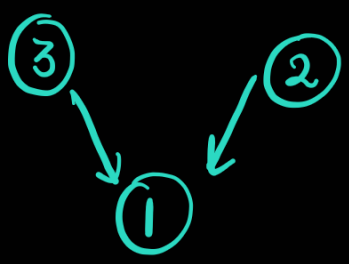
① Topological Labelling

directed



Node 3 2 1

label all nodes

label 1 3 2

2 3 1 unique no. [1 ... N]

Sequence 2 3 1

go in order of nodes 1 → N

print label

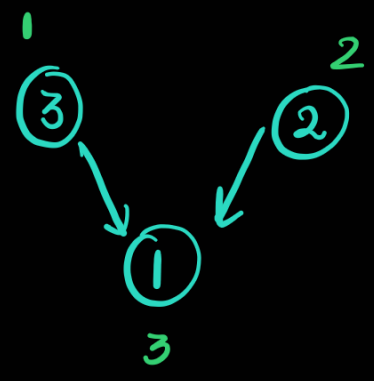
Condition

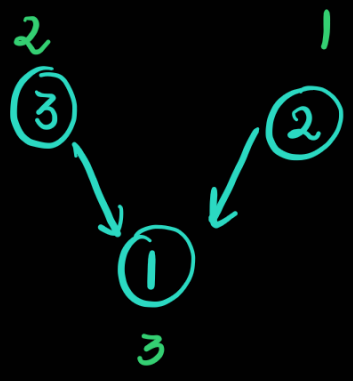
$v \longrightarrow u$

$lab[v] < lab[u]$

Node	3	2	1
Label	1	2	3

2 1 3





Sequence 3 2 1

3 1 2

lexicographically smallest sequence

Directed

Edge condition

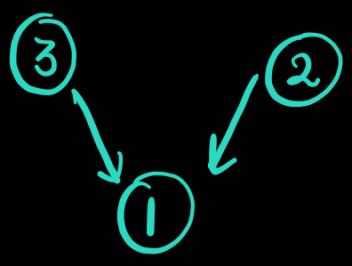
IF I write topological ordering

a b c d e f g

if $v \to u$ edge, always v comes

AlgoZenith

first, then u



3 2 1

2 3 1

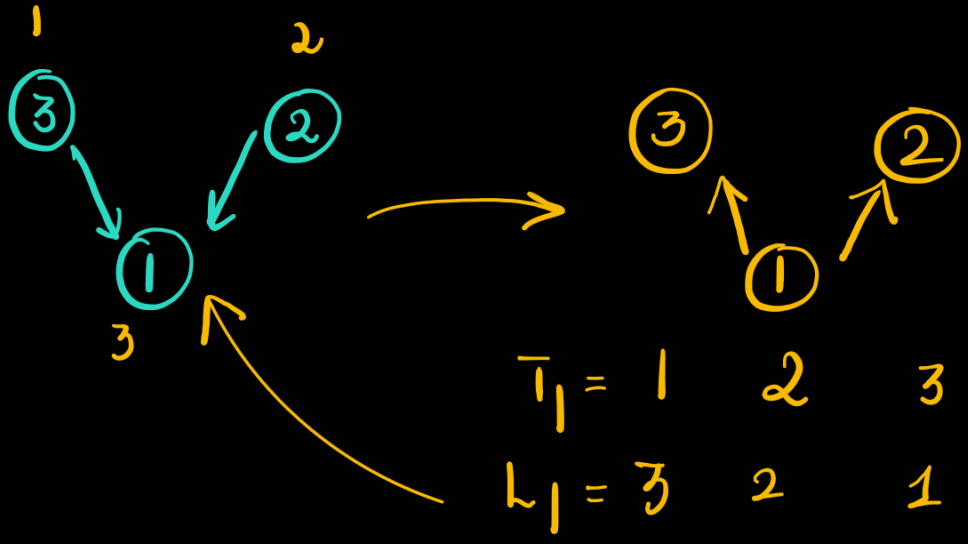
T → v _ _ _ _ u

Assign from [1 ... N] then lab[v] < lab[u]

v → u

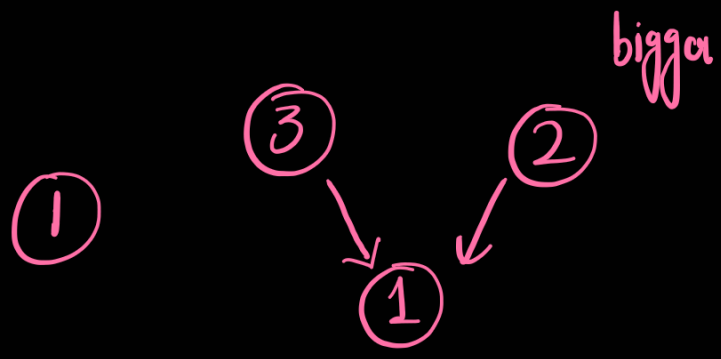
lab[v] < lab[u]

1) Top sort, label inc order

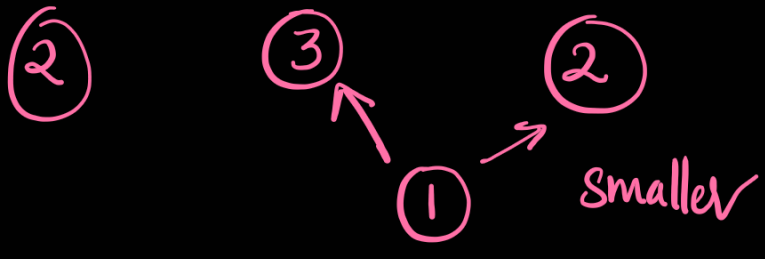


AlgoZenith

2) Reverse, Top sort, label dec order



T :	3	2	1	2	3	1
L :	1	2	3	1	2	3
S :	3	2	1	3	1	2



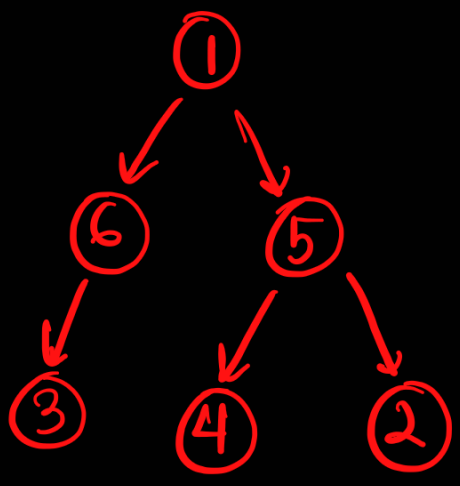
T :	1	2	3	1	3	2
L :	3	2	1	3	2	1
S :	3	2	1	3	1	2

No case other than these 4

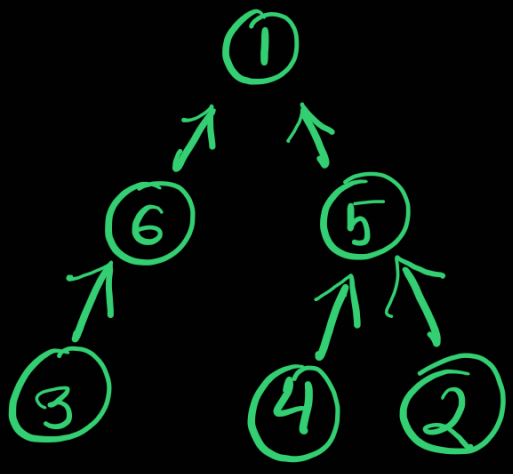
AlgoZenith

Can do Topo sort H times
but we want to do once

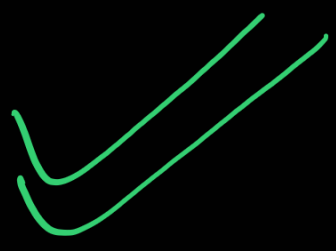
Topo → Smaller / Rev → Topo → Bigger



T	1	5	2	4	6	3
L	1	2	3	4	5	6
S	1	3	6	4	2	5

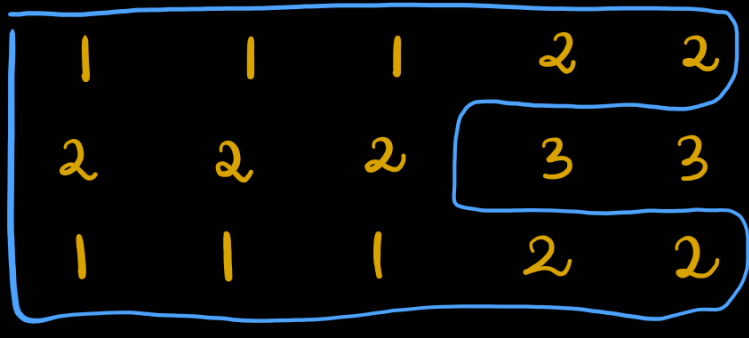


T	4	3	6	2	5	1
L	6	5	4	3	2	1
S	1	3	5	6	2	4



Winner :- reverse + topo sort (bigger)

(2) Two kingdoms



Max 2 distinct color

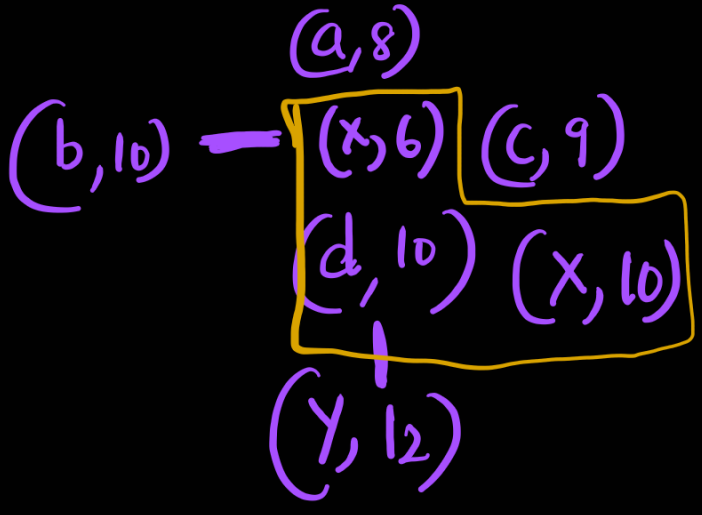
Maximize Component size

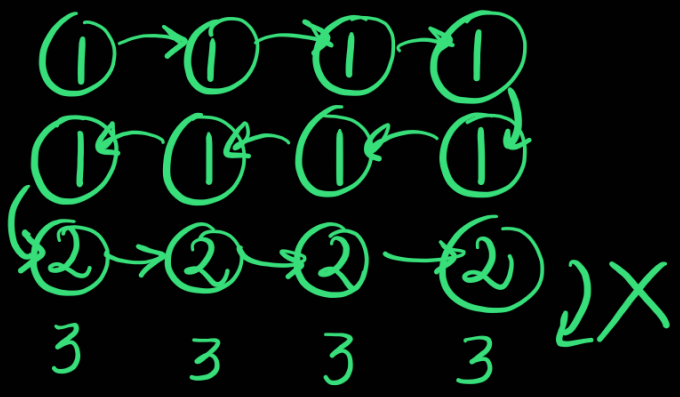
AlgoZenith

dfs on every cell

utmost 2 colors → in a set

$(n * n)^2$ complexity





1 1 1 1

1 1 1 1

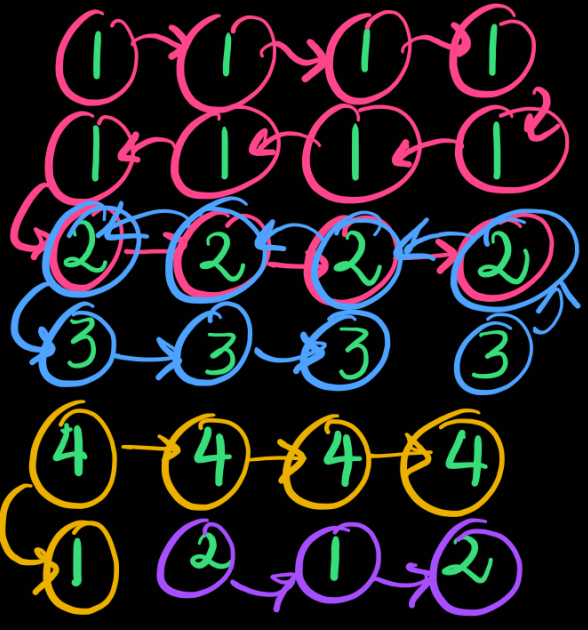
2 2 2 2

3 3 3 3

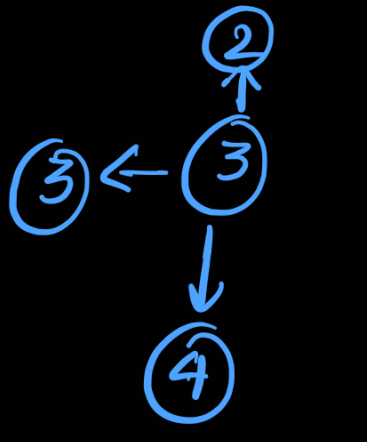
AlgoZenith

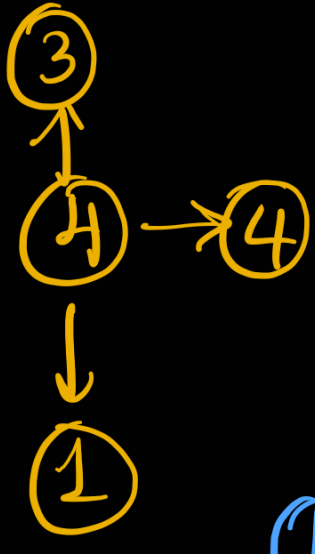
color visit {1, 2} both fail
node visit

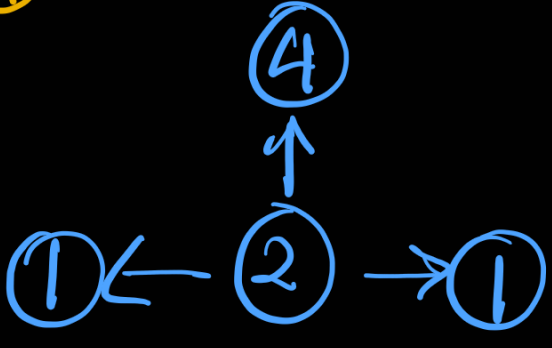
EDGE visit



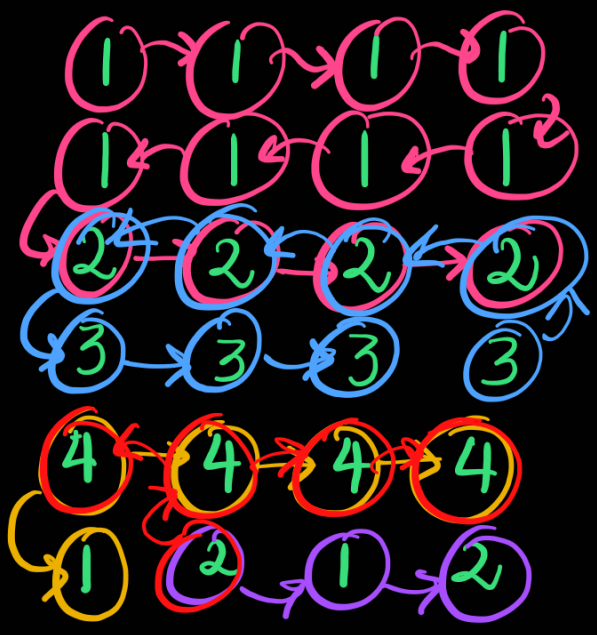
3 options



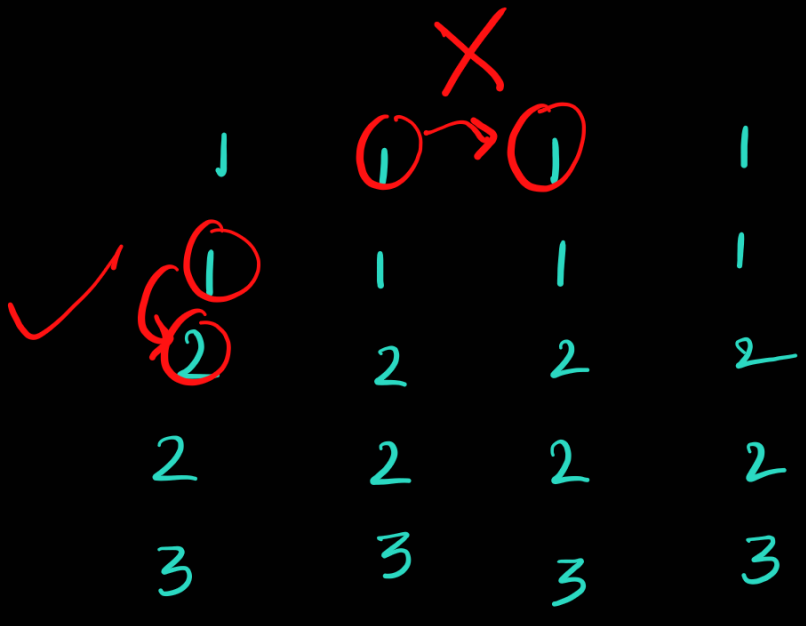




AlgoZenith

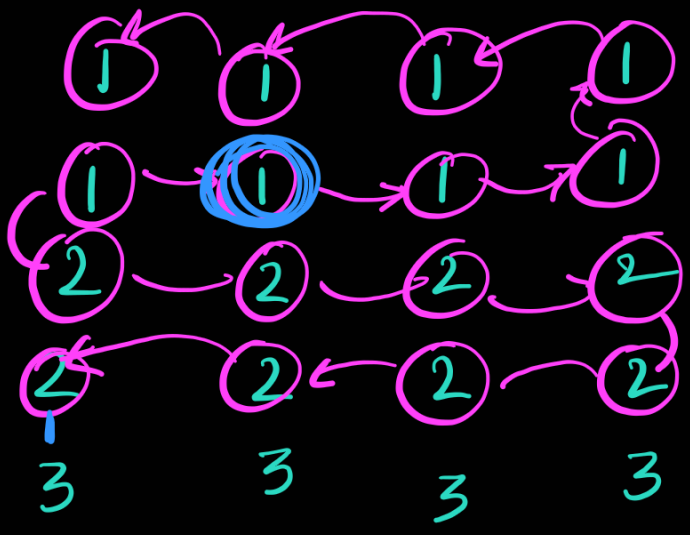


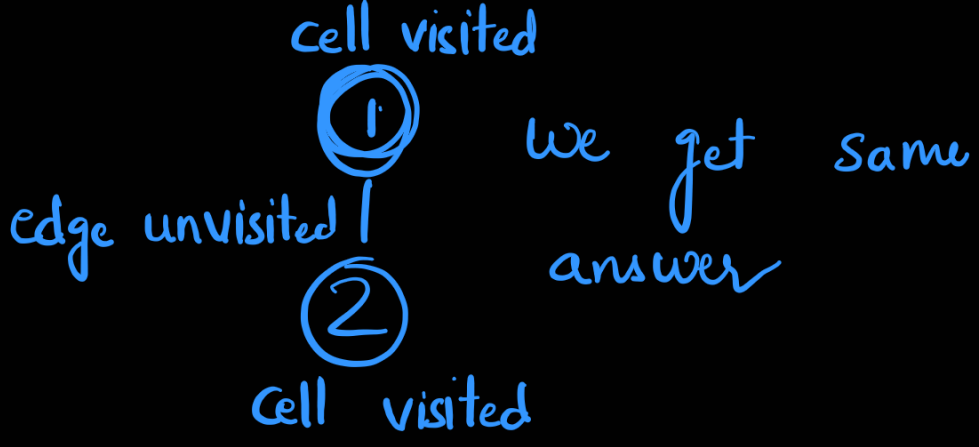
DFS on unvisited edge



Whenever DFS, start it only when there is connection

AlgoZenith





① x — y

② either x or y is not visited yet

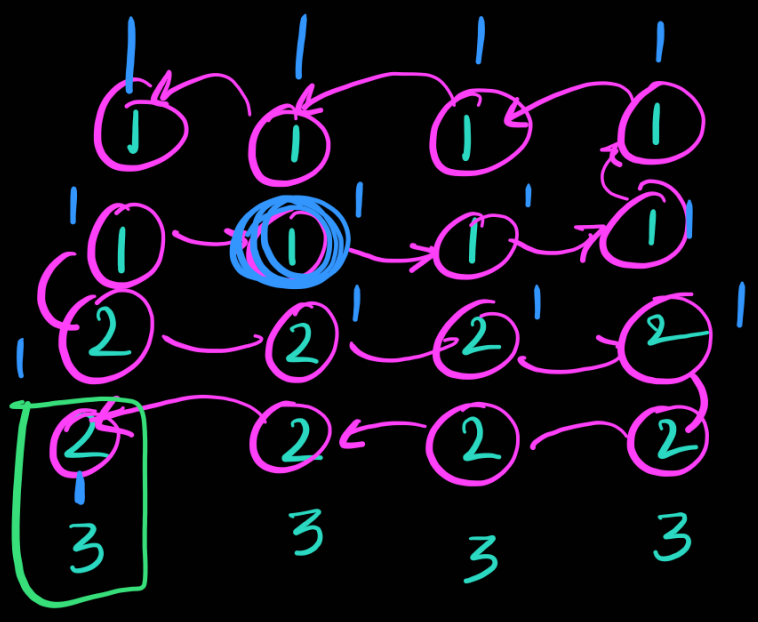
AlgoZenith

But issue is 2 → 3
we don't know 2's size

vis = 0
flag = 0
↳ increments whenever dfs calls

vis[i][j] < flag means unvisited

flag = 1

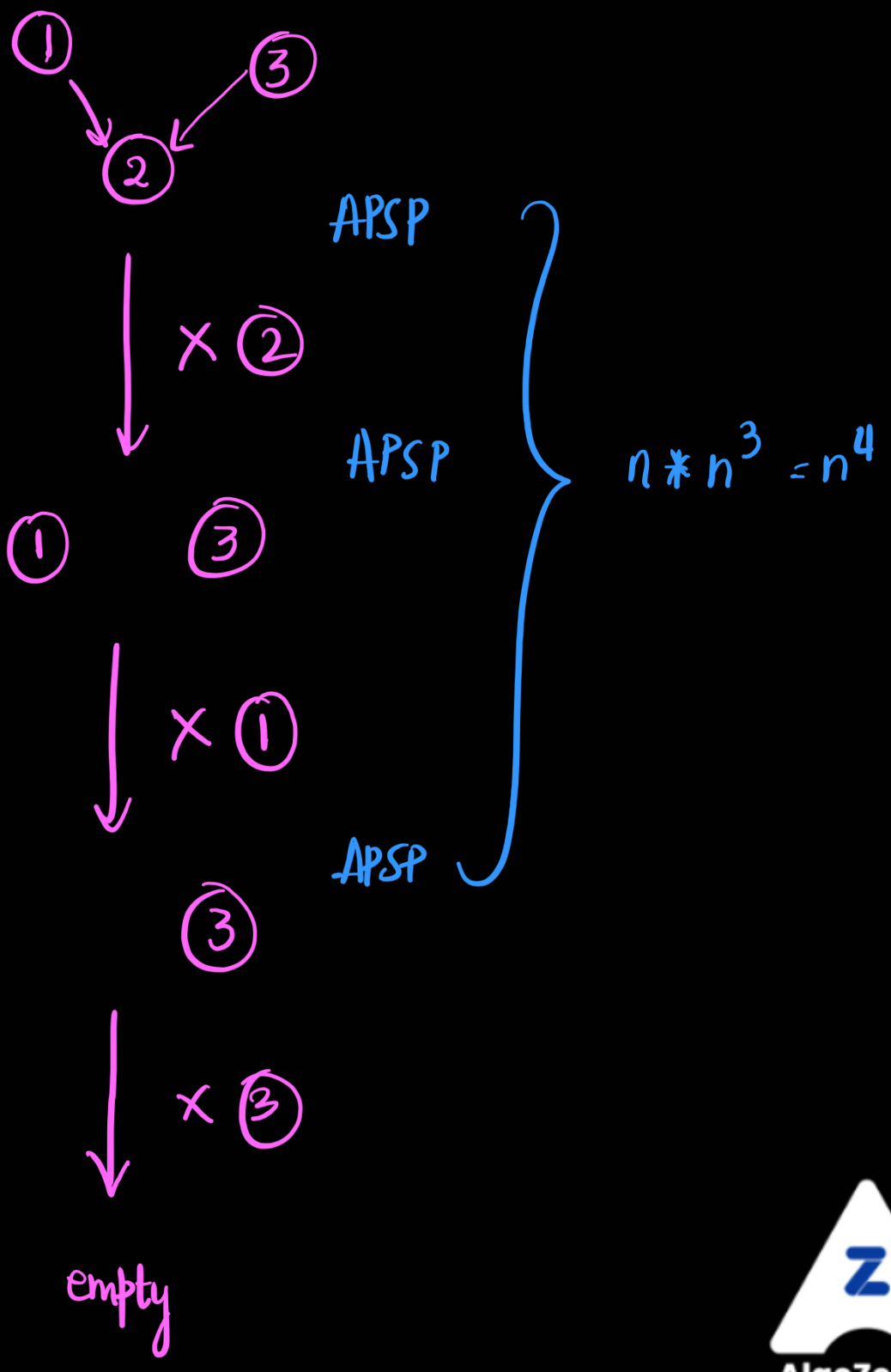


flag = 2
Now 1 < 2 so 2 is considered unvisited

AlgoZenith

③ All Pair Shortest Path

directed weighted

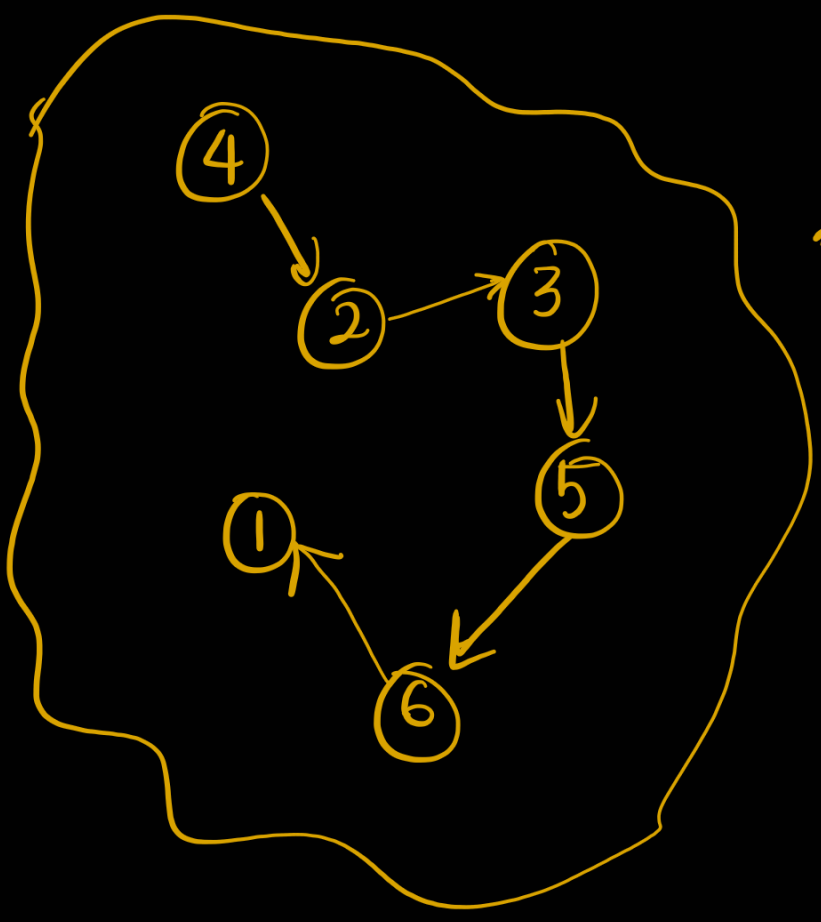


AlgoZenith

```
for  k = 1 → n
    for  i = 1 → n
        for  j = 1 → n
```

$$d[i][j] = \min(d[i][j],\ d[i][k] + d[k][j])$$

either direct OR middle man

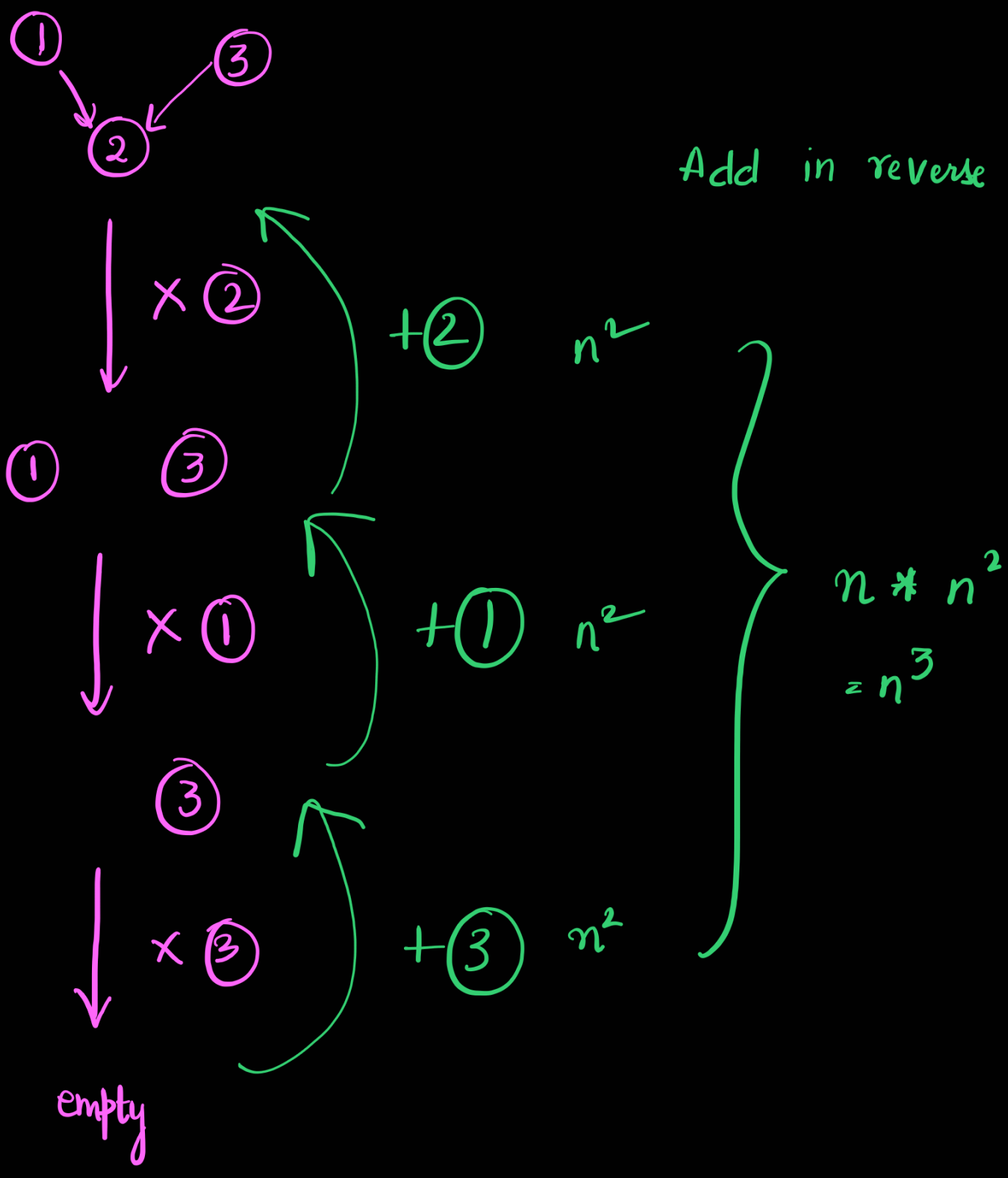


connection of x is given

```
for  i = 1 to 6
    for  j = 1 to 6
        d[i][j] = min(d[i][j], d[i][x] + d[x][j])
```

Algozenith

d[x][j])



AlgoZenith