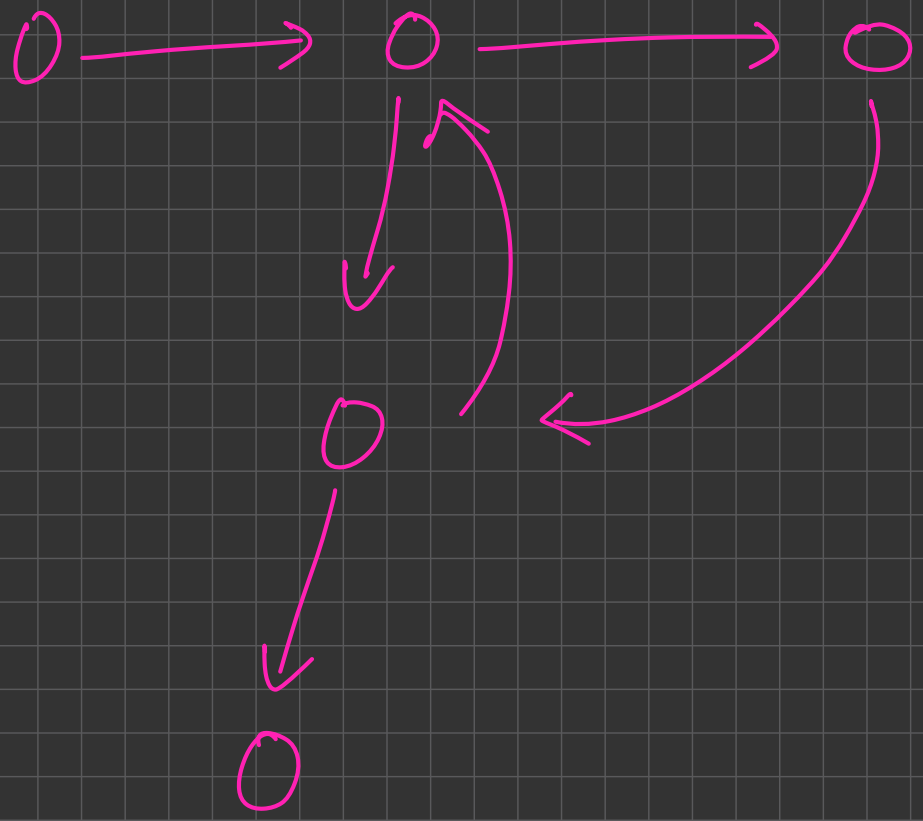
Directed Graphs

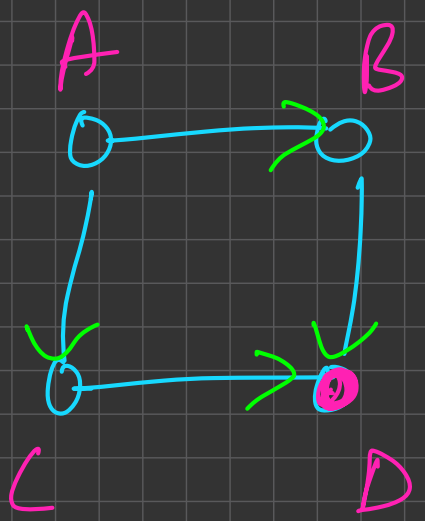
Graph Theory

DP on Graphs

DP on Trees

Directed Graphs, Cycle Detection and DAGs, Topo Sort & Strongly Connected Components

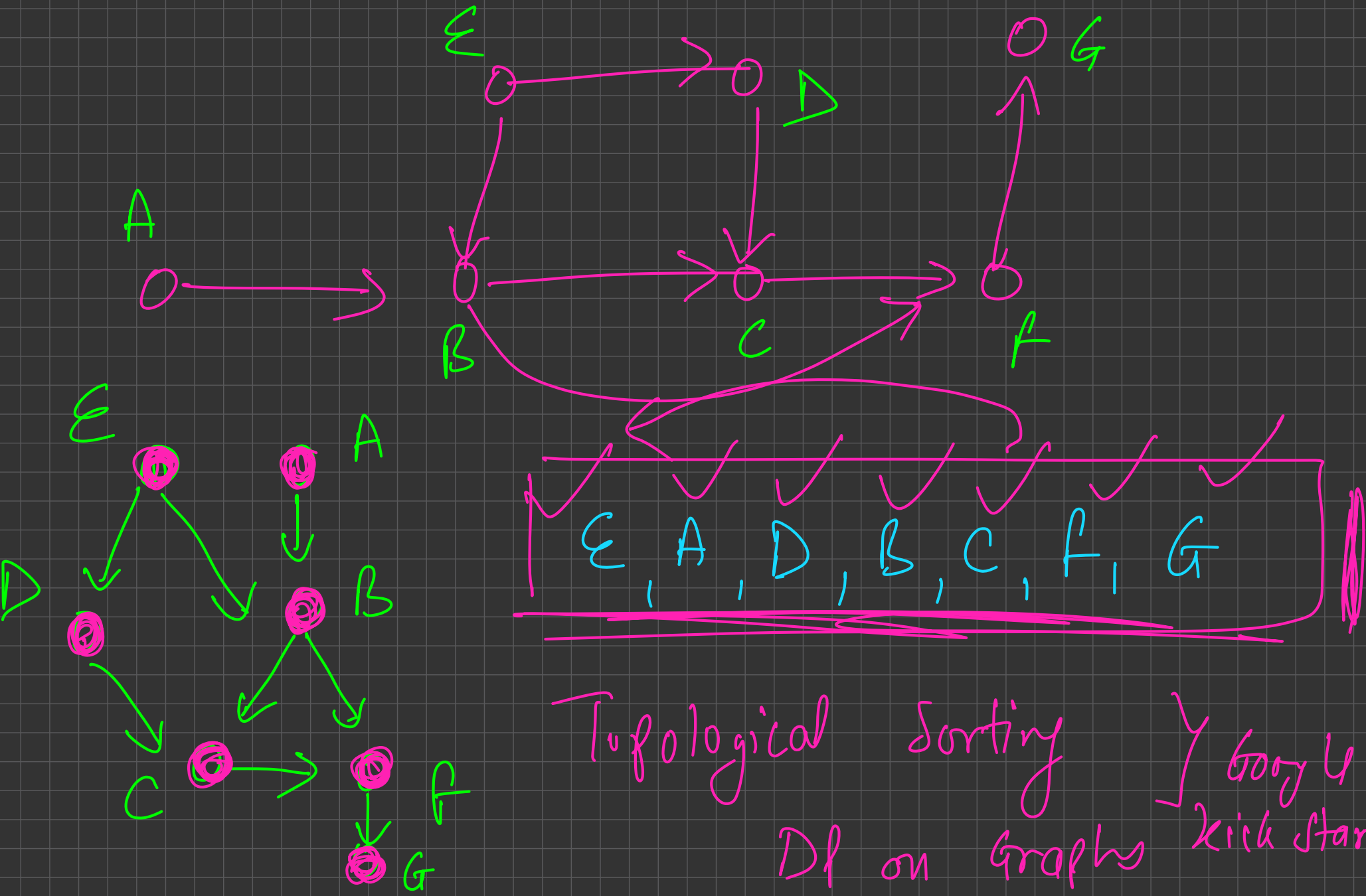




Undirected : if there exist more than 1 path from x to y

Directed : if there exists a path from x to y and also from y to x

Directed Acyclic Graph



A E D B C F G

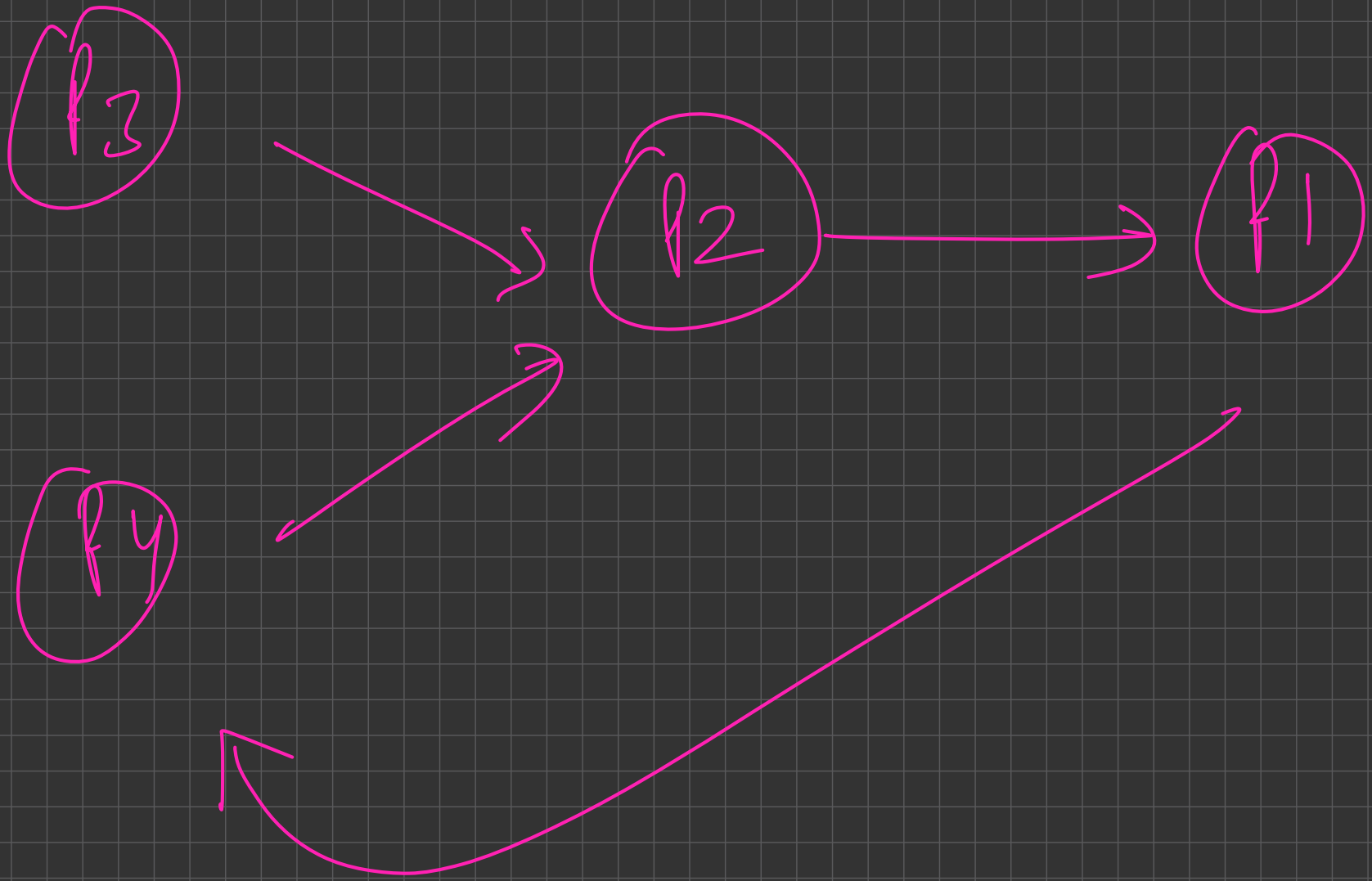
lazy polling

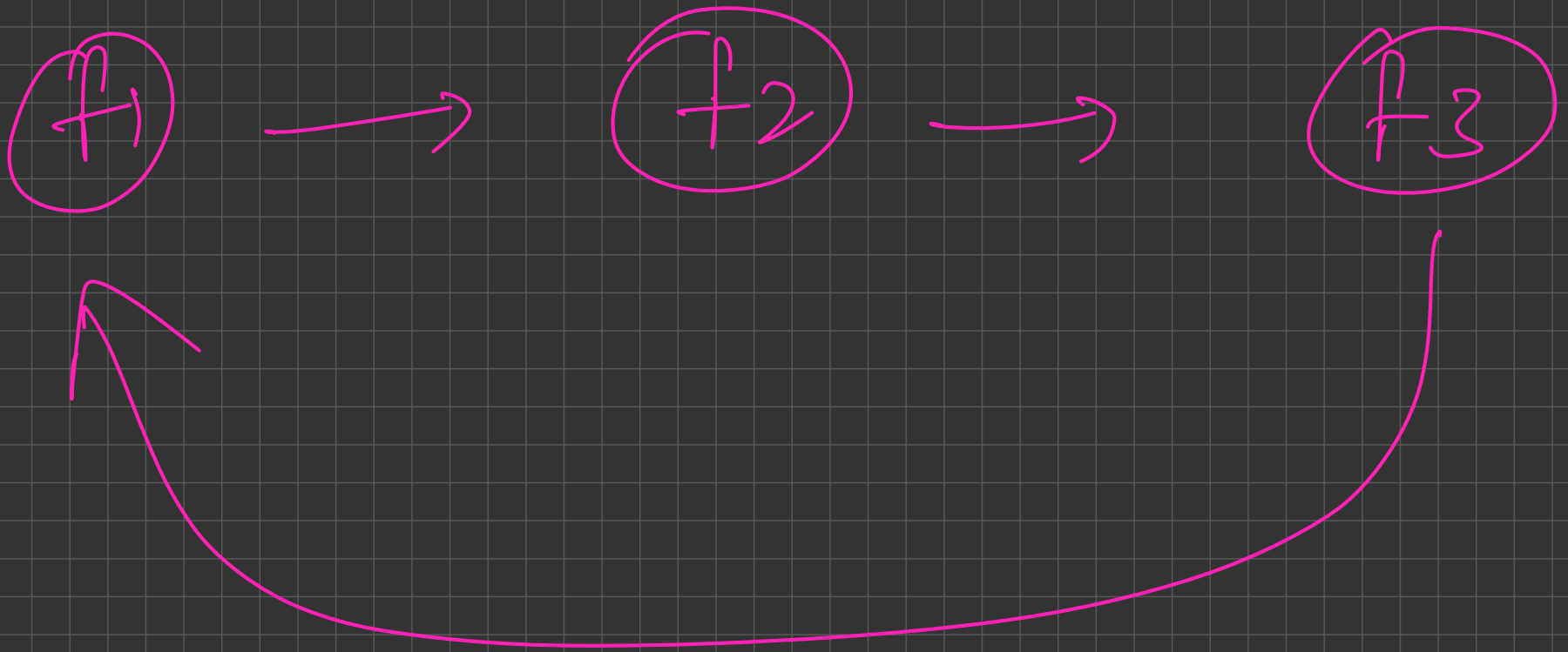
$$\{A, E, B, D, C\}$$

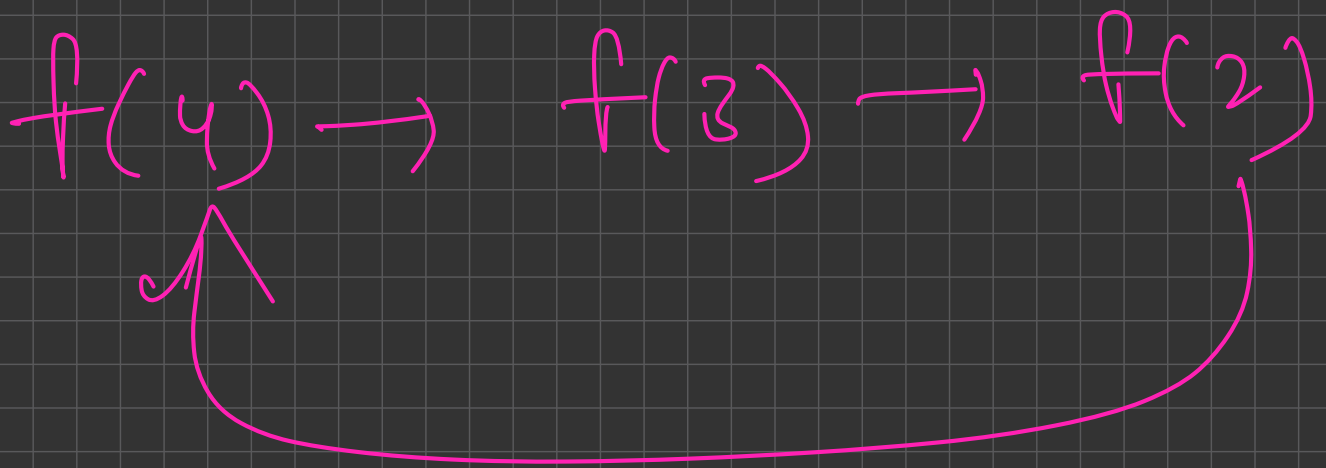
$$\{F, G\}$$

$$\{x, \{y\}\}, \{x, \{y\}\}$$

$$\{u, \{z\}\}, \{u, \{v\}\}$$


f3
f2
f1
f4

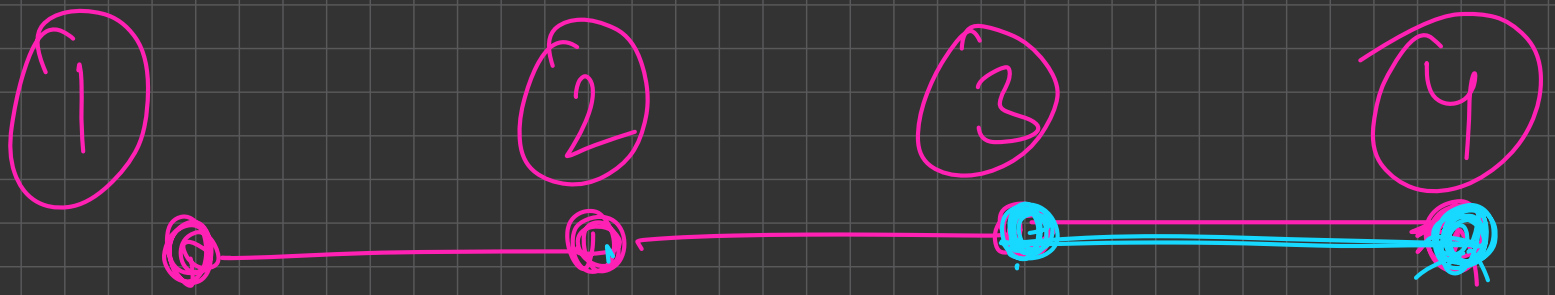

f1
f2
f3

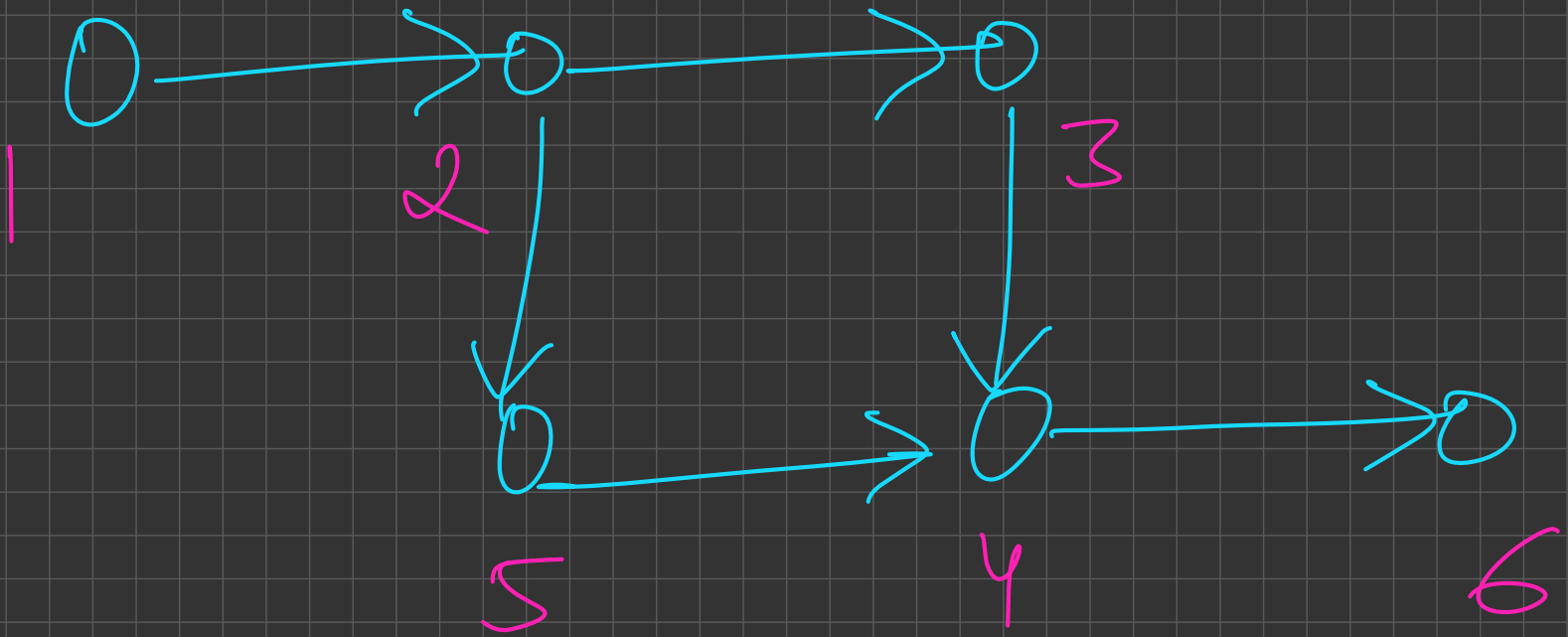

f(4)
f(3)
f(2)

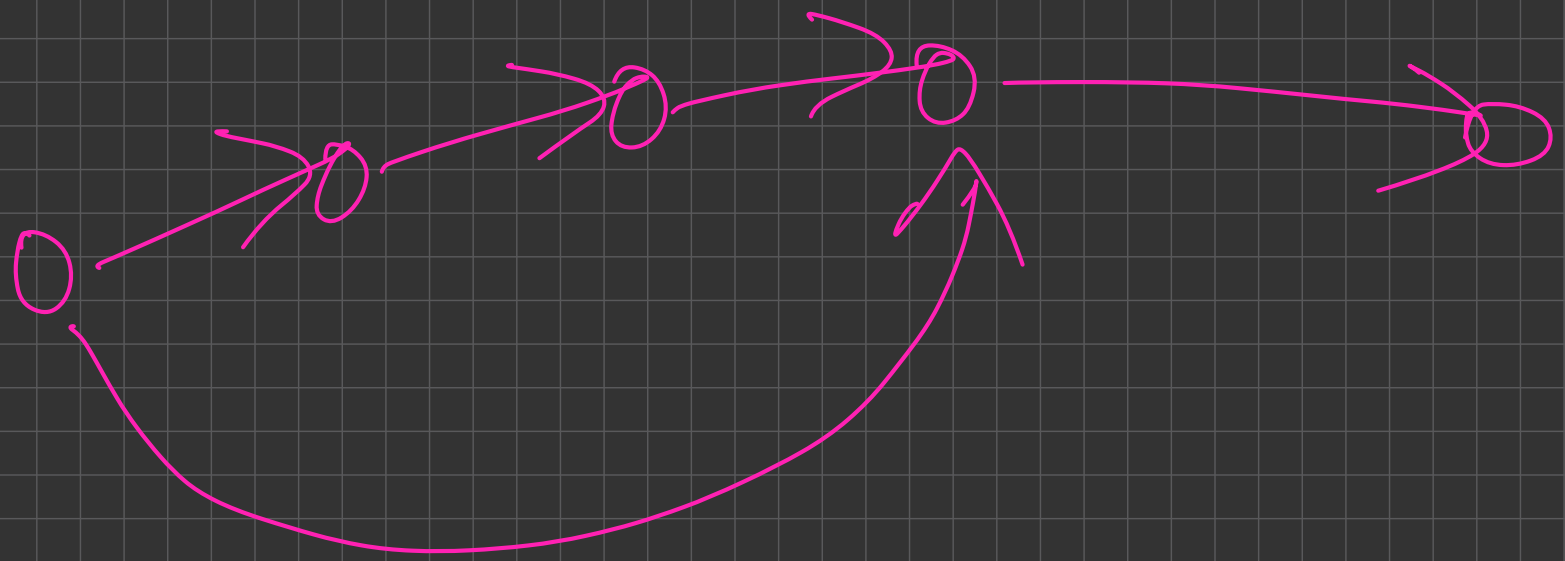
Bfs

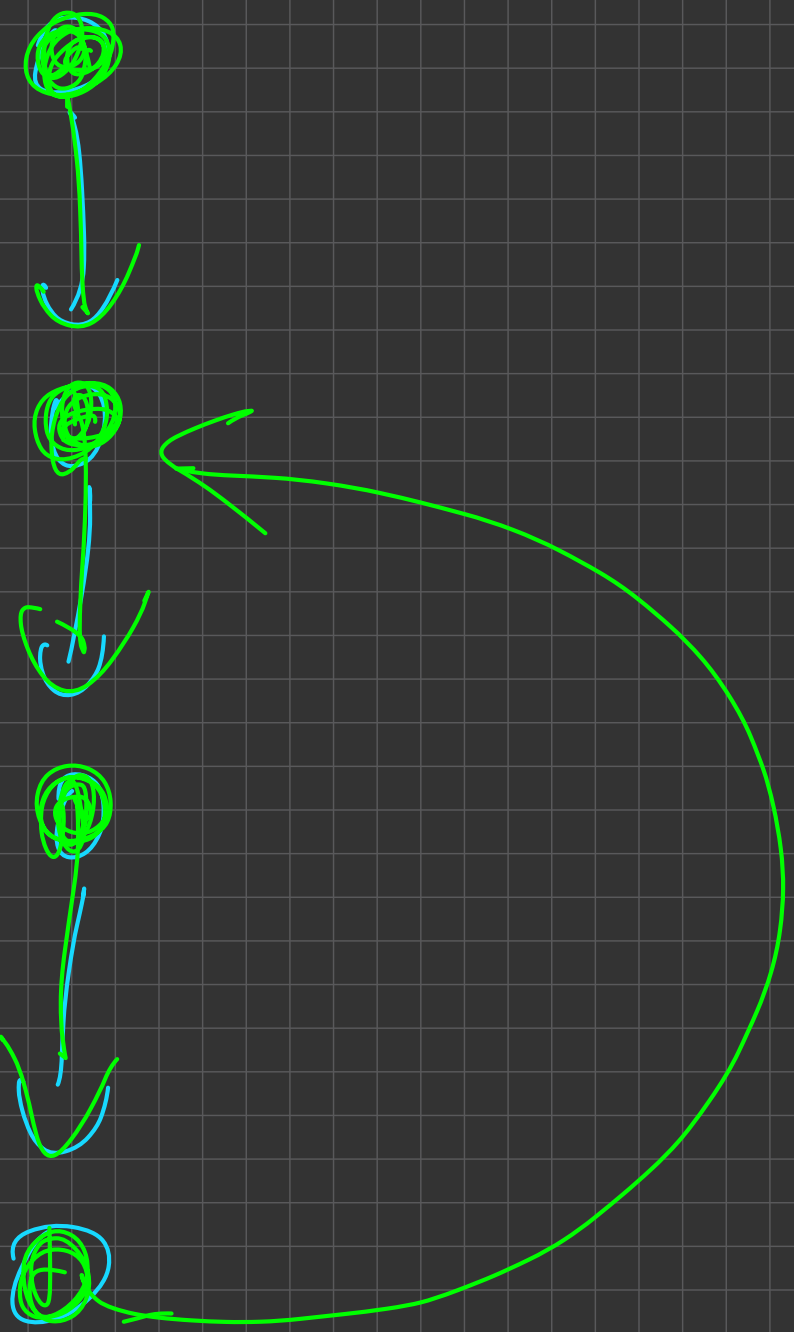
Dfs

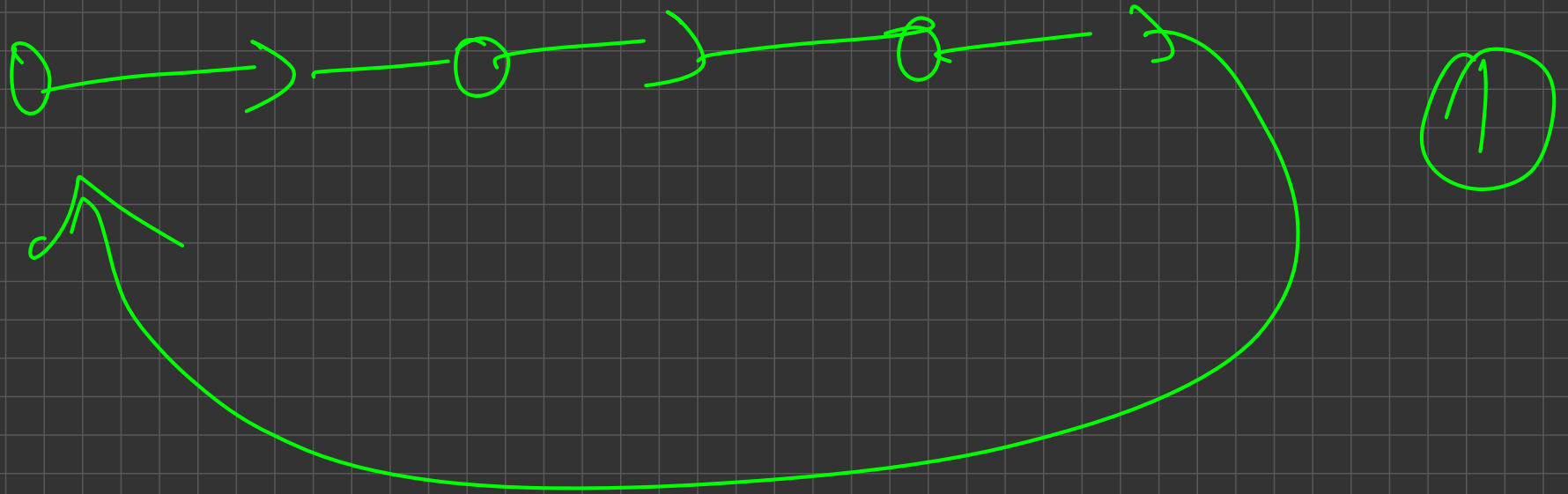
Simple graph

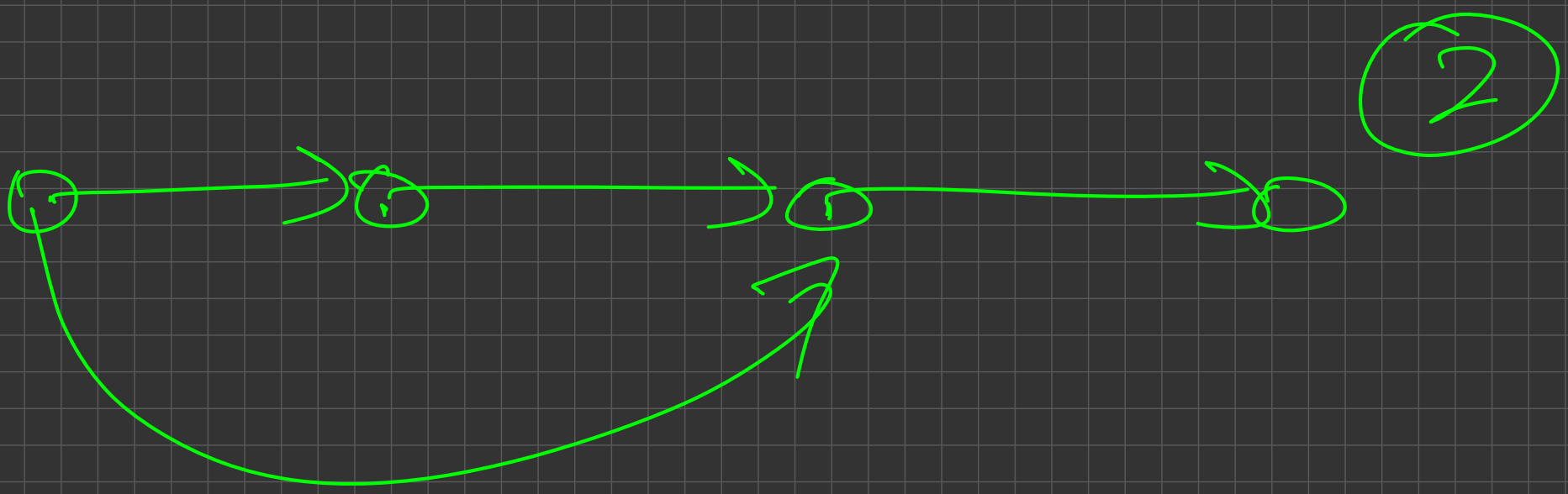


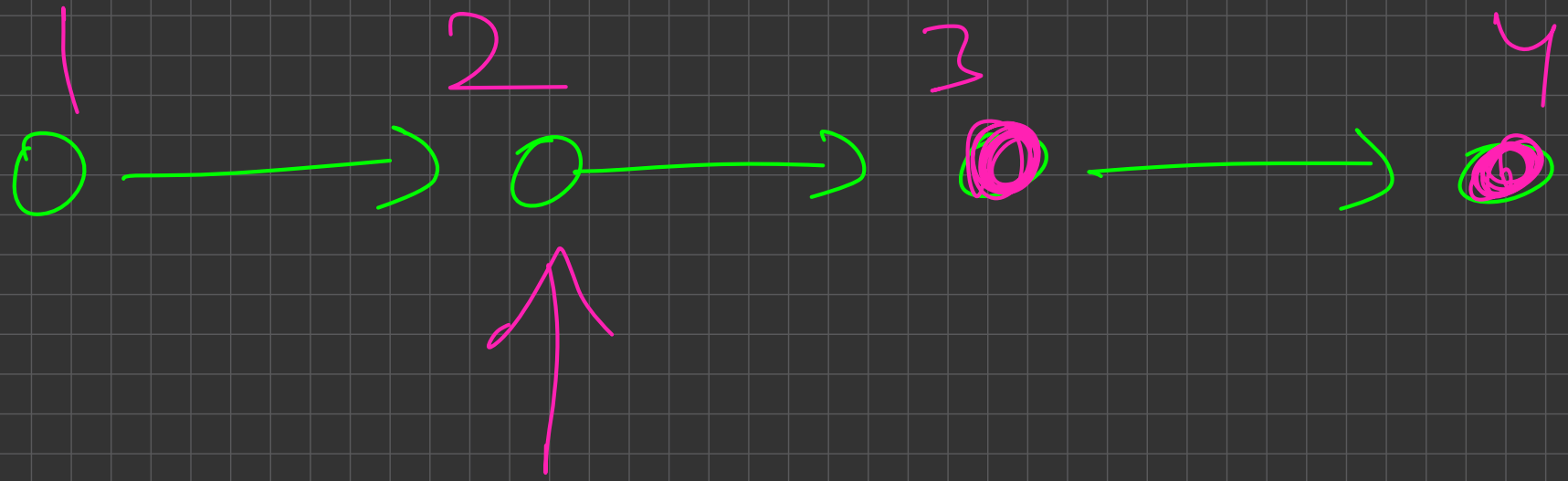

1
2
3
5
4
6

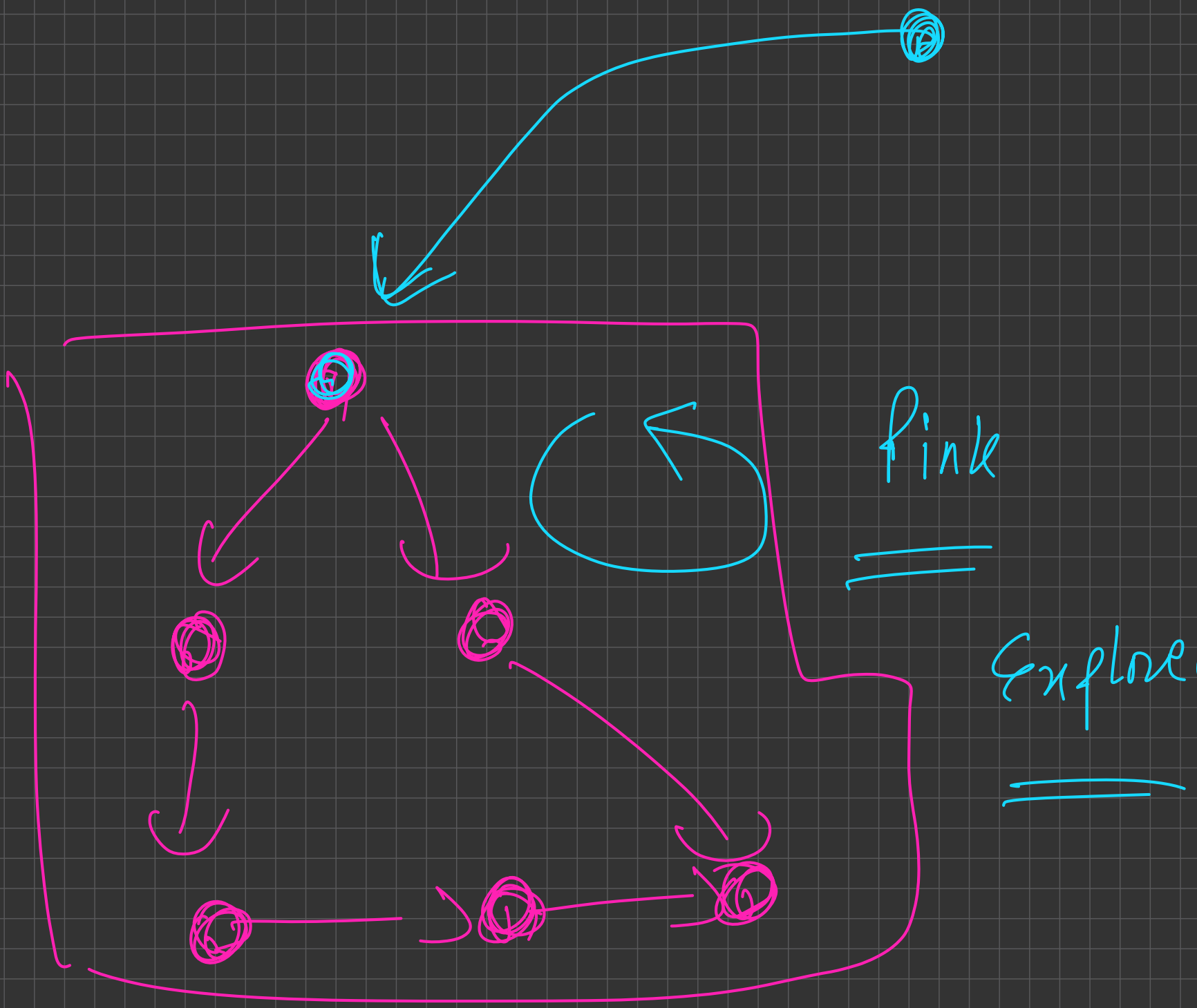


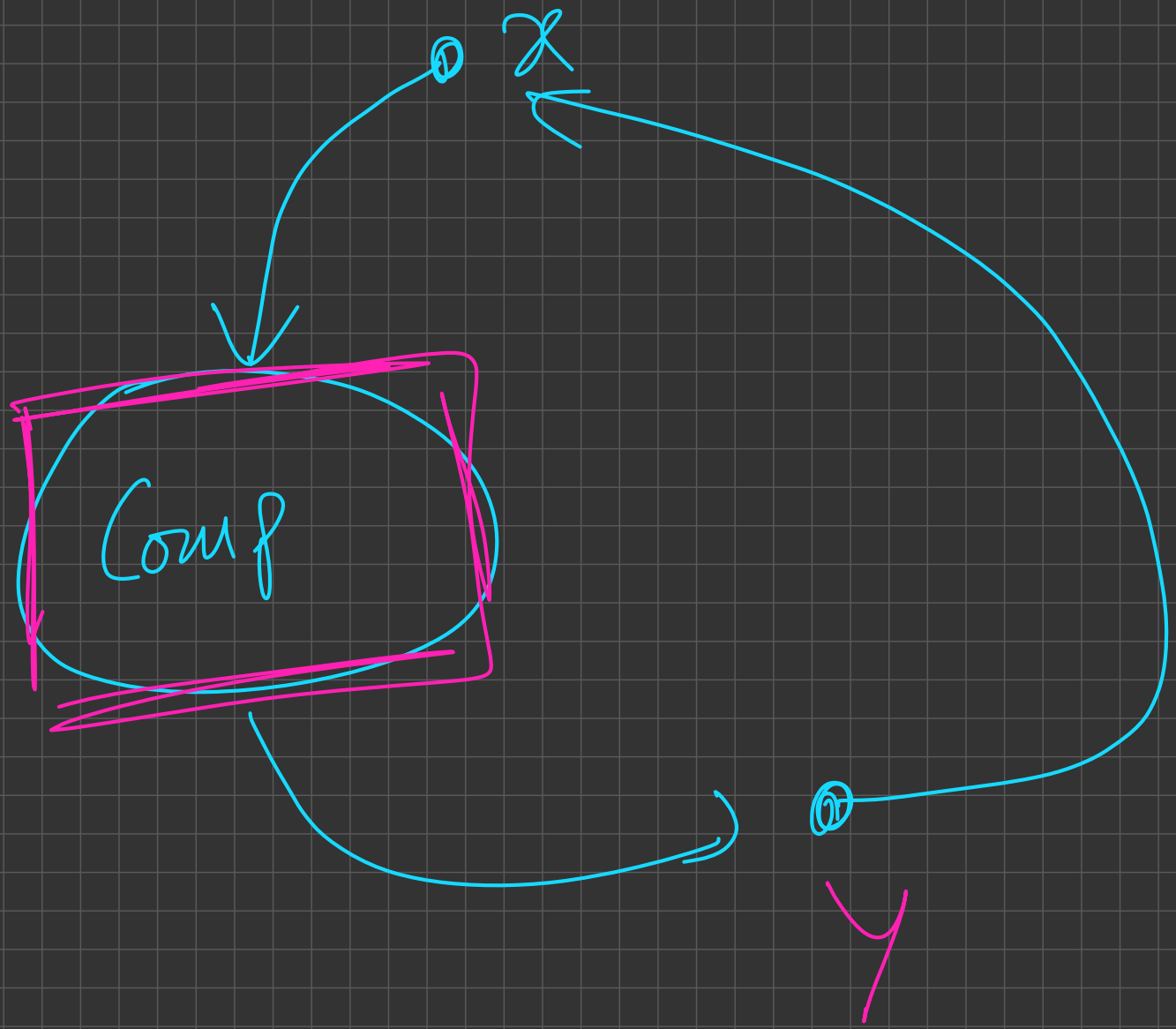



1


2


1
2
3
4

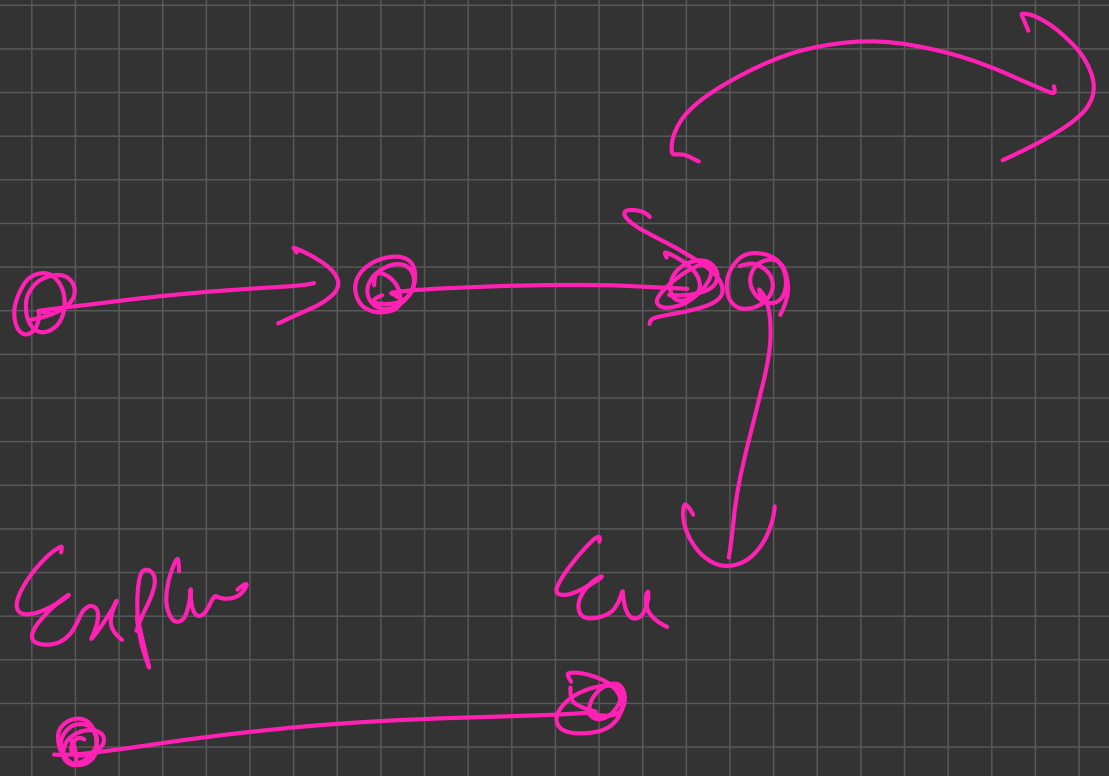

Pink
Exploded

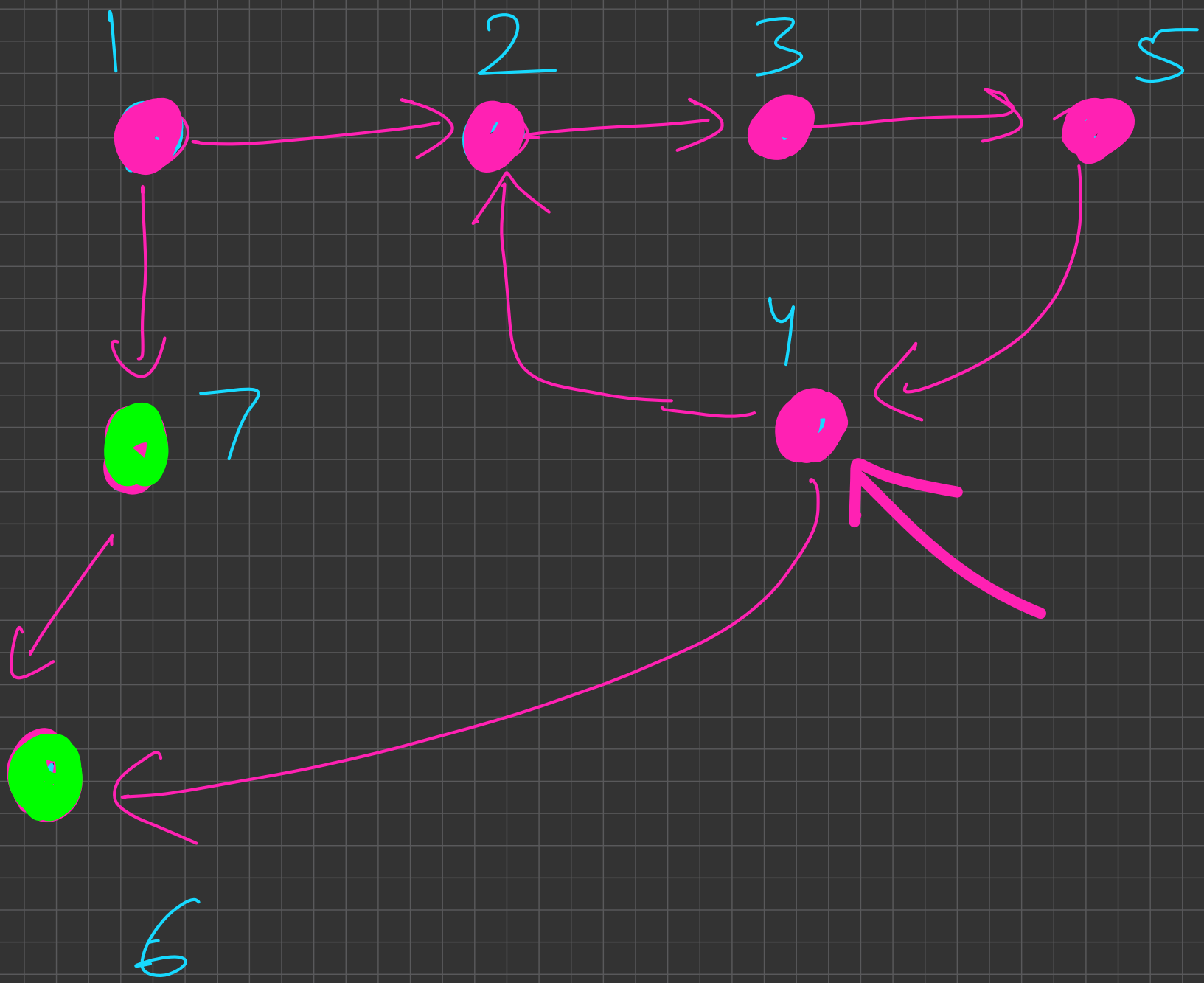

x
Comp
y

Unexplored

Exploring

Explored





U

End

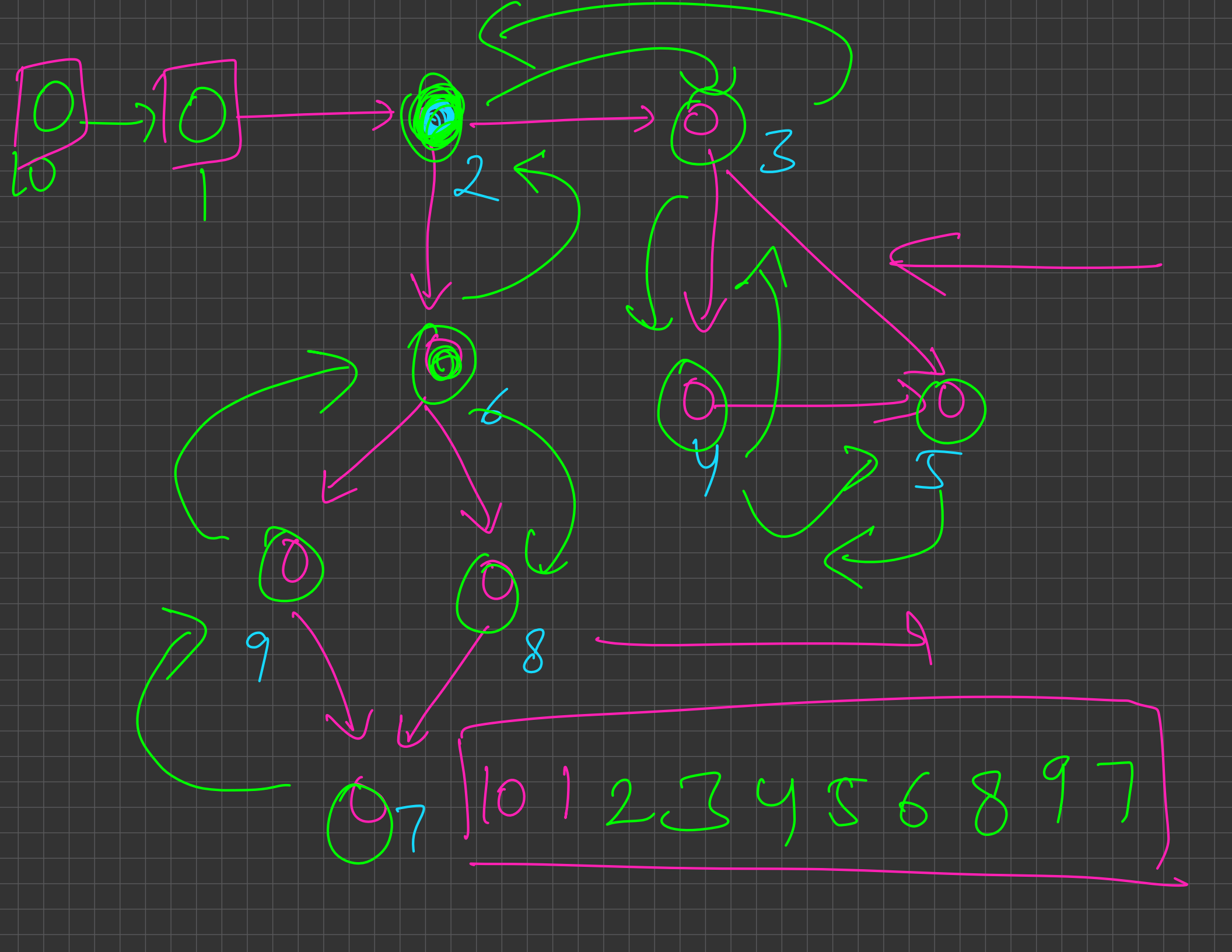
Exploring

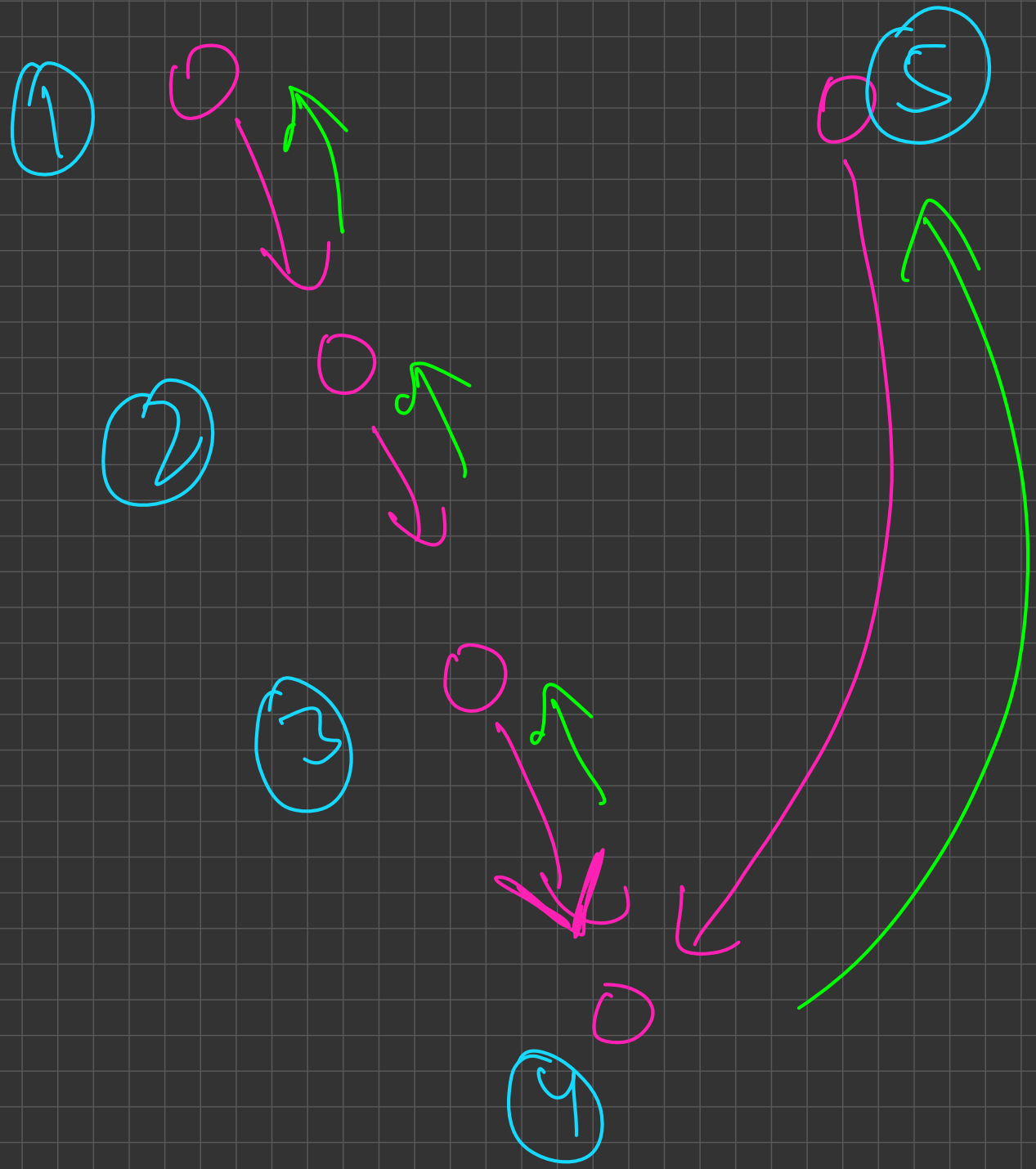
Detecting a cycle

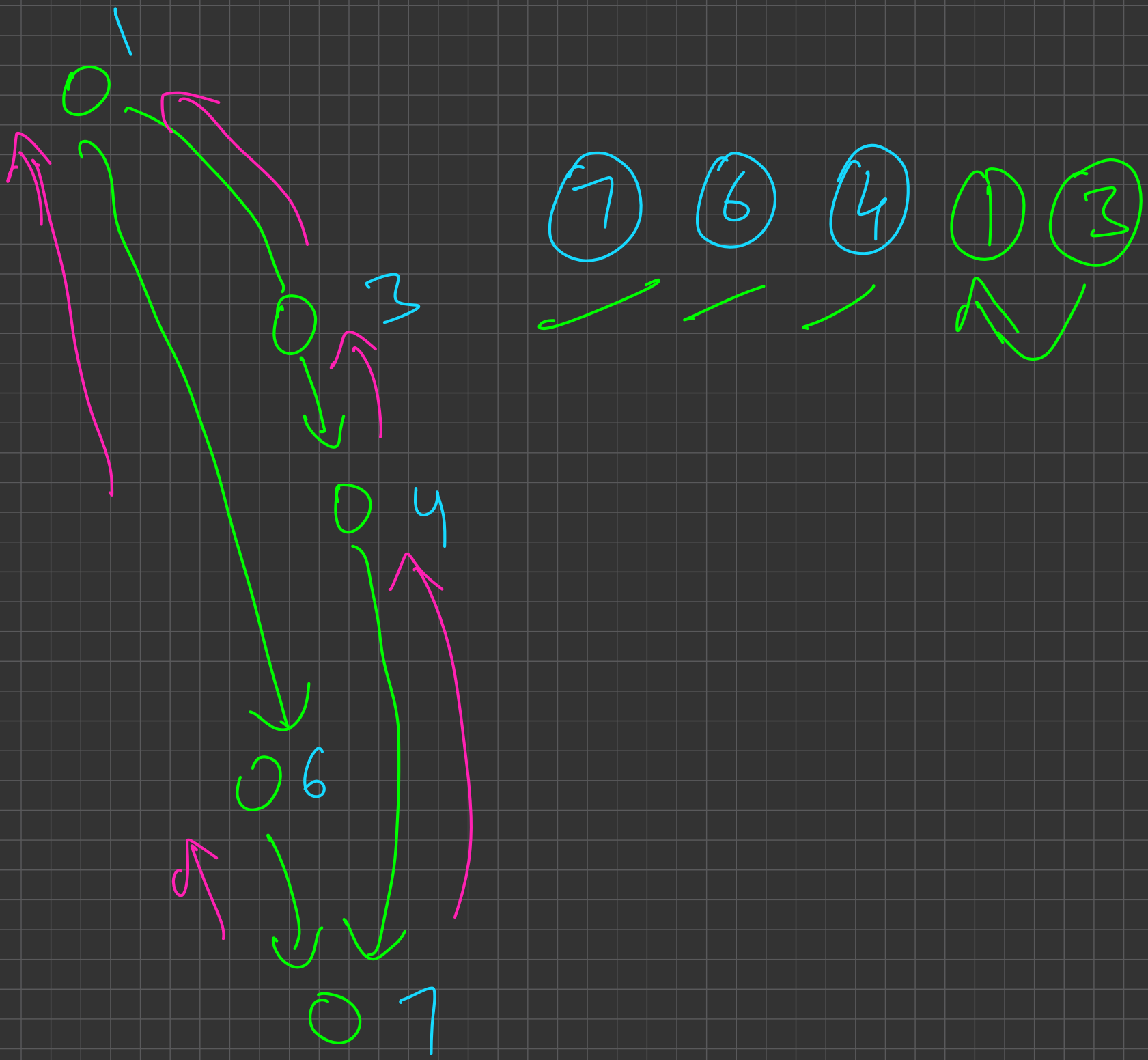
O(n+m)

Directed Acyclic Graph

DAG

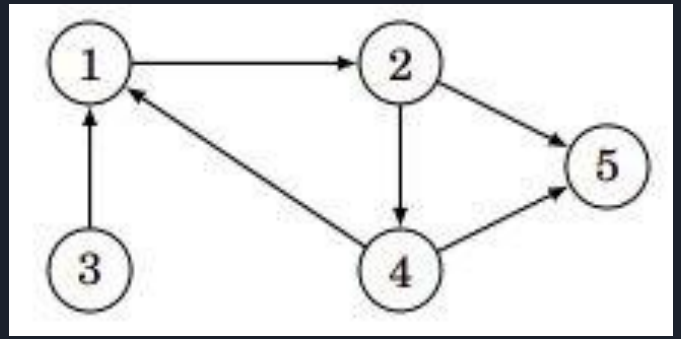

0
10
1
2
3
4
5
6
7
8
9
0 7 10 1 2 3 4 5 6 8 9 7


1
2
3
4
5


1
0
0
3
0
4
0
6
0
7
7
6
4
1
3

Directed Graph

A graph is directed if the edges can be traversed in one direction only. For example, the following graph is directed:



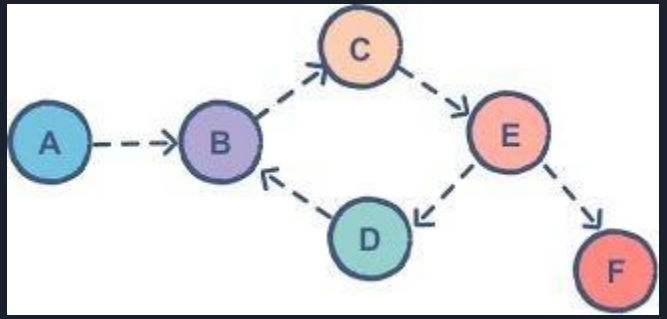
The above graph contains a path $3 \rightarrow 1 \rightarrow 2 \rightarrow 5$ from node 3 to node 5, but there is no path from node 5 to node 3.

Directed Graph

In a directed graph:

The **indegree** of a vertex V is the number of edges coming into vertex V.

The **outdegree** of a vertex V is the number of edges going out from vertex V.

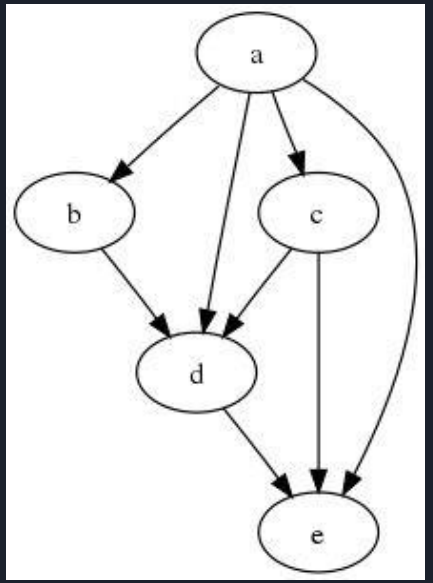


Here, the indegree of node B is 2 and outdegree of node B is 1.

Directed Acyclic Graph

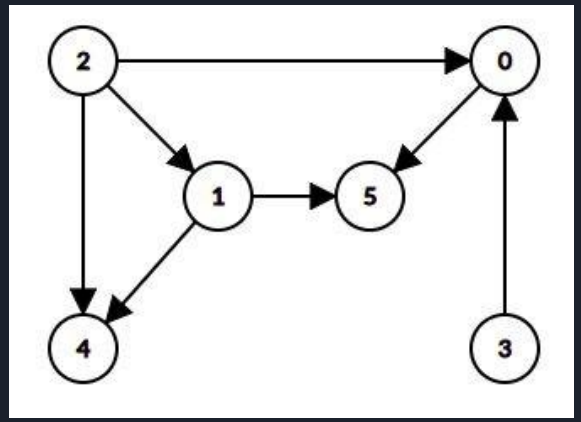
Acyclic graphs: A graph is acyclic if there are no cycles in the graph, so there is no path from any node to itself.

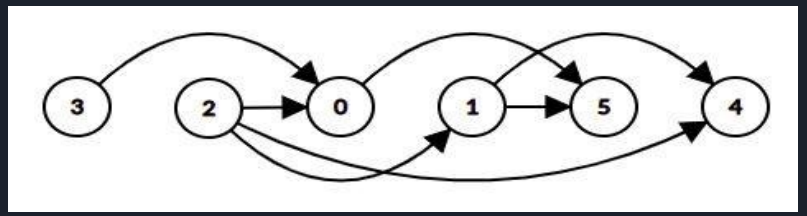
DAG: A directed acyclic graph (DAG) is simply a directed graph which contains no cycles. That is, it consists of vertices and edges, with each edge directed from one vertex to another, such that following those directions will never form a closed loop.

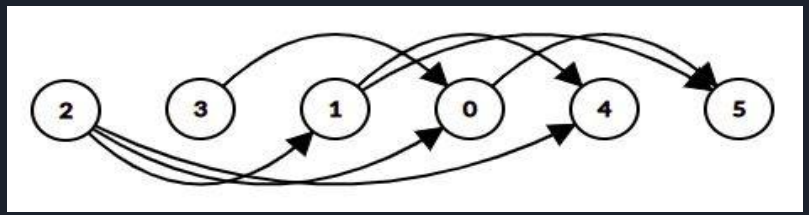


Topological Sort

A topological sort is an ordering of the nodes of a directed graph such that if there is a path from node a to node b, then node a appears before node b in the ordering.







Topological sort of the graph is [3, 2, 0, 1, 5, 4] or [2, 3, 1, 0, 4, 5]

Topological Sort

An acyclic graph always has a topological sort. If the graph contains a cycle, it is not possible to form a topological sort, because no node of the cycle can appear before the other nodes of the cycle in the ordering

Topological Sort

In a topological sorting of a DAG, which node comes first?

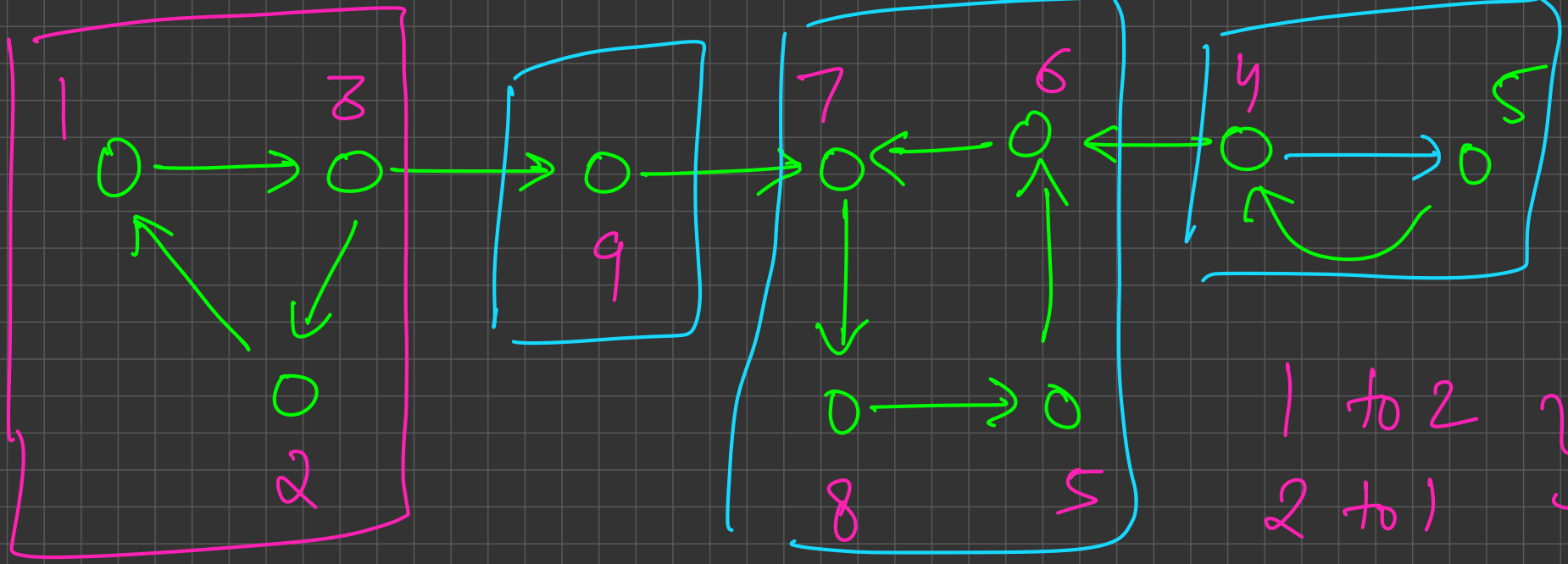
- The first node in a topological sorting should have the indegree equal to 0.

The topological sorting algorithm:

1. Find a node with indegree equal to 0 and position it the first in the topological sorting.
2. Remove that node from the graph.
3. The remaining part of the graph is also a DAG, so go back to step 1.

Topological Sort

We maintain a queue like bfs and an array which contains the indegree of each node. The nodes with indegree 0 is added into the queue. We take the nodes one by one from the queue and add it to the sorted list. When a node is added to the sorted list, the indegree of each adjacent node gets decreased by 1.



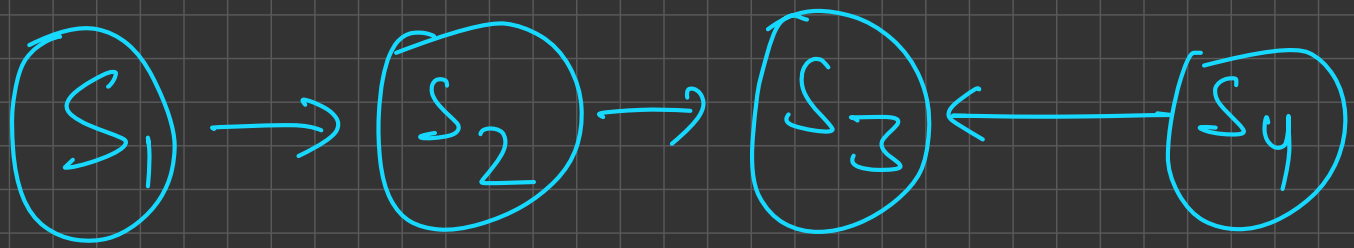
1 to 2 ✓

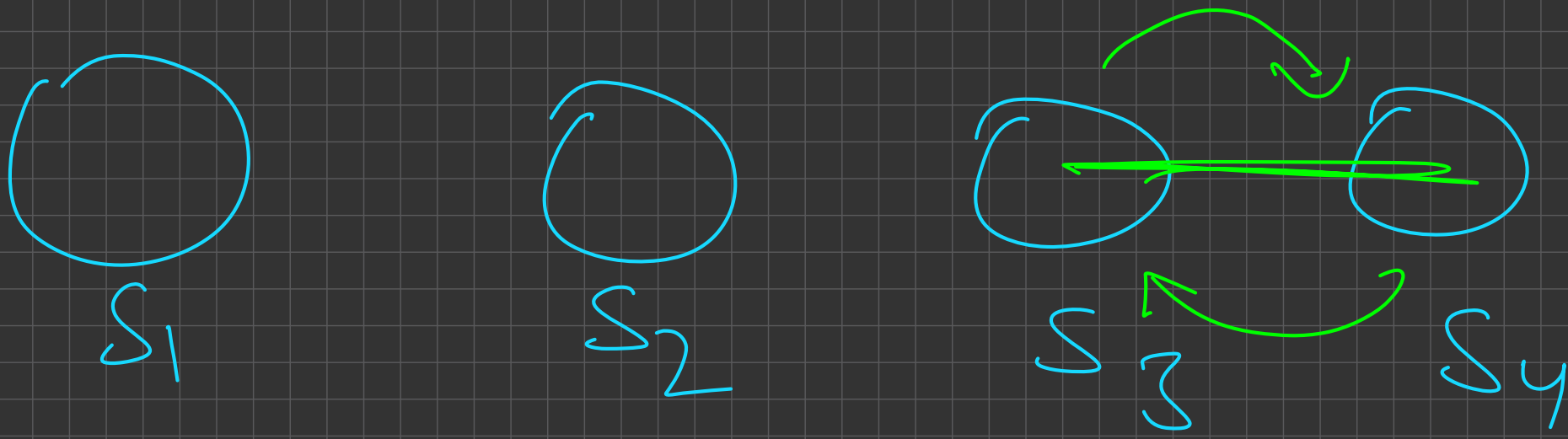
2 to 1 ✓

x to y

1 to 7 ✓

7 to 1 ✓

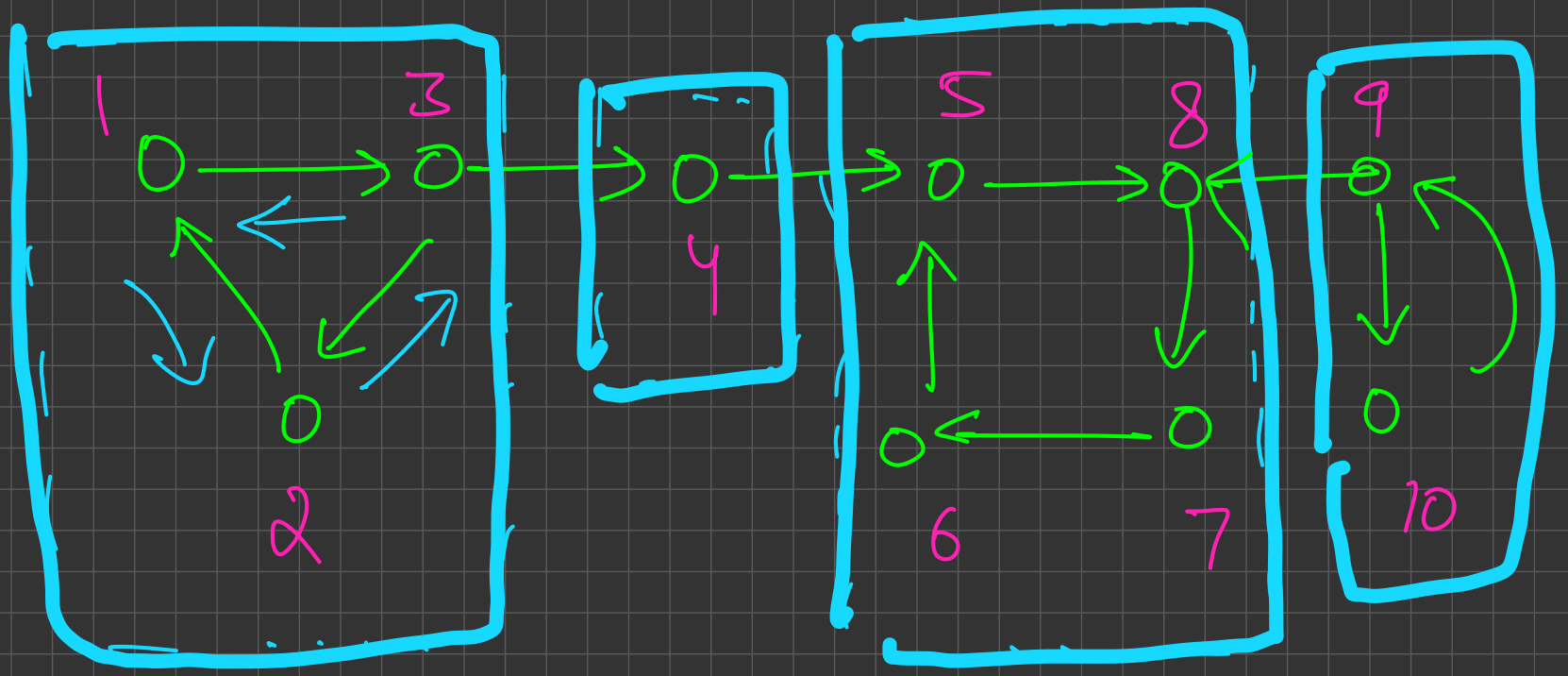



S_1
S_2
S_3
S_4

Separating out the graph into strongly connected components results in a DAG

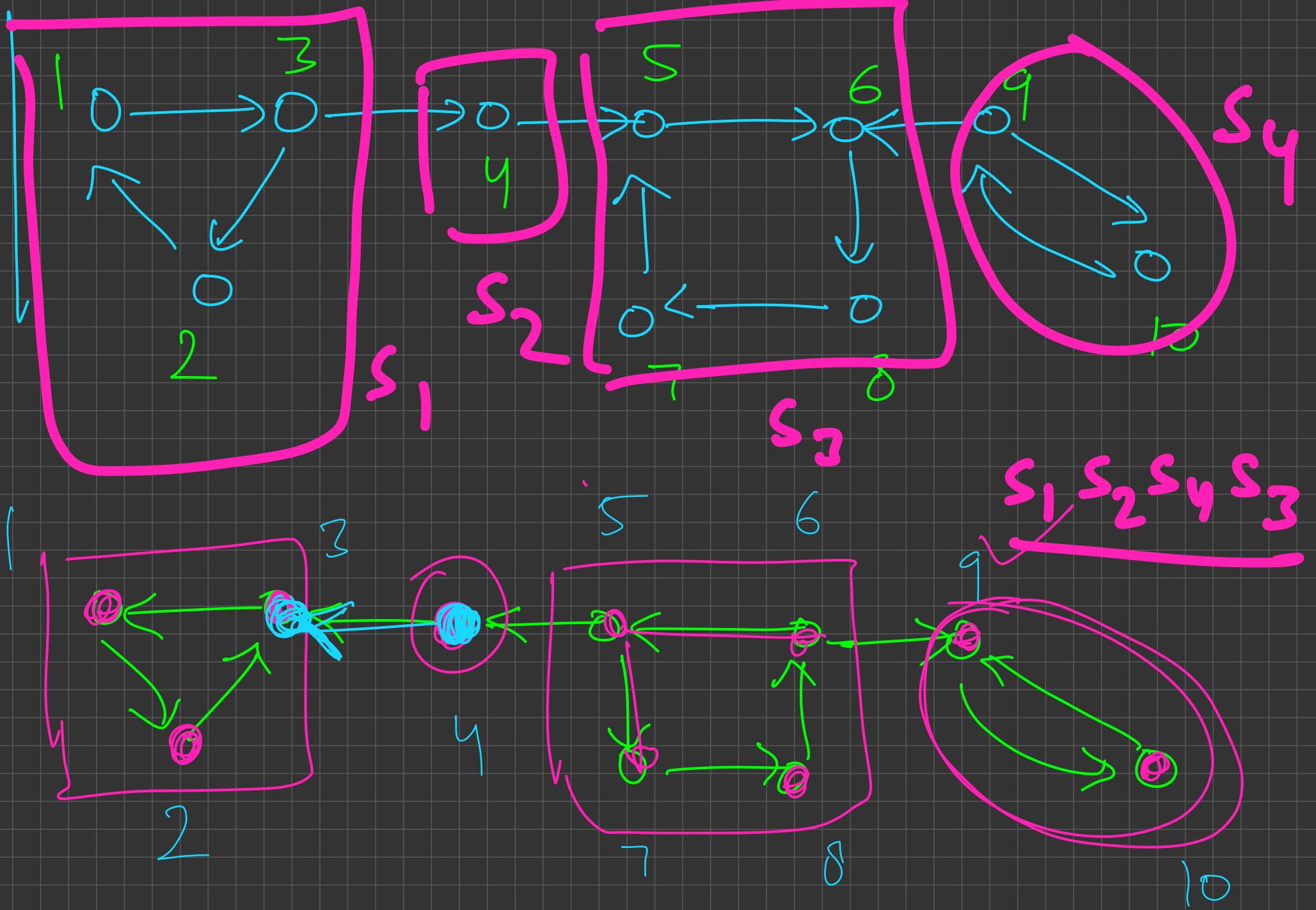
Connected ✗

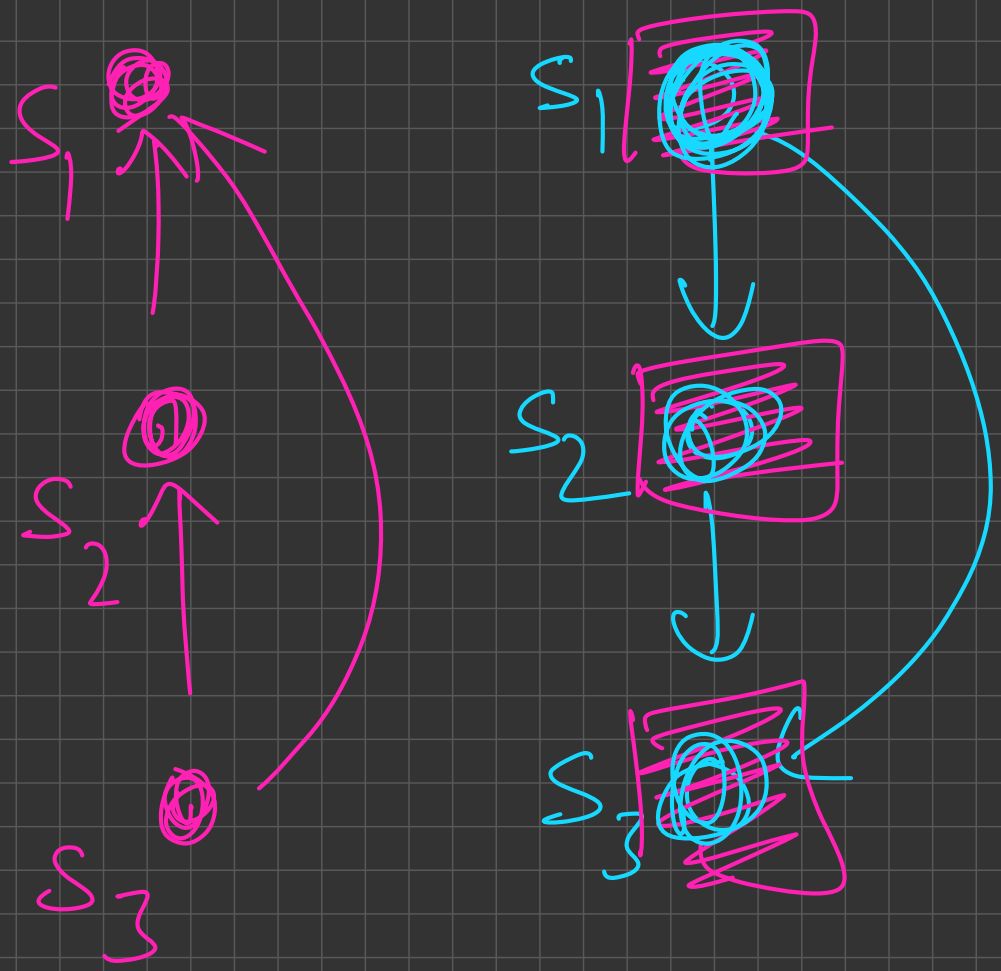
Strongly connected : x & y are strongly connected if there is a path from x to y & y to x



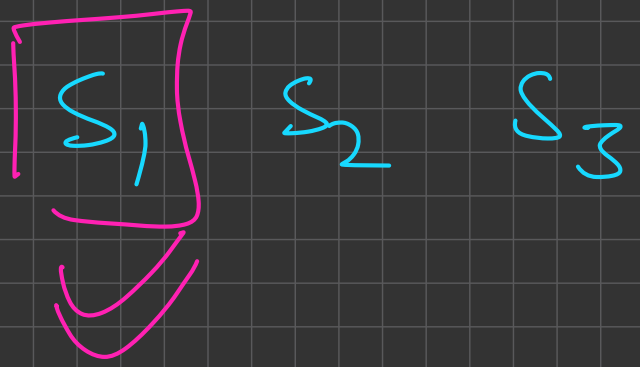
$$S_1 \rightarrow S_2 \rightarrow S_3 \leftarrow S_4$$

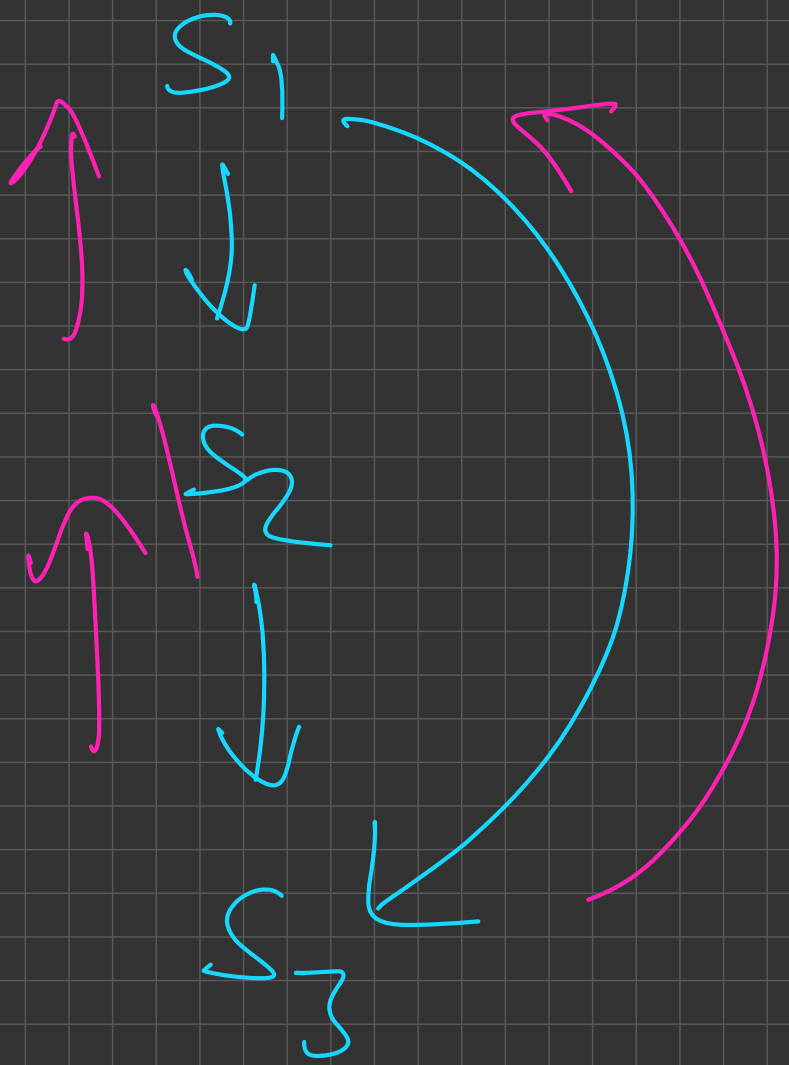
$$S_1 \leftarrow S_2 \leftarrow S_3 \rightarrow S_4$$


1
3
2
4
5
6
9
10
7
8
S_1
S_2
S_3
S_4
$S_1 S_2 S_4 S_3$

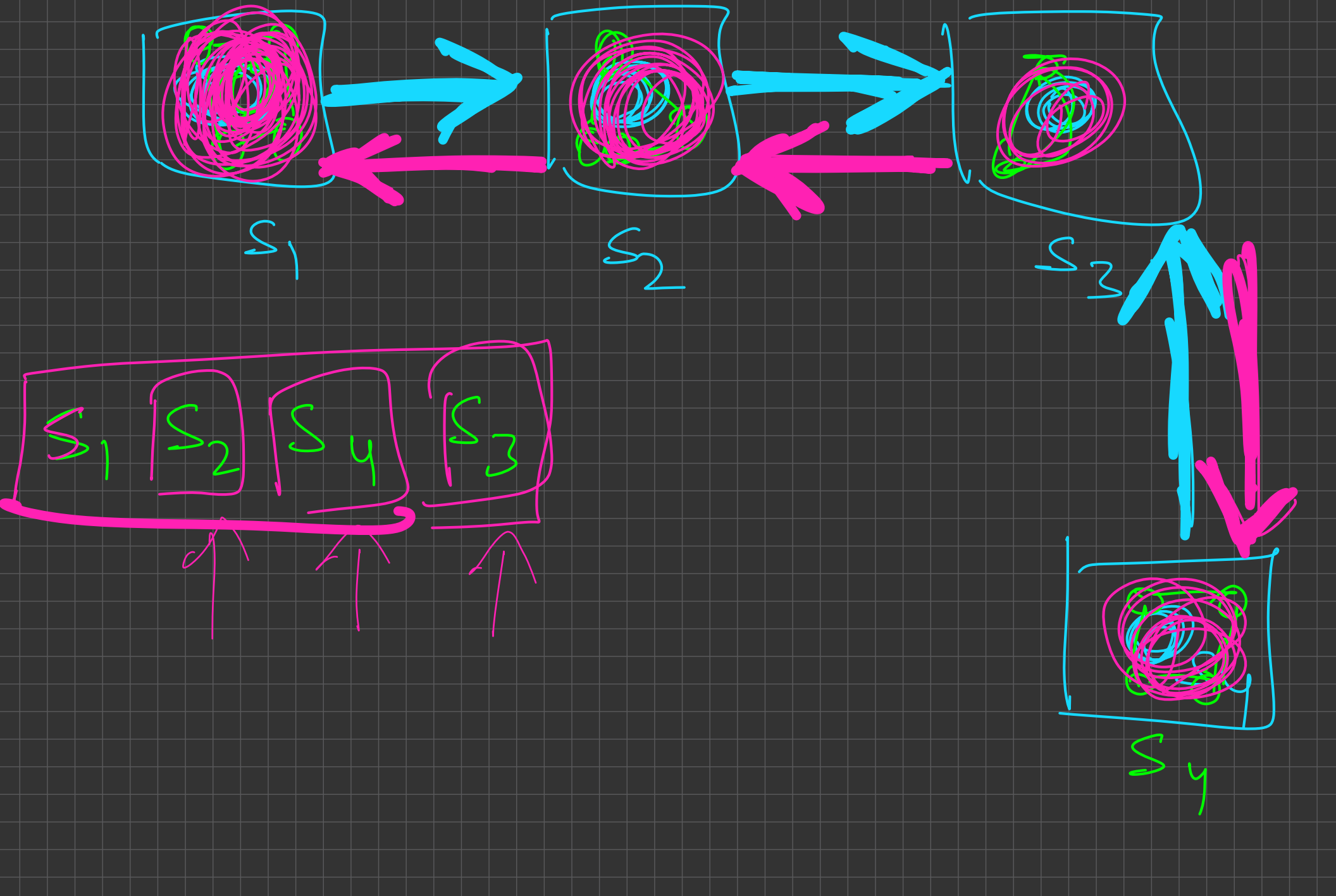

S_1
S_2
S_3

S_1
S_2
S_3


S_1
S_2
S_3

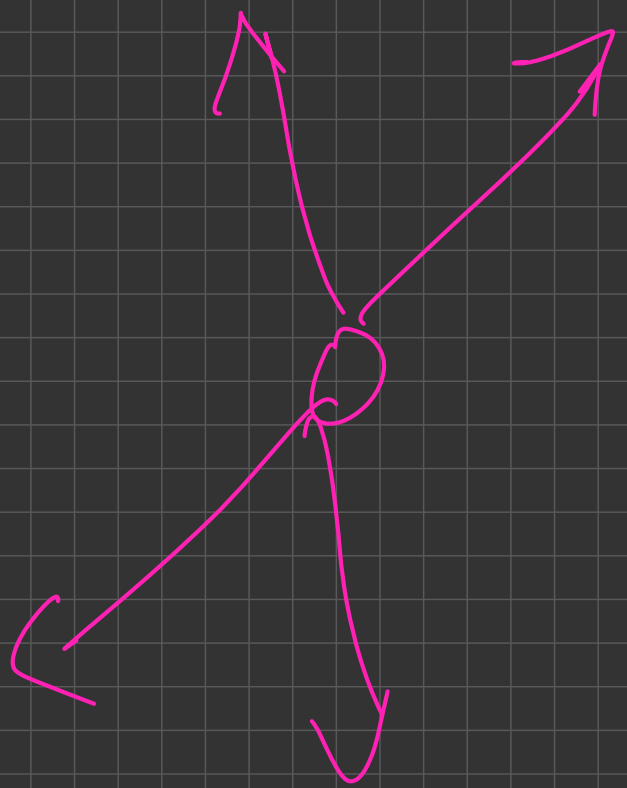


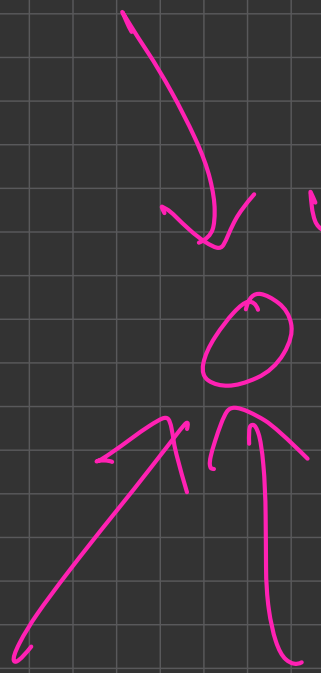
S_1 S_2 S_3

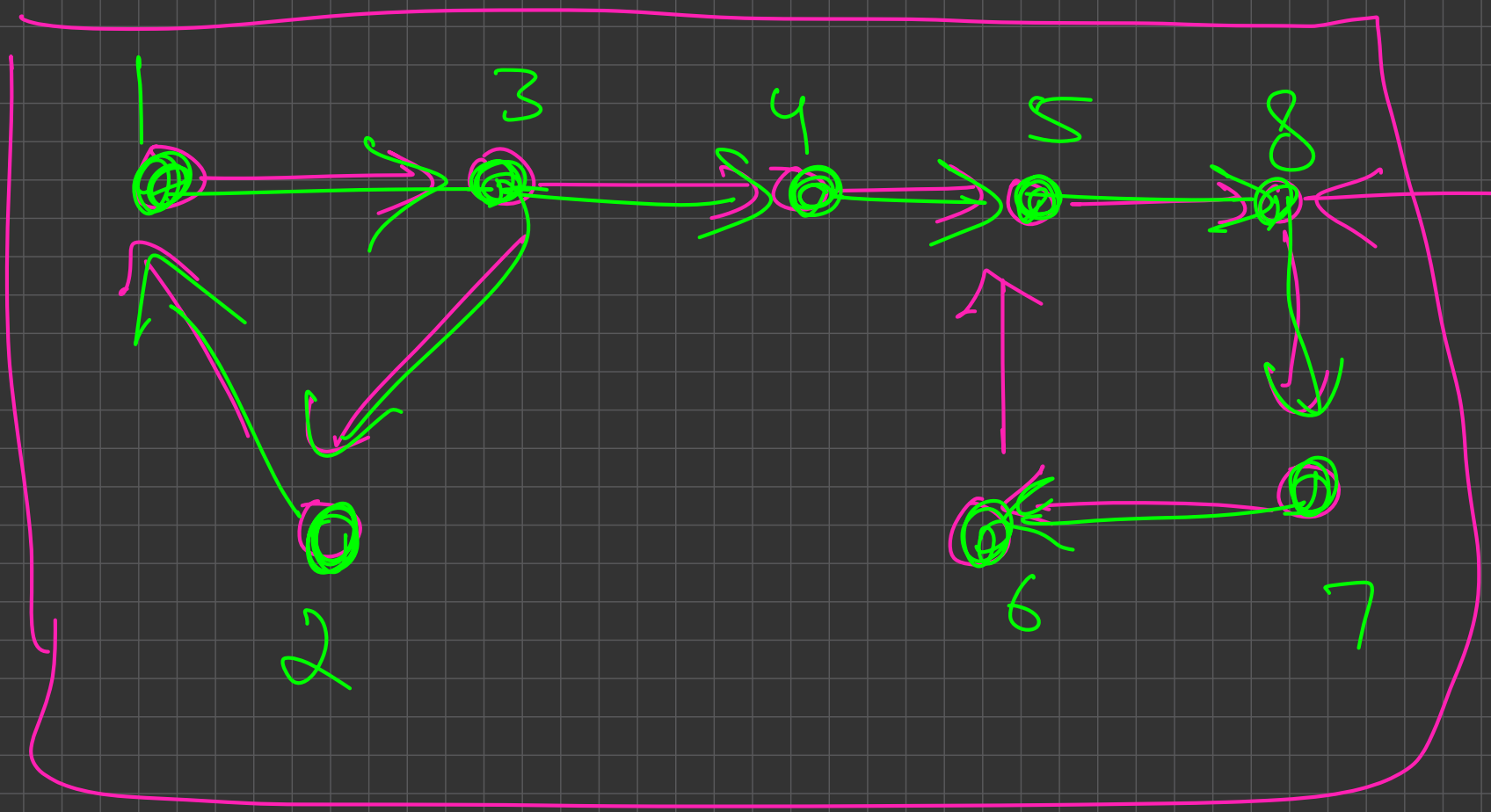

S1
S2
S3
S1
S2
S4
S3
S4

$S_1 \longrightarrow S_2$

$S_1 \longrightarrow S_2$ with reversed edges



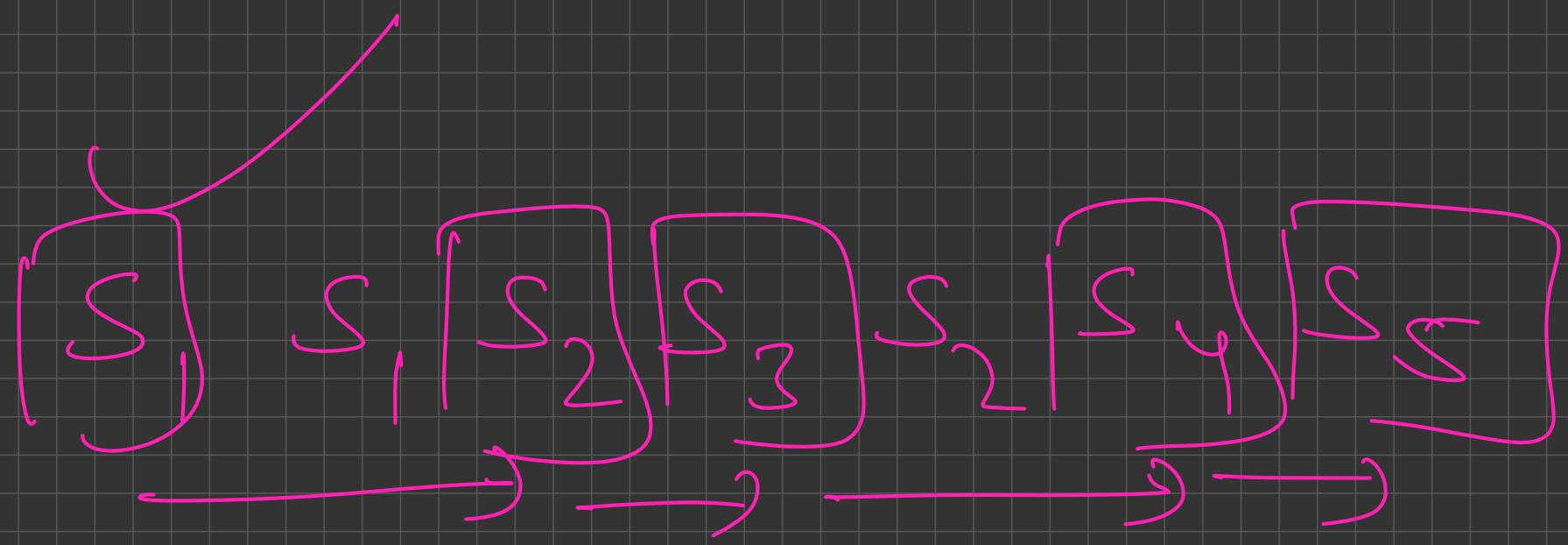




9 0 9 5

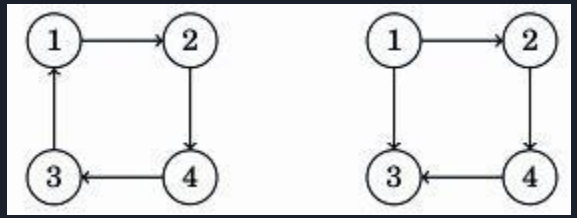
DP on Graphs

Topo Sort → DAG



Strongly Connected Component

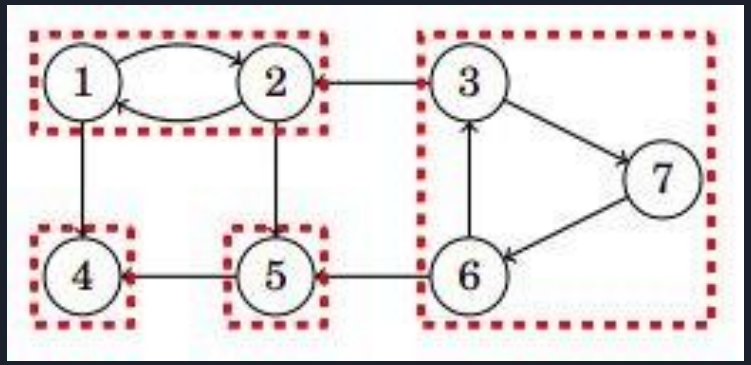
A graph is strongly connected if there is a path from any node to all other nodes in the graph.



The left graph is strongly connected but the right graph is not.

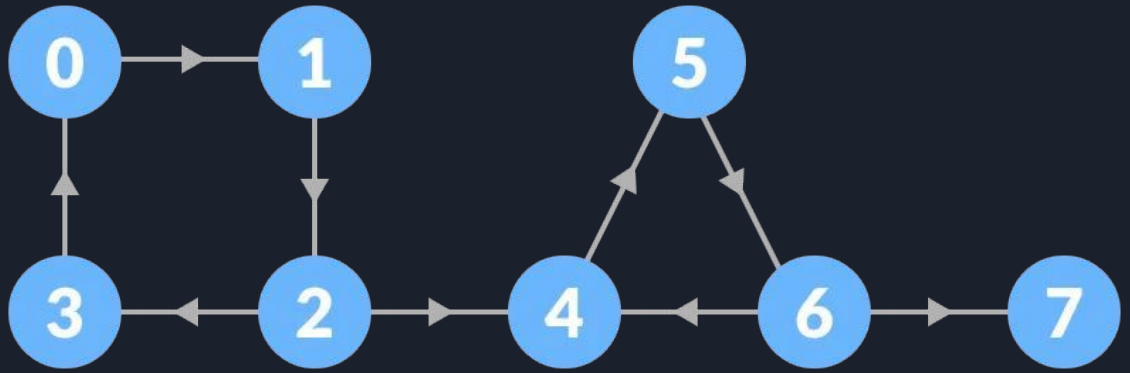
Strongly Connected Component

The strongly connected components of a graph divide the graph into strongly connected parts that are as large as possible.

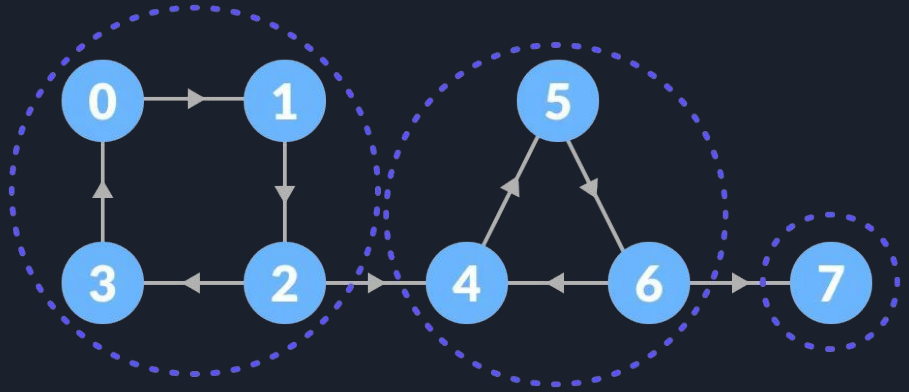


Kosaraju’s Algorithm

Given
Graph:



Kosaraju’s Algorithm



Problem (SCC)

A game has n rooms and m tunnels between them. Each room has a certain number of coins. What is the maximum number of coins you can collect while moving through the tunnels when you can freely choose your starting and ending room?

Input:

The first input line has two integers n and m: the number of rooms and tunnels. The rooms are numbered 1,2,…,n.

Then, there are n integers k1,k2,…,kn: the number of coins in each room.

Finally, there are m lines describing the tunnels. Each line has two integers a and b: there is a tunnel from room a to room b. Each tunnel is a one-way tunnel.

Output:
Print one integer: the maximum number of coins you can

collect. Problem Source: CSES-1686