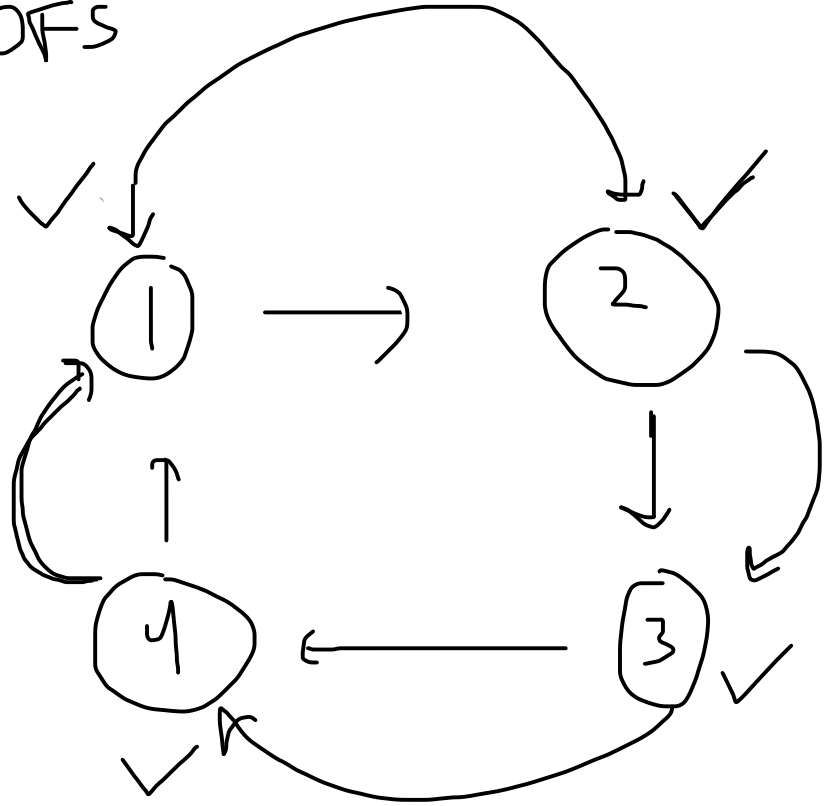
isBipartite

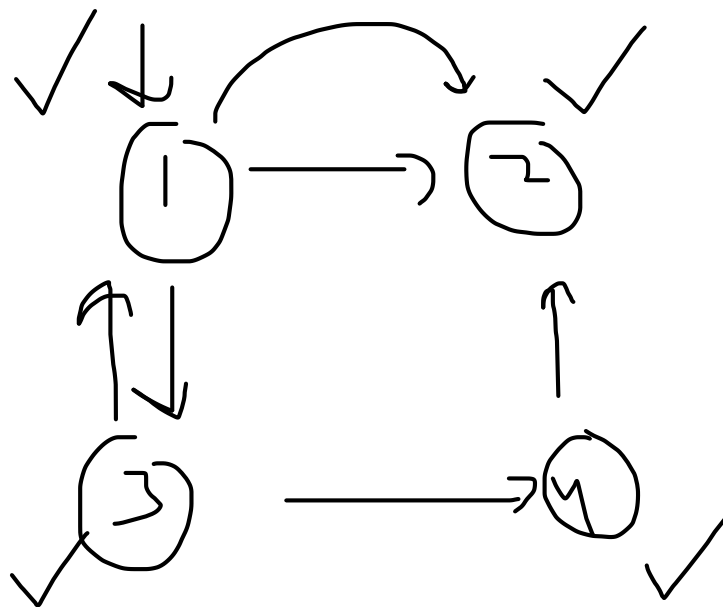


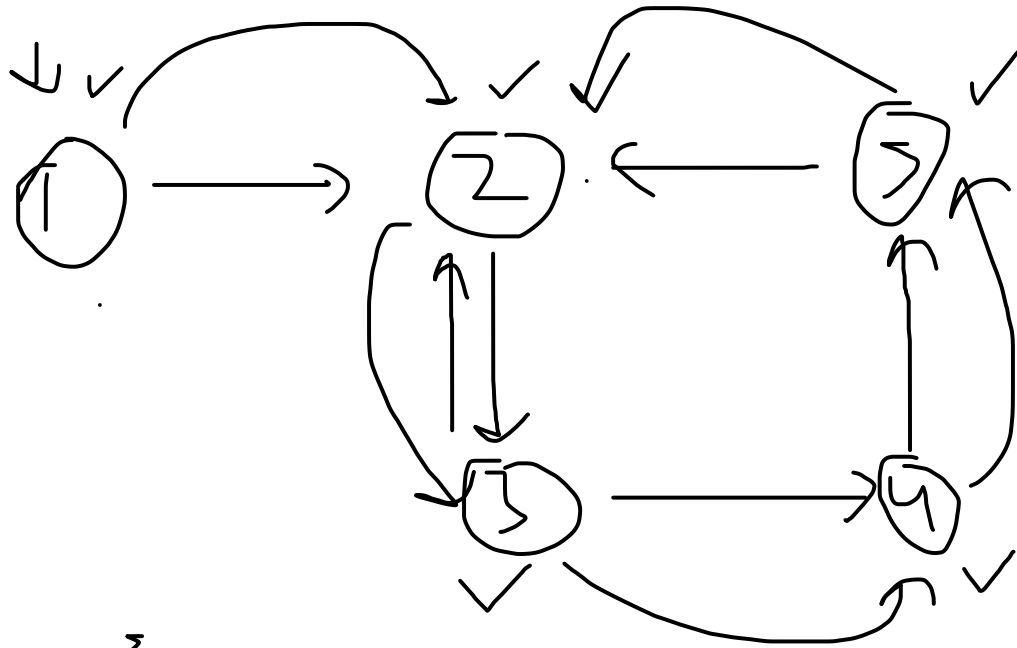
Detect Cycle in a Directed Graph



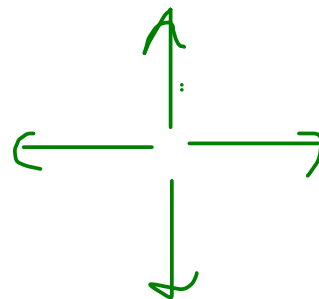
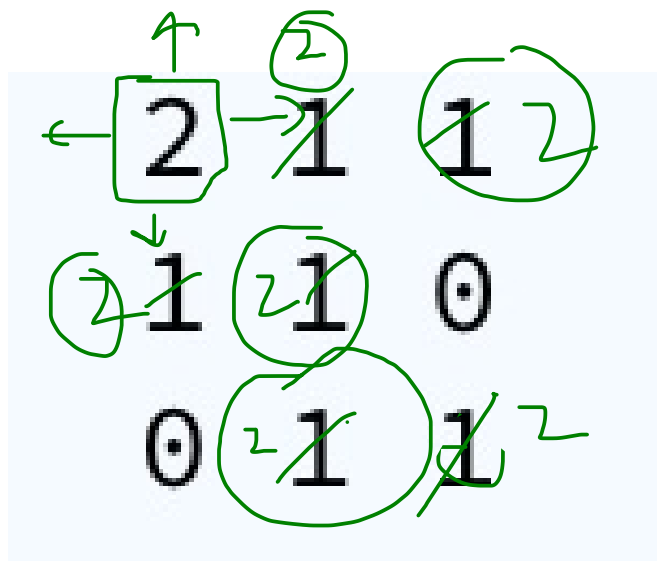
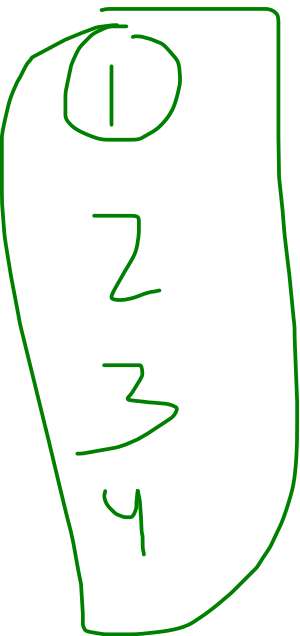
DFS

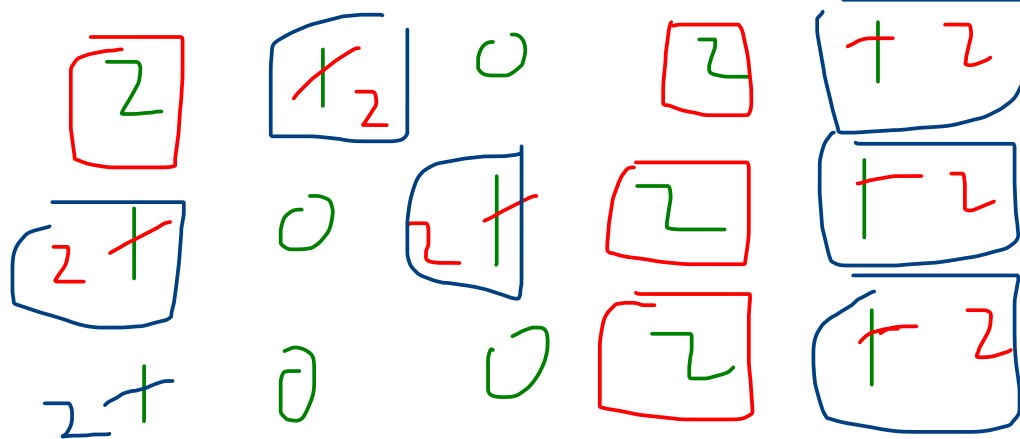
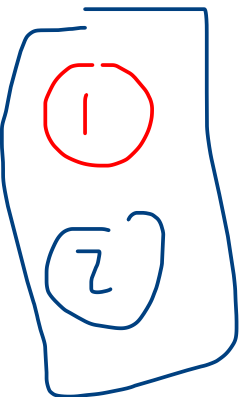






1	2	3	4	5
F	T	F	F	F
T	T	T	T	T





multisource BFS

	0	1	2	3	4
0	2	1	0	2	1
1	1	0	1	2	1
2	1	0	0	2	1
3	1	1	1	1	1

num of Fresh = ~~12~~ 10 ~~9~~ 8 7 6
~~5~~ 4 3
~~2~~ 1
0

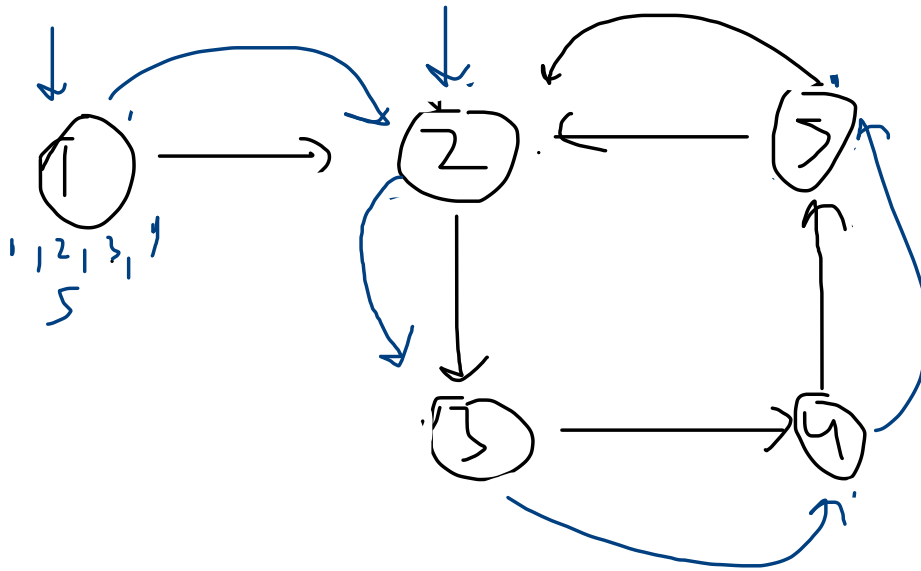
	0	1	2	3	4
0	2	+2	0	2	+2
1	+2	0	+2	2	+2
2	+2	0	0	2	+2
3	+2	+2	+2	2	+2

Time = ~~0~~
~~1~~
~~2~~
3

Power
 2¹⁰

	0 level	1 level	2 level
0	(0/0)	(0/1)	(0/2)
1	(1/0)	(1/1)	(1/2)
2	(2/0)	(2/1)	(2/2)
3	(3/0)	(3/1)	(3/2)

~~(3,0) (3,1)~~



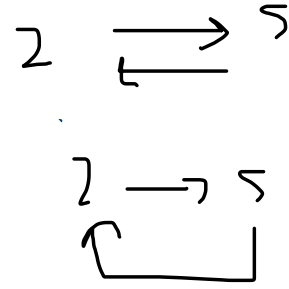
```
// base cases
if (inStack[vtx]) {
    return true;
}
if (visited[vtx]) {
    return false;
}
visited[vtx] = true;
inStack[vtx] = true;
for (Edge e: graph[vtx]) {
    if (isCycle(e.nbr, visited, inStack, graph)) {
        return true;
    }
}
inStack[vtx] = false;
return false;
```

visited

0	1	2	3	4	5
F	F	F	F	F	F
	T	T	T	T	T

in stack

0	1	2	3	4	5
F	F	F	F	F	F
	T	T	T	T	T



5					
4	1	2	3	4	5
3	1	2	3	4	5
2		1	2	3	4
1			1	2	3

Return stack