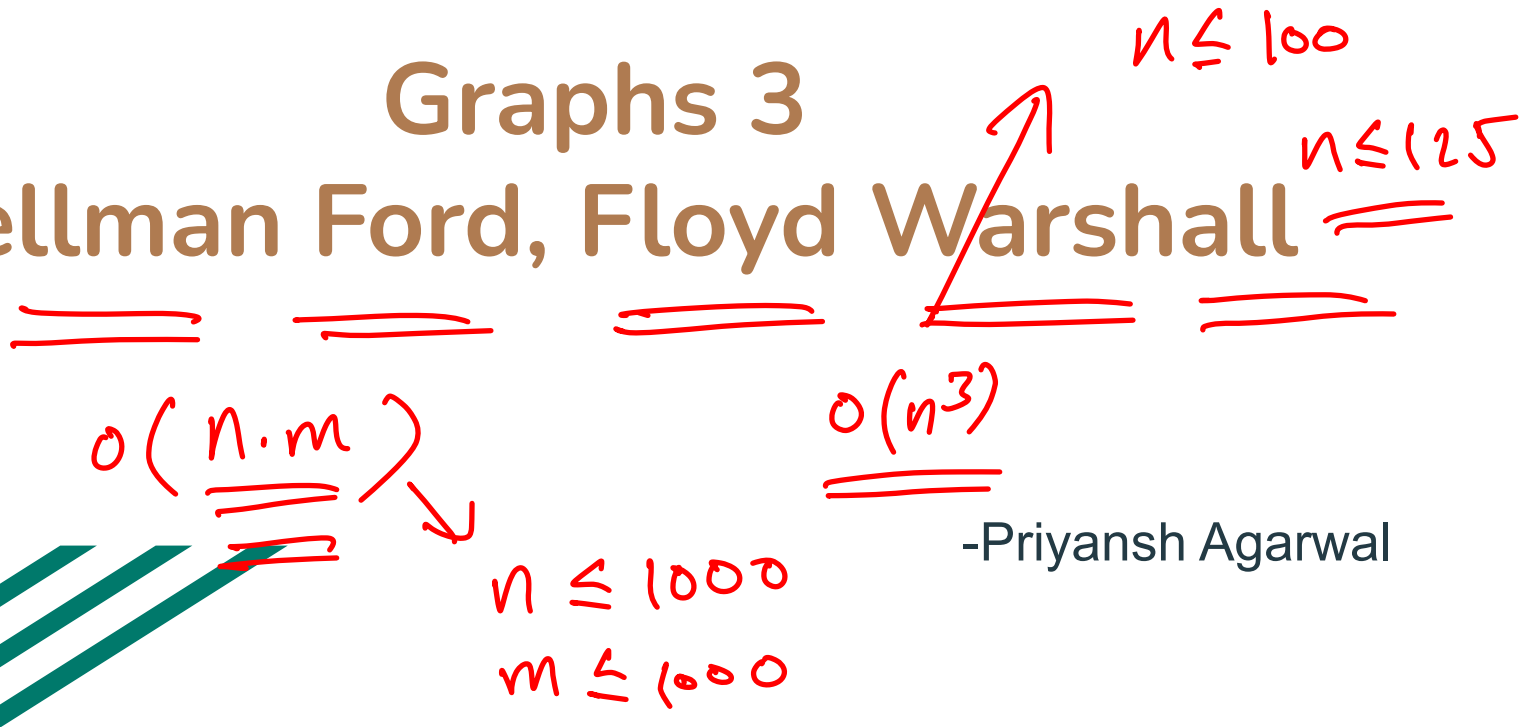
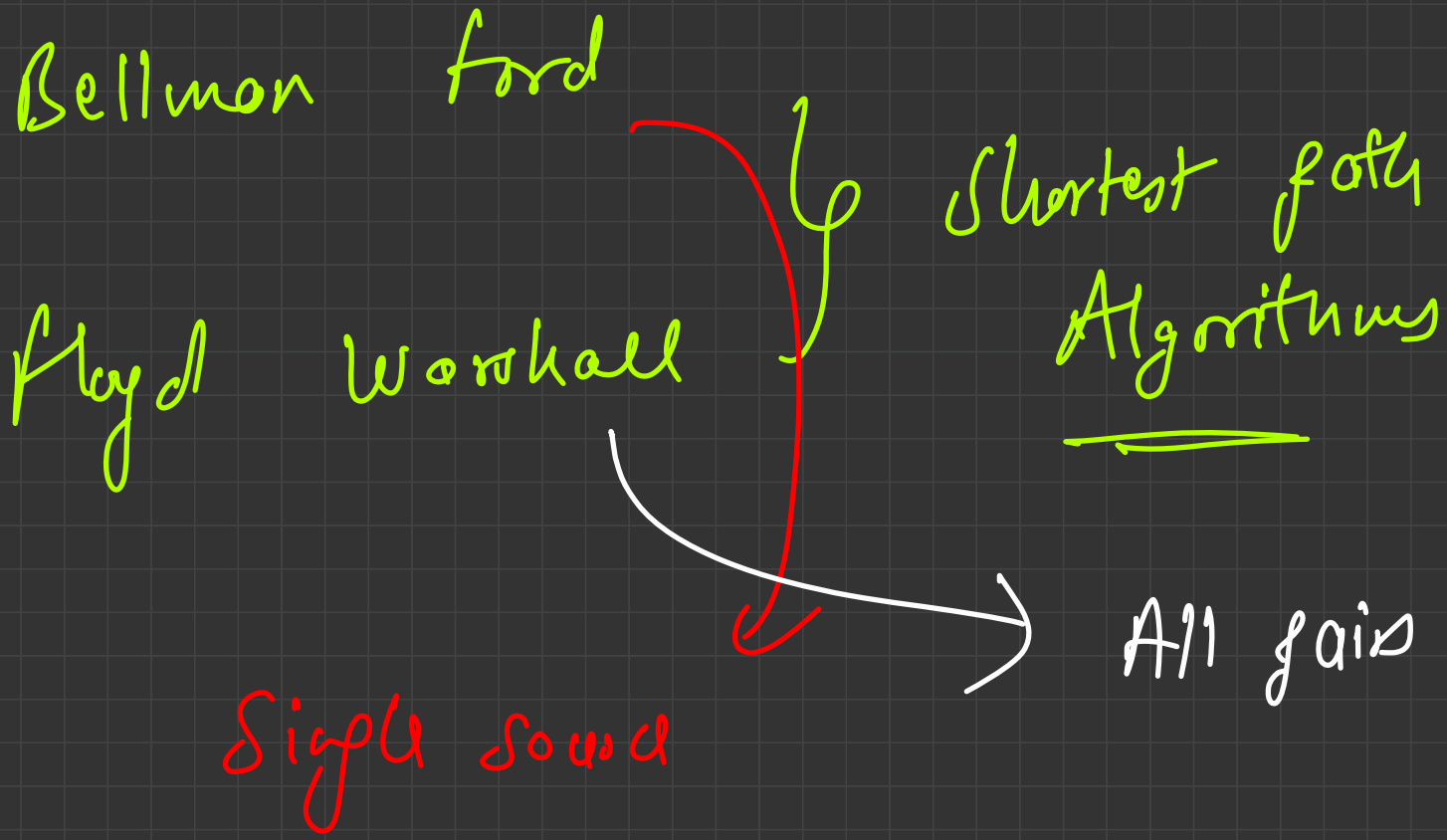
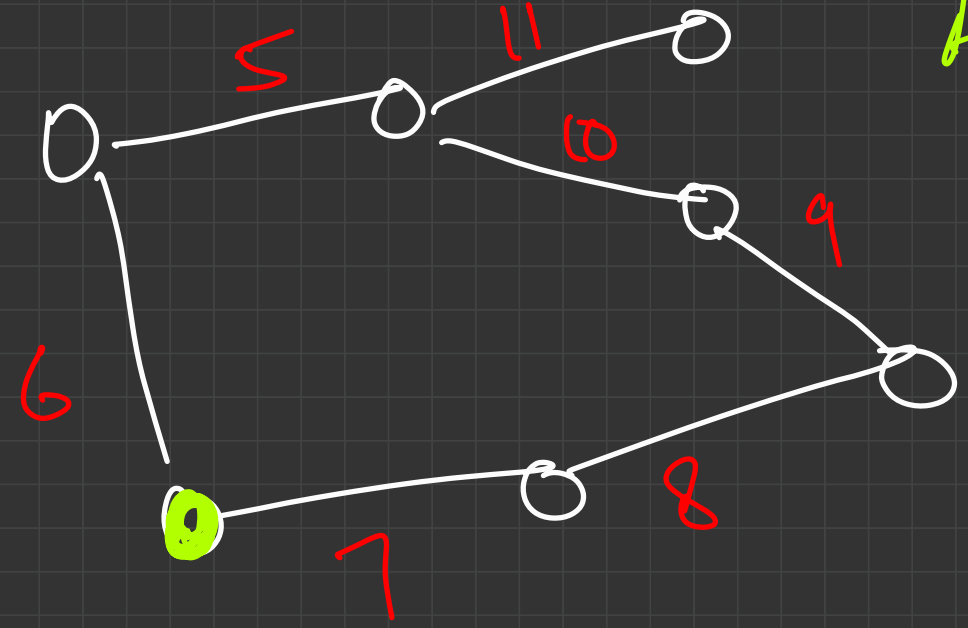
Graphs 3
Bellman Ford, Floyd Warshall



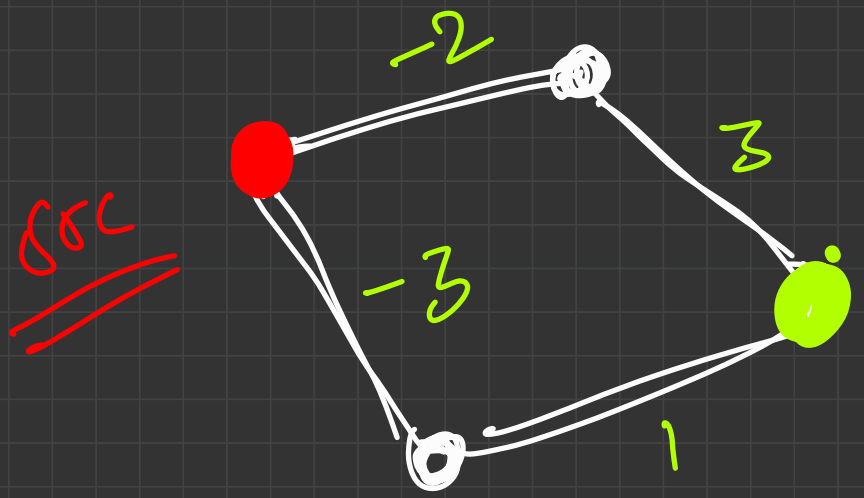
-Priyansh Agarwal


Bellman ford
Floyd Warshall
Shortest path Algorithms
All pair
Single source



Also supports negative edge weights

Dijkstra → all edge weights ≥ 0

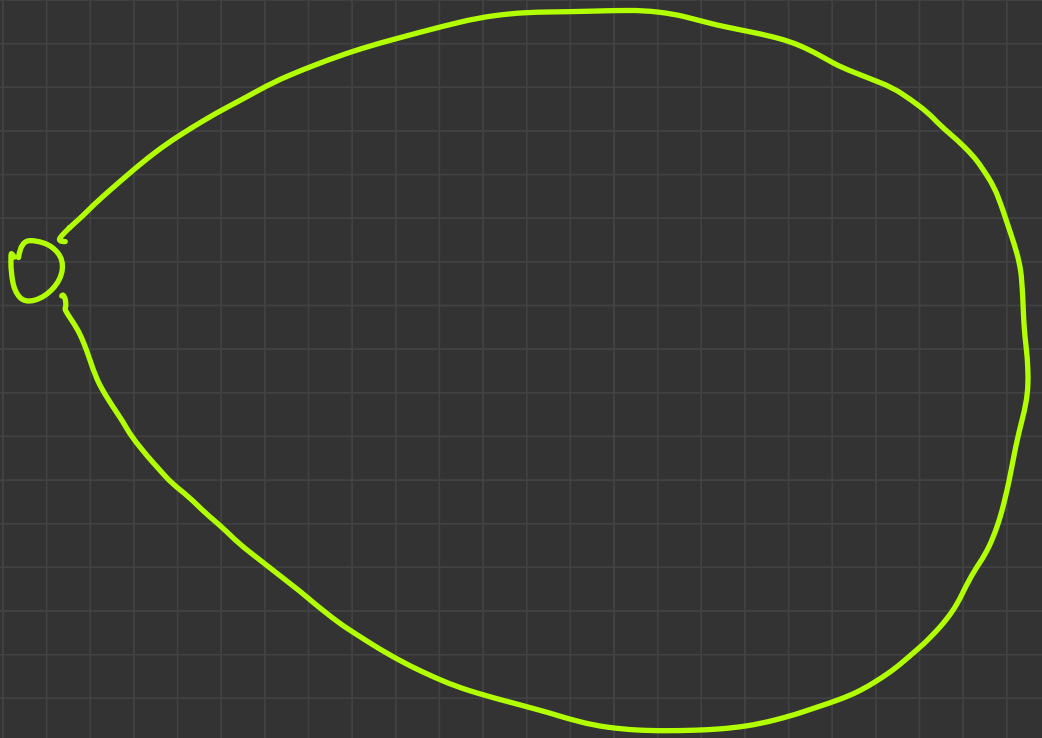


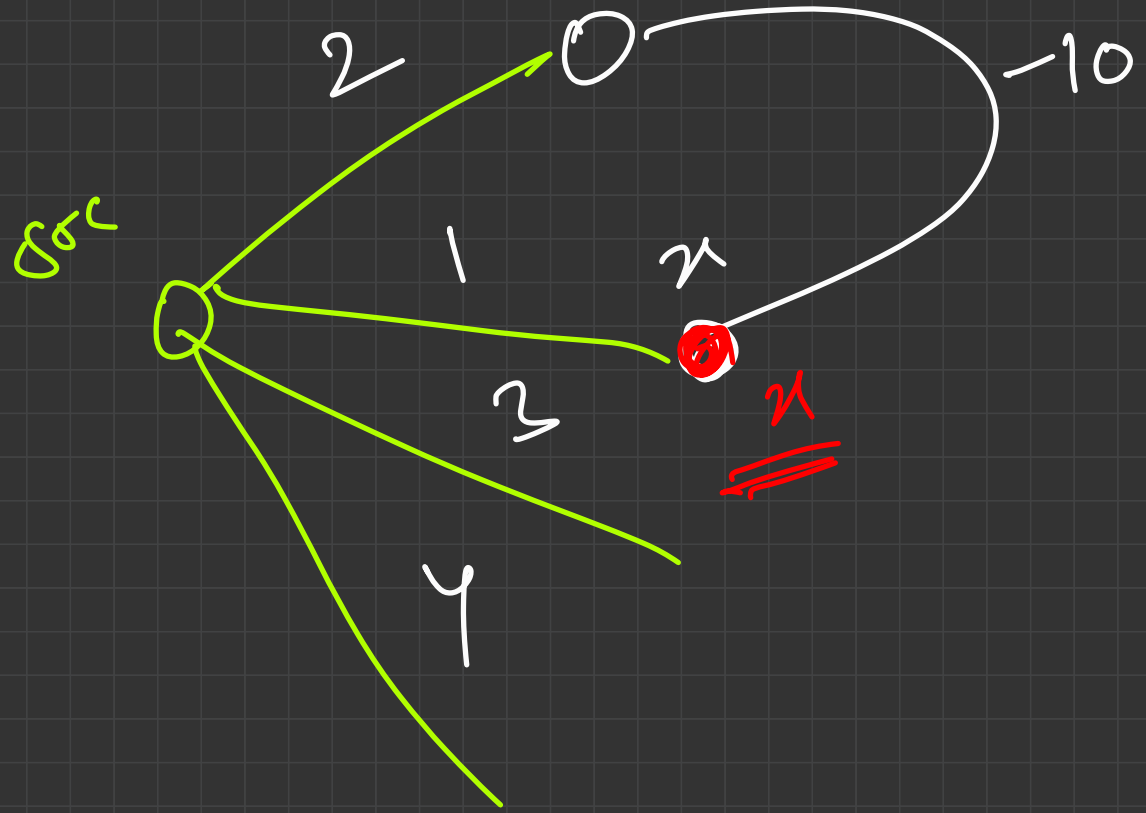
negative cycle

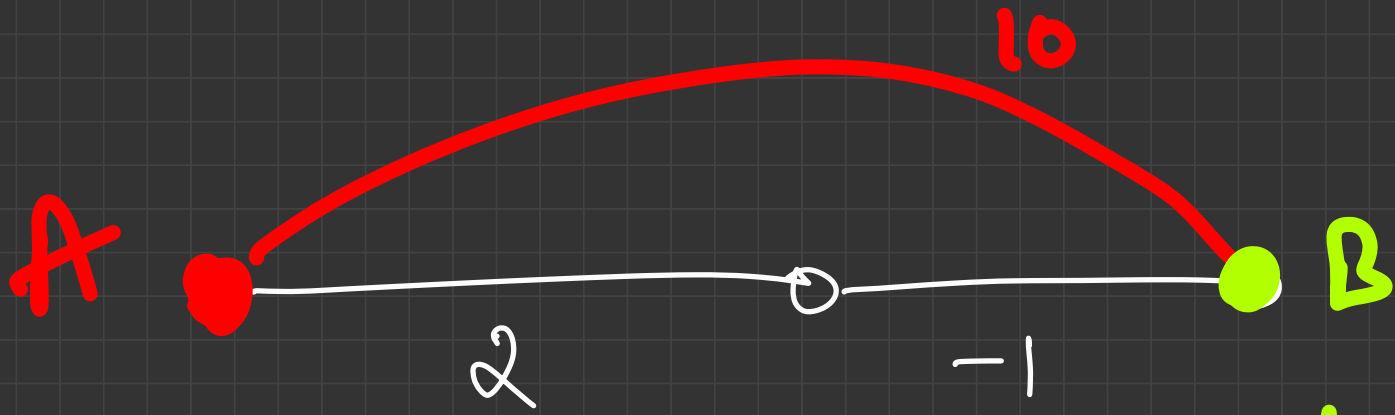
$-2 + 3 + 1 - 3$

-2

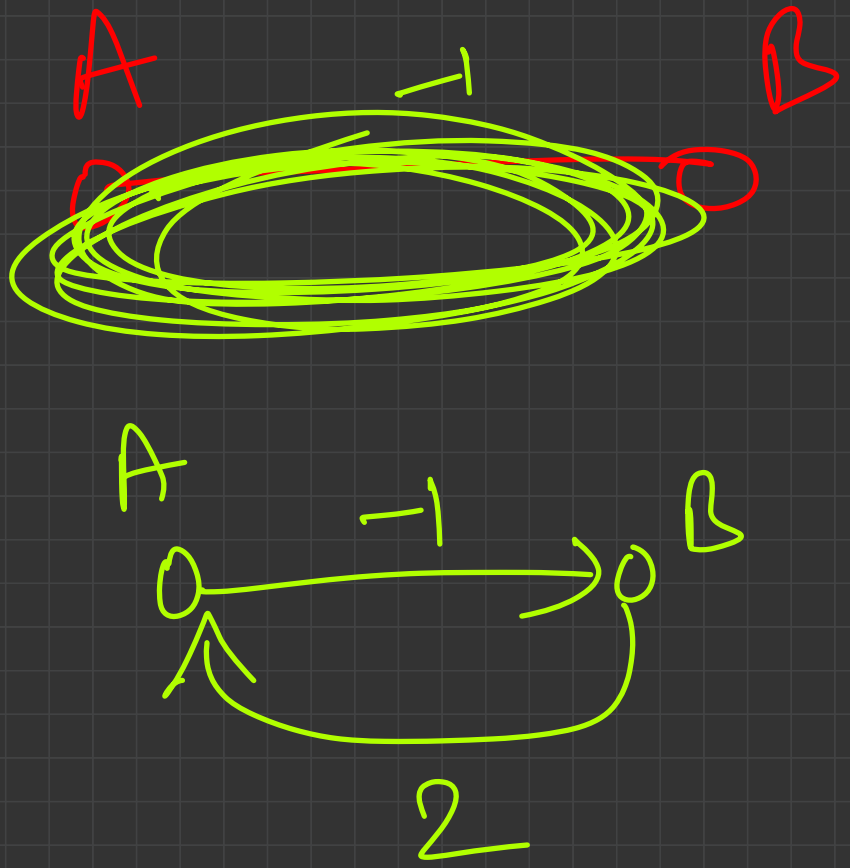
$-\infty$




2
0
-10
Src
1
n
3
n
4

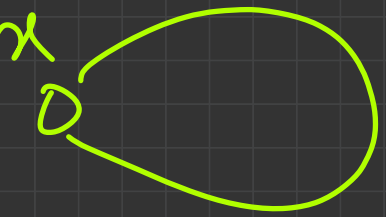


Undirected Graph


A
1
B
A
1
B
2

① Dijkstra doesn't work on negative edge weight

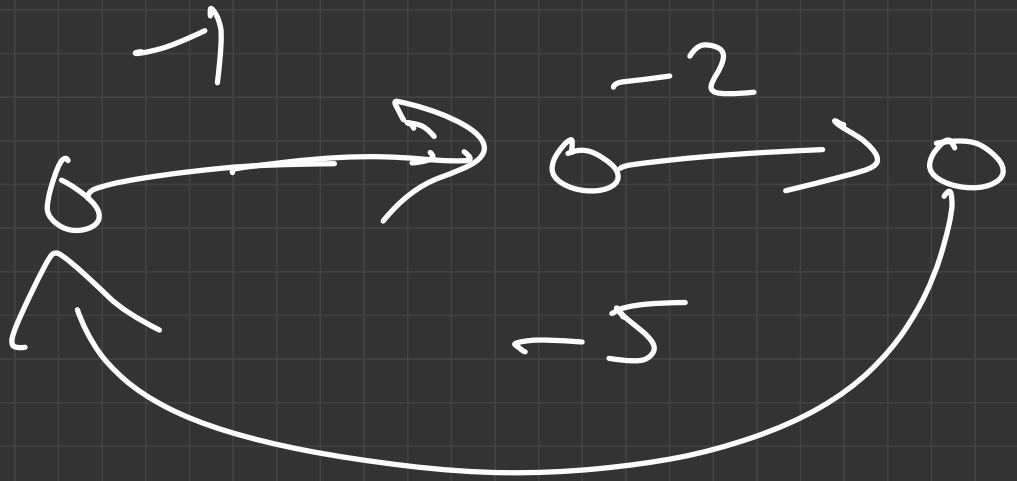
② Negative cycle →

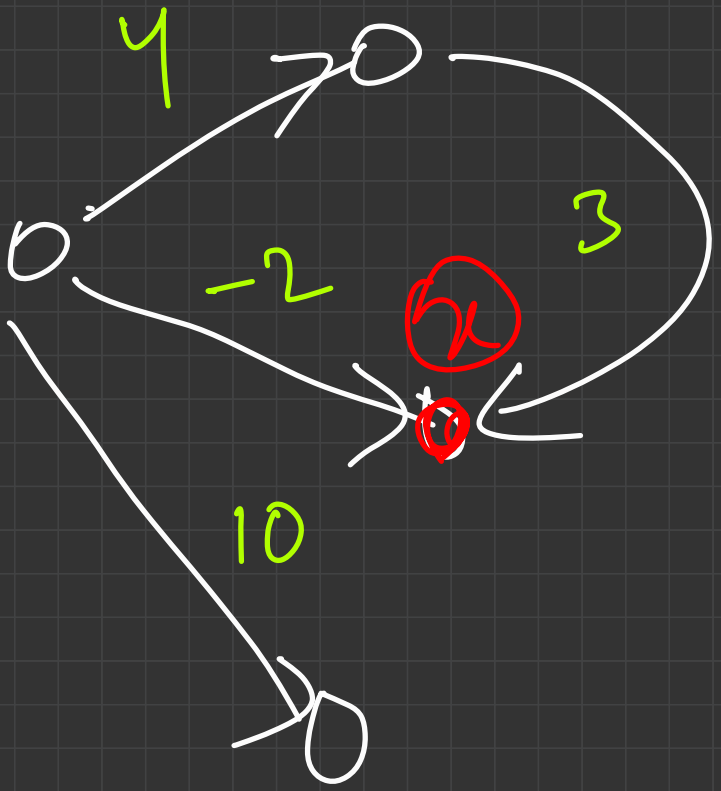


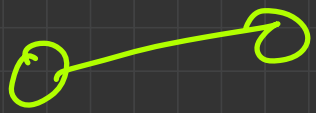
cumulative sum < 0

③ If graph is undirected and it has negative edge : then it is a

negative cycle as well



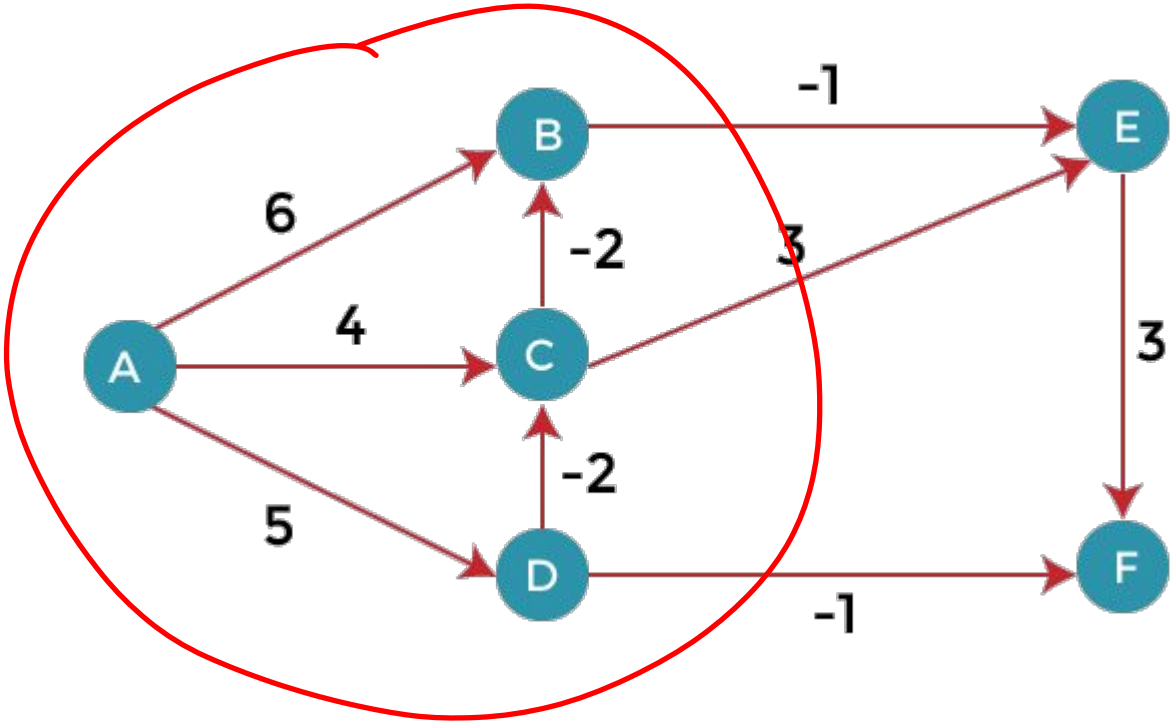

4
3
-2
10

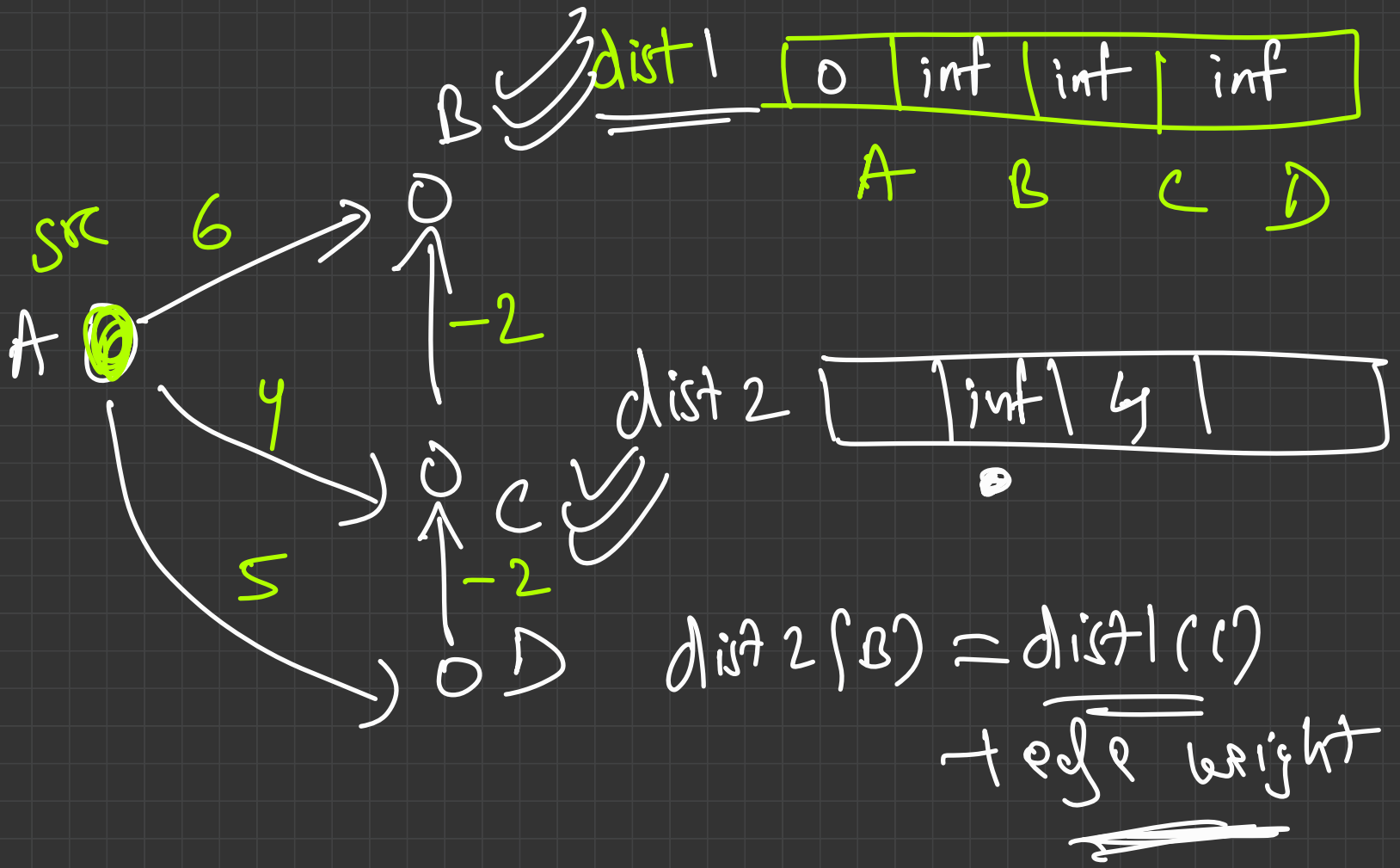


Bellman Ford

- Single Source Shortest Path Algorithm - Idea + Intuition
- Negative edge weights (in directed) without negative cycles
- Proof/Intuition:
    - Principle of Mathematical Induction
- Code
- Breaking Early
- Retrieving the shortest path?
- Finding a negative cycle

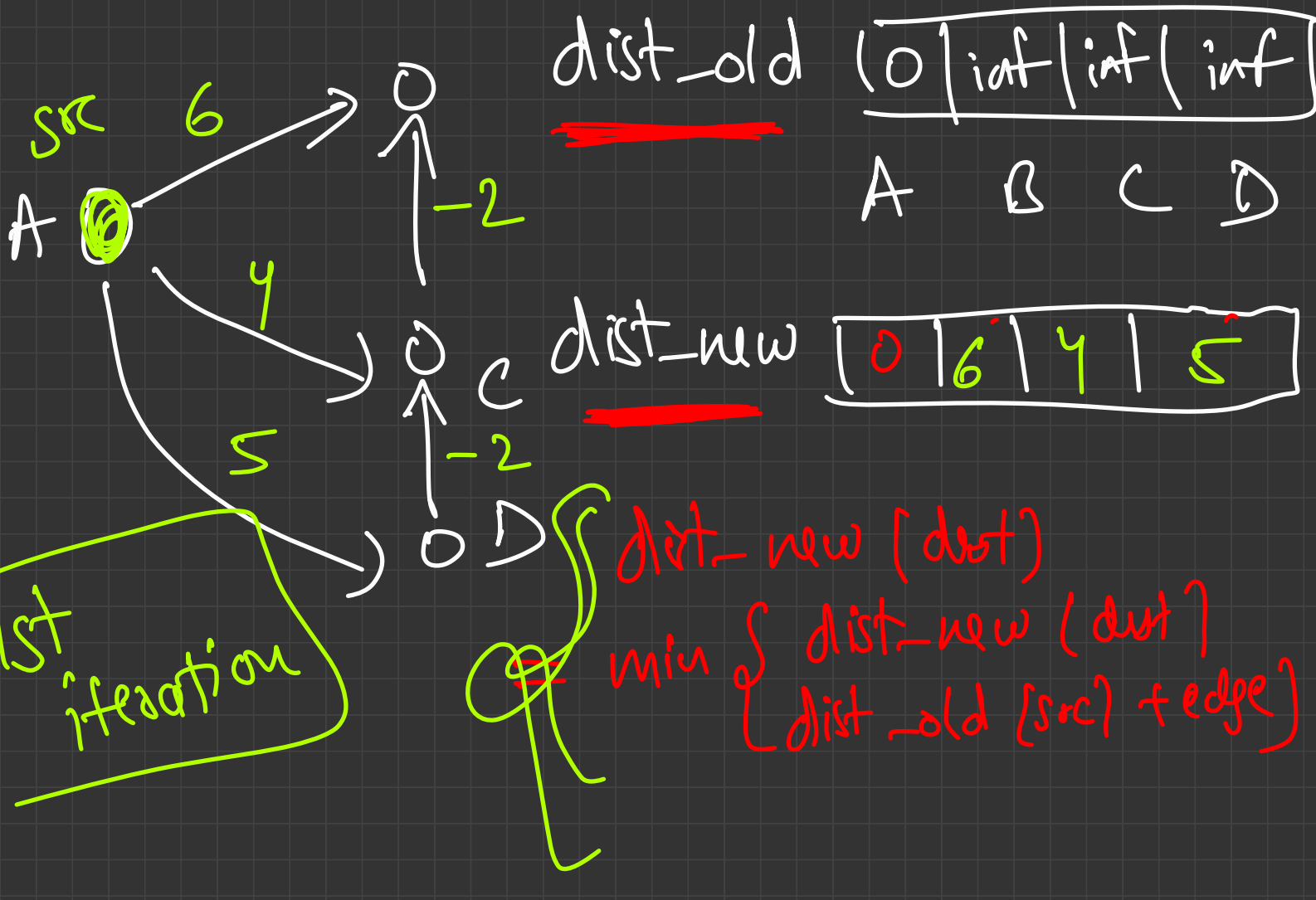
Bellman Ford Visualization



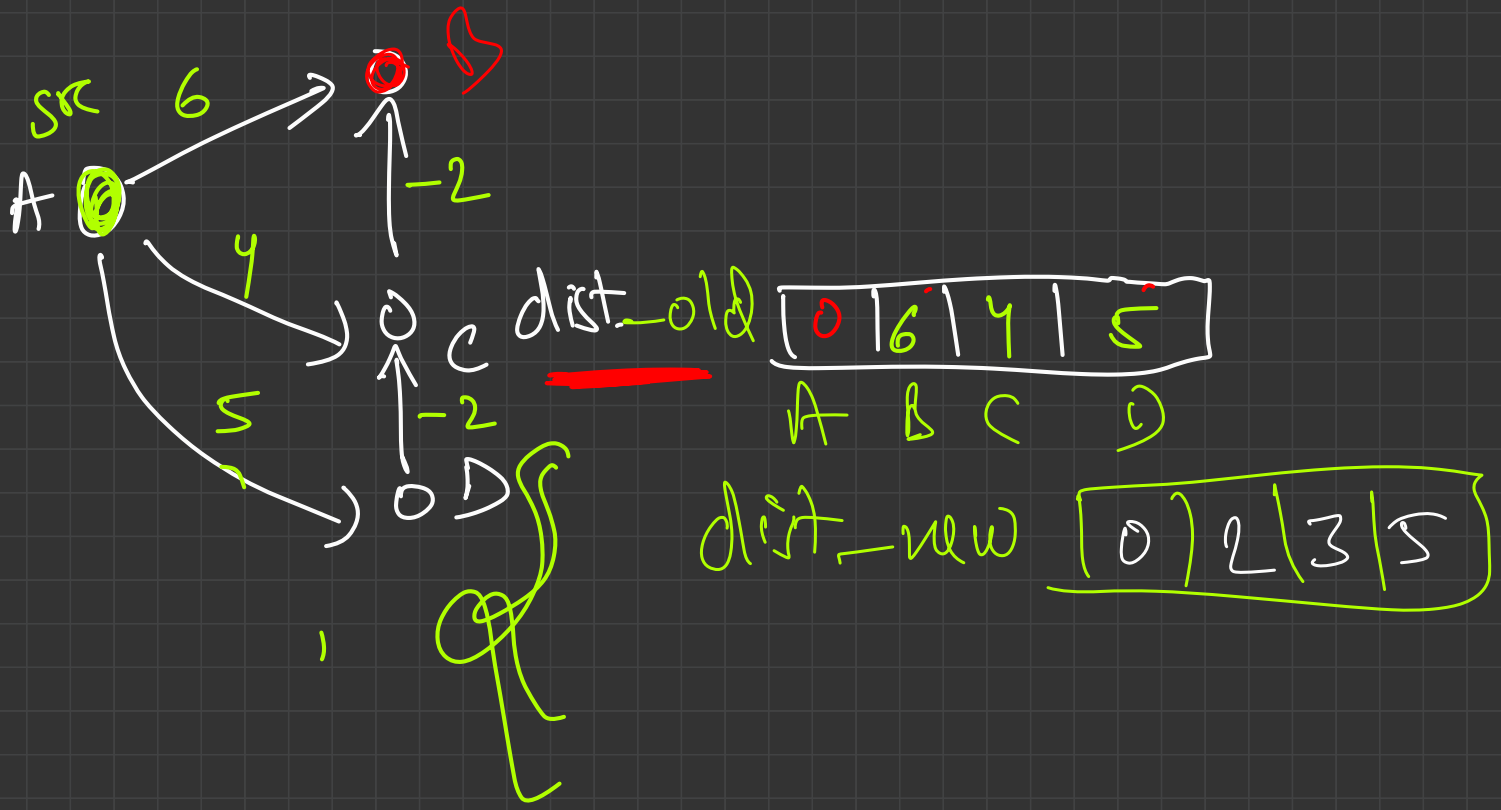

dist 1
0
inf
inf
inf
A
B
C
D
src
6
-2
4
5
-2
dist 2
inf
4
dist2(B) = dist1(C)
+ edge weight

Go through all the edges and do

(dist_new (dest) = dist_old [src] + edge weight)

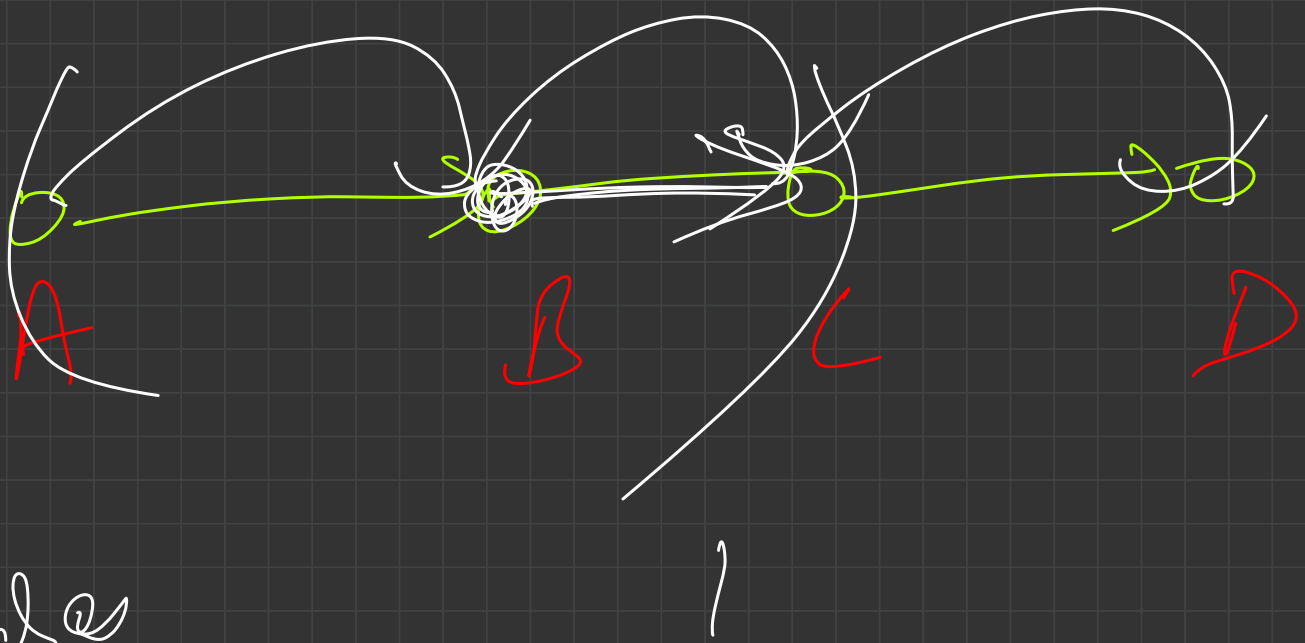

dist_old
0 | inf | inf | inf
A B C D
src
6
4
5
-2
-2
A
C
D
dist_new
0 | 6 | 4 | 5
1st iteration
dist_new (dest)
= min { dist_new (dest)
dist_old (src) + edge }

dist_new tells the shortest path to any node from the source with just 1 edge


src
A
6
B
-2
4
C
5
-2
D
dist_old
0 6 4 5
A B C D
dist_new
0 2 3 5

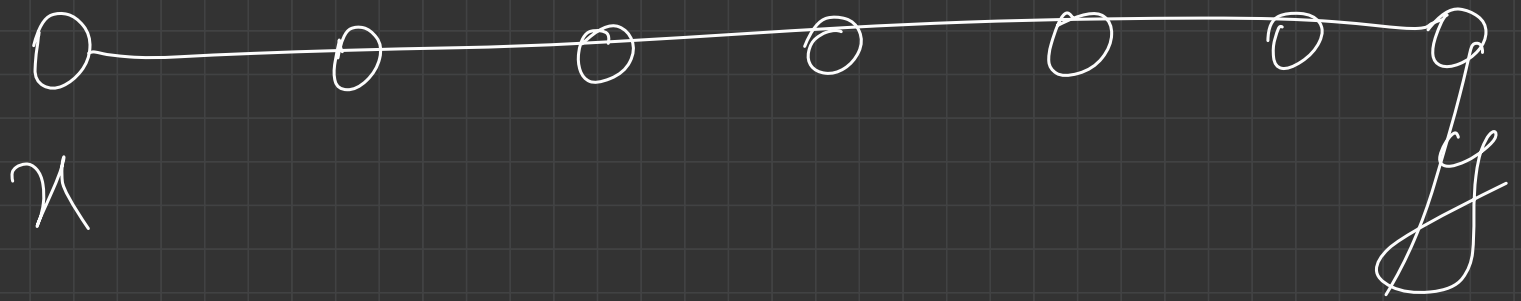
① You get shortest path to all those nodes whose shortest path comprises of 1 edge only

② 2 edges

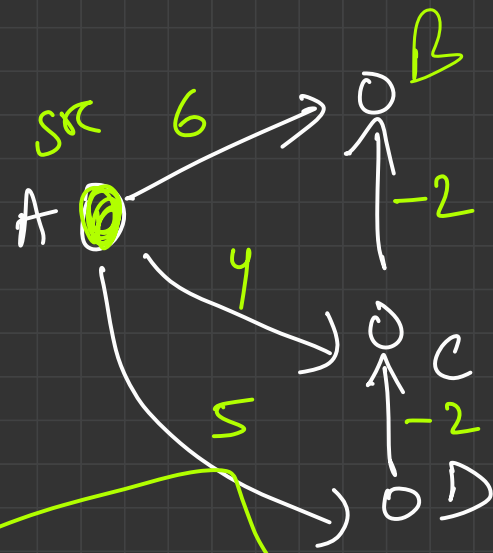


(2) edge

n nodes in graph



① Update the dist of dest based on previous dist of src

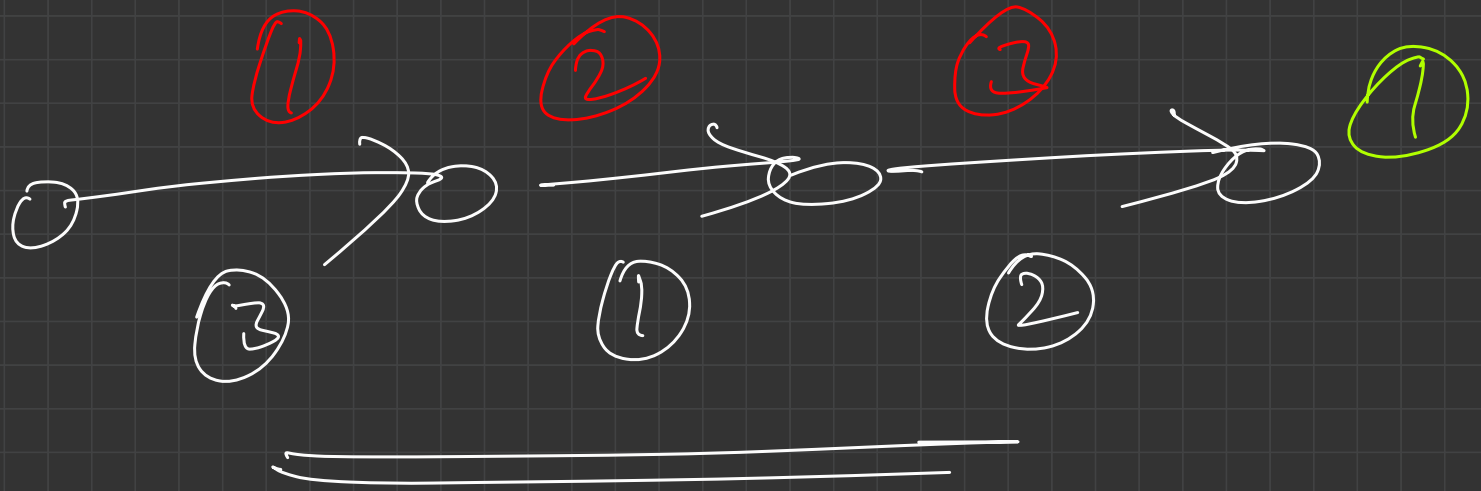


dist = | 0 | 6 | 4 | 5 |

A B C D

dist(dest) = min(dist(dest), dist(src) + edge)

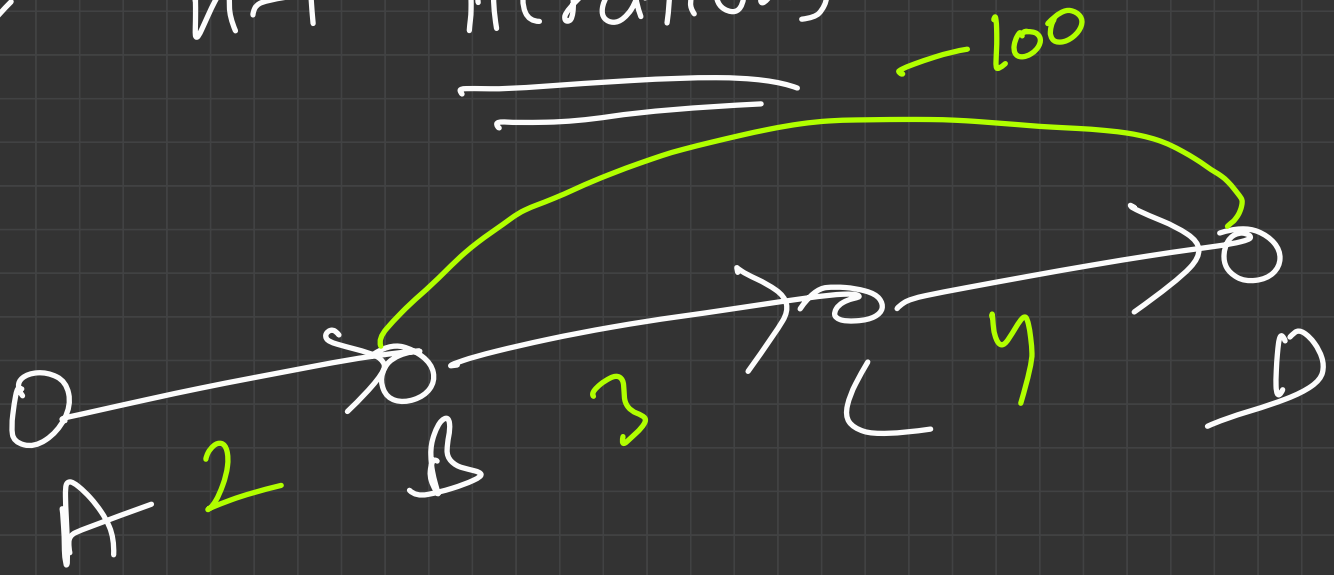
0	1	3	5



$n-1$

After k iterations all nodes with shortest paths having $\leq k$ edges would have found their shortest paths

After n-1 iterations



Floyd Warshall (DP)

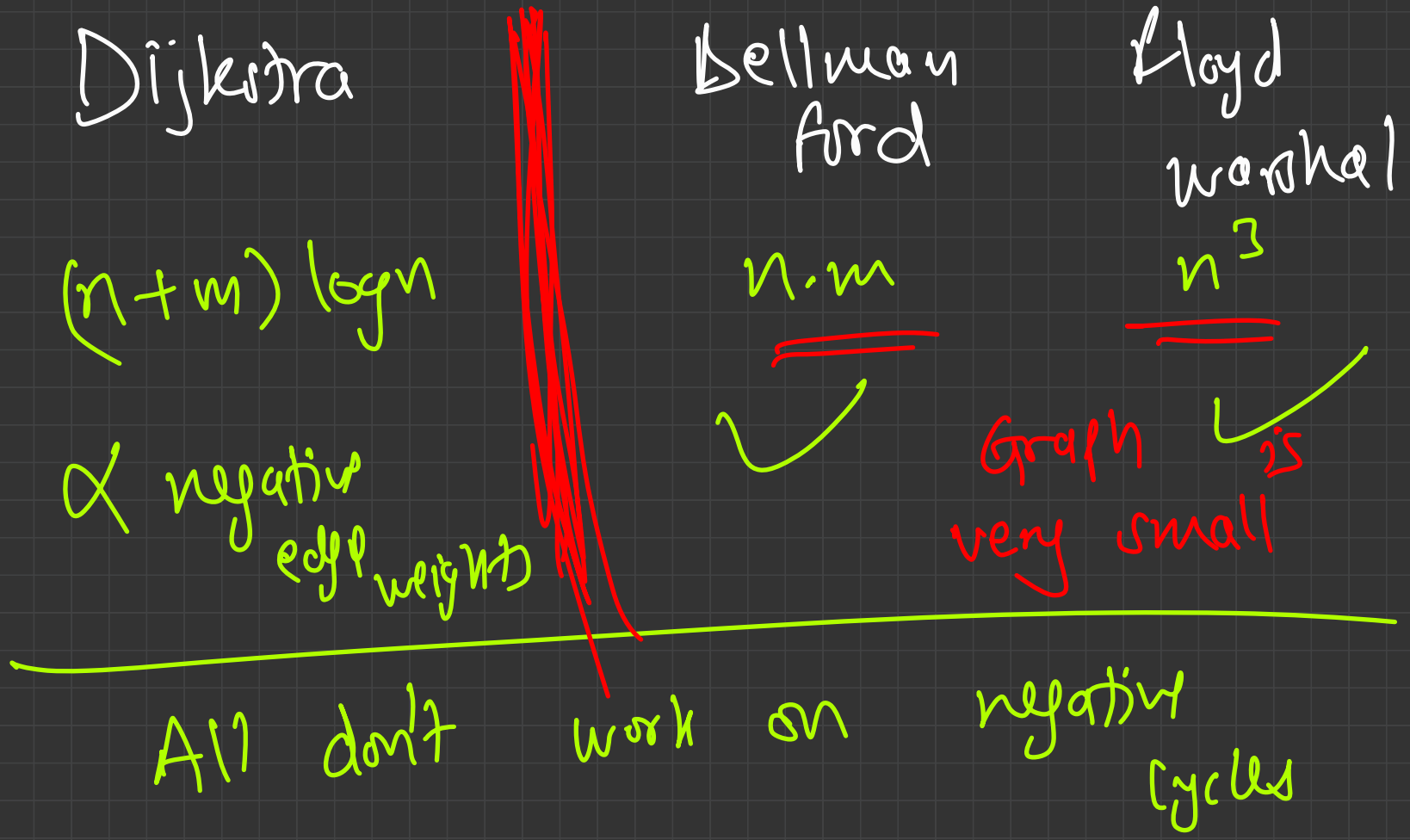
- Multi Source Shortest Path Algorithm - Idea + Intuition
- Negative edge weights (in directed) without negative cycles
- Proof/Intuition:
    - Principle of Mathematical Induction
- Code (Handling important loopholes)
- Detecting negative cycle

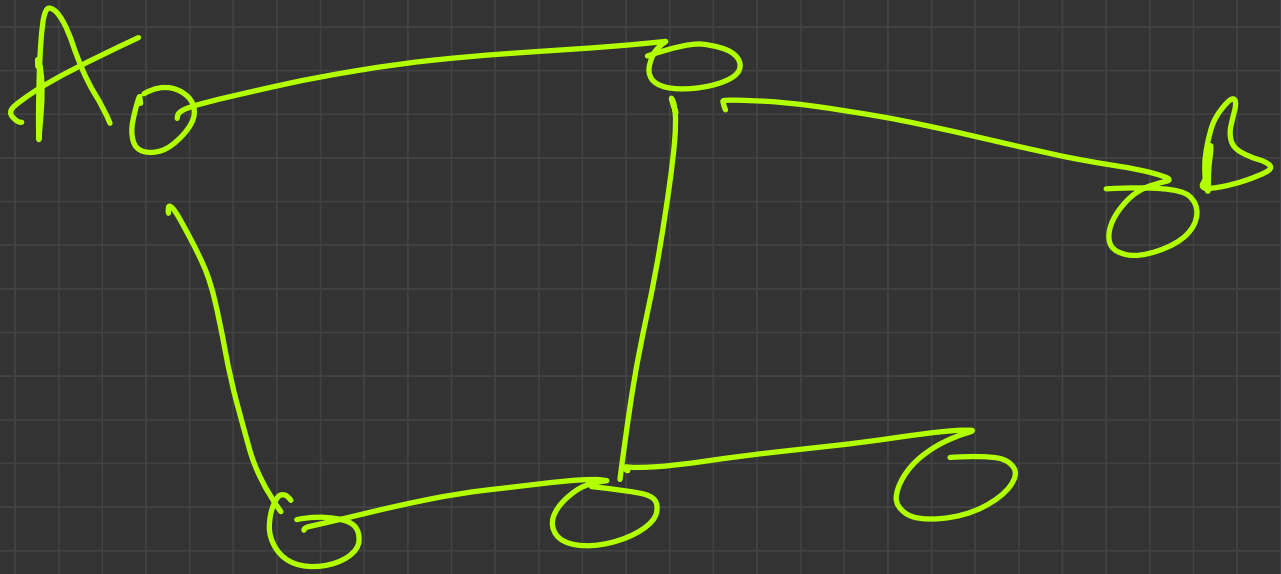
All pair shortest path algorithm

$n \leq 125$

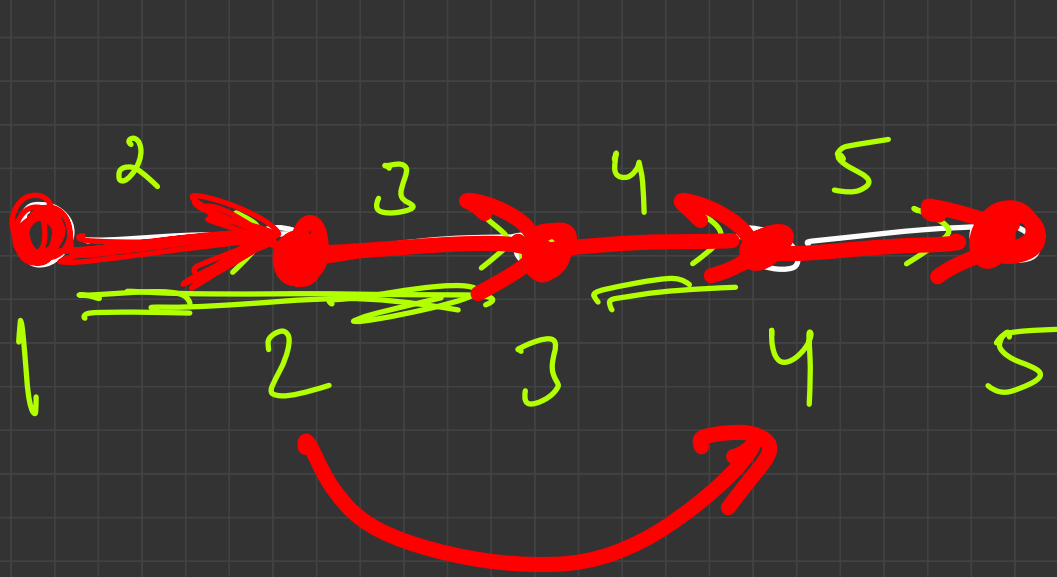
$m \leq 10^6$

$O(n) \cdot O((n+m) \cdot \log(n))$

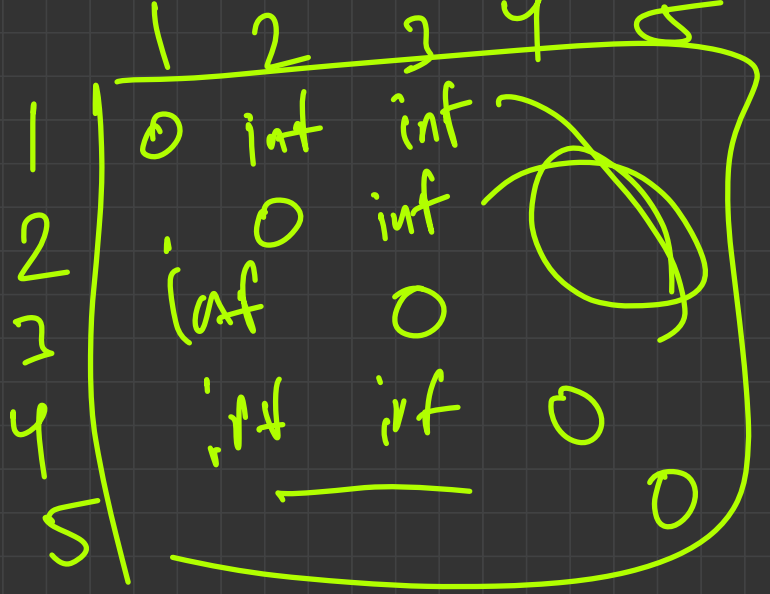

Dijkstra
Bellman ford
Floyd warshall
(n+m) logn
n.m
n^3
Graph is very small
x negative edge weights
All don't work on negative cycles


A

dist[i][j] $\Rightarrow$ int $\equiv$ represents min distance from i to j

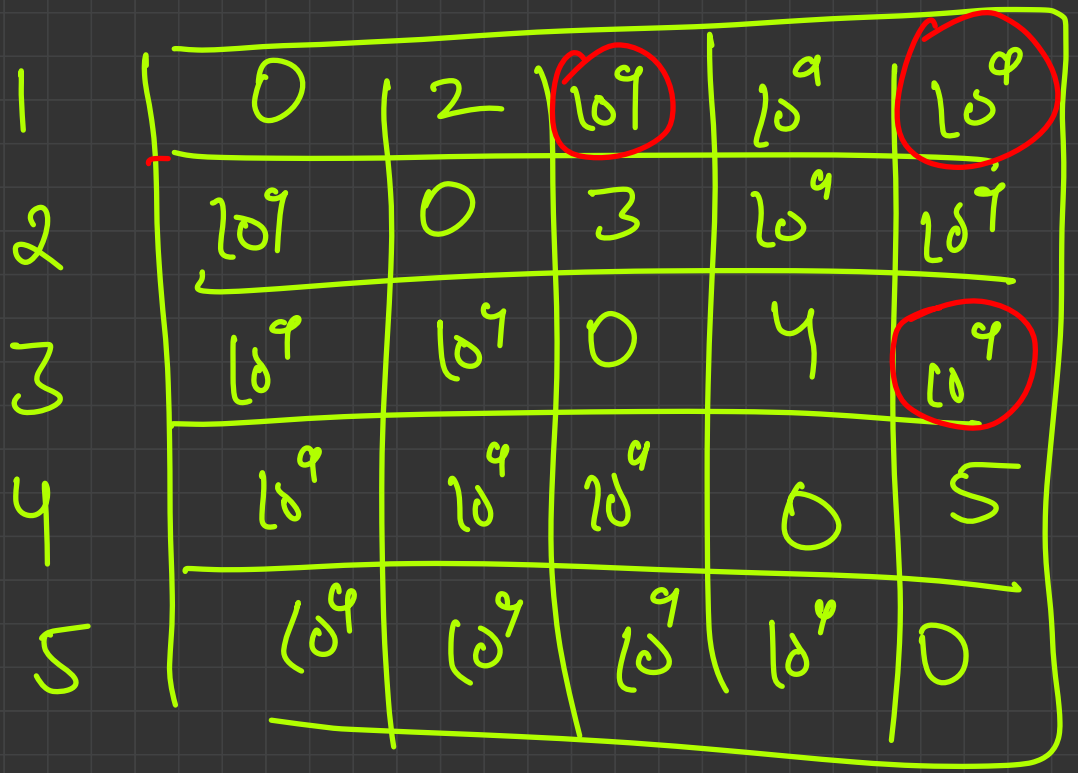


14



$$dist[i][j] = \min \begin{cases} dist[i][j] \\ dist[i][3] + dist[3][j] \end{cases}$$

1 2 3 4 5



dist[i][j]

dist[i][3]

+ dist[3][j]

dist[1][5]

Iterate over all the nodes and
Consider the node as intermediate
te and then try to update the
dist[i][j] for all pair (i, j)
using condition

$$\begin{cases} \min \begin{cases} dist[i][j] \\ dist[i][node] \\ dist[node][j] \end{cases} \end{cases}$$

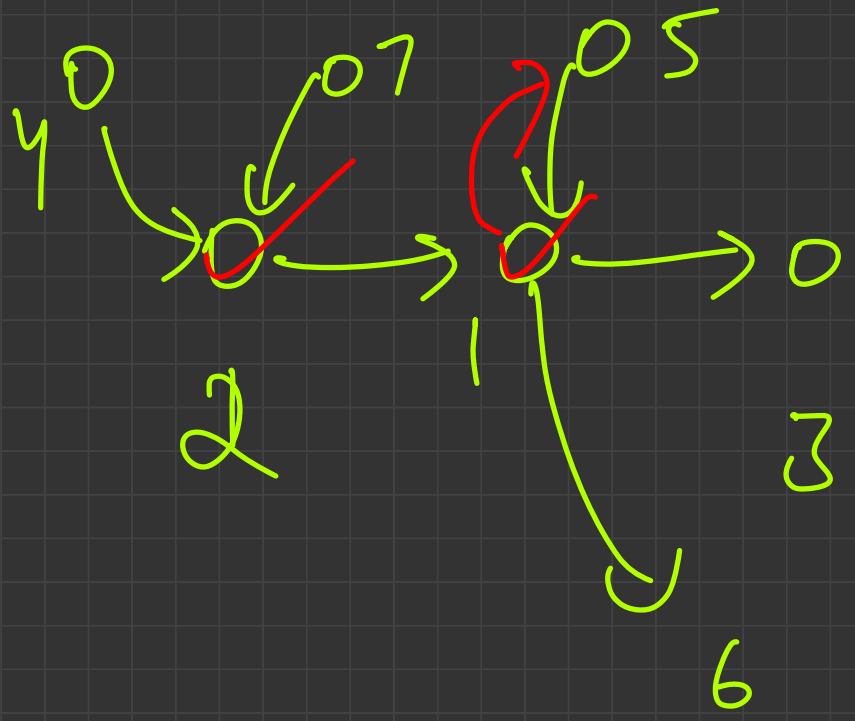
```
for( int k=0 ; k<n ; k++)
{
    for (int i -> {0, n-1})
    {
        for (int j -> {0, n-1})
        {
            dist[i][j] = min(dist[i][j],
                dist[i][k] + dist[k][j]);
        }
    }
}
```

after 3 iterations

After n iterations you would have found out all the shortest paths comprising of the first n nodes

$n=1$

Good indexing



(2 → 3) ✓

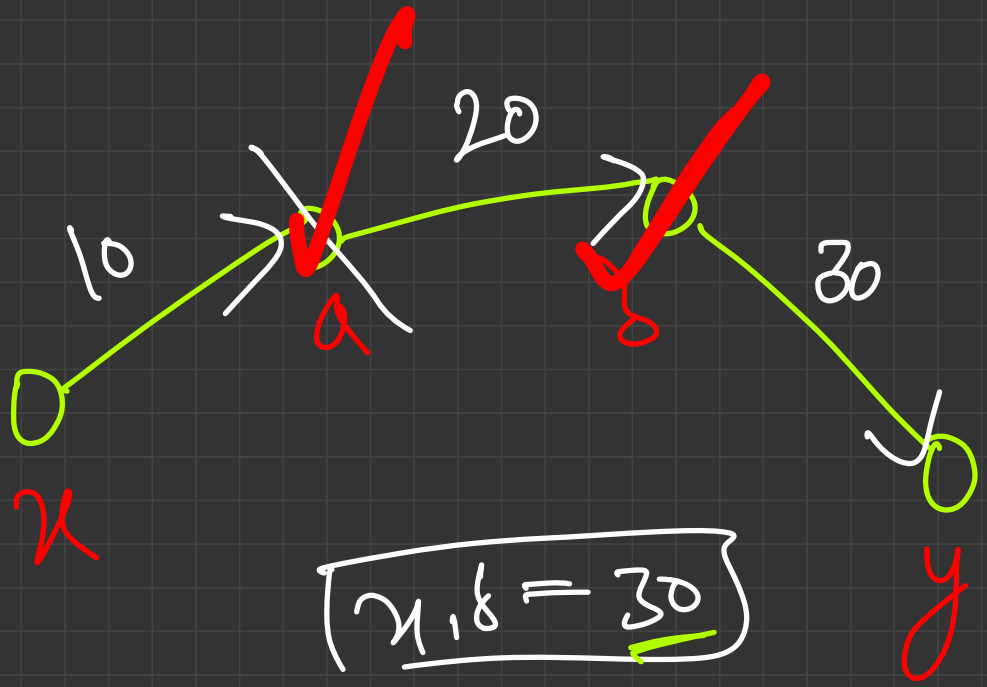
(5 → 6) ✓

(7 → 5)

(7 → 3) (4 → 3)

(4 → 6) (4 → 6)

(4 → 5)



$a_{1y} = 50$

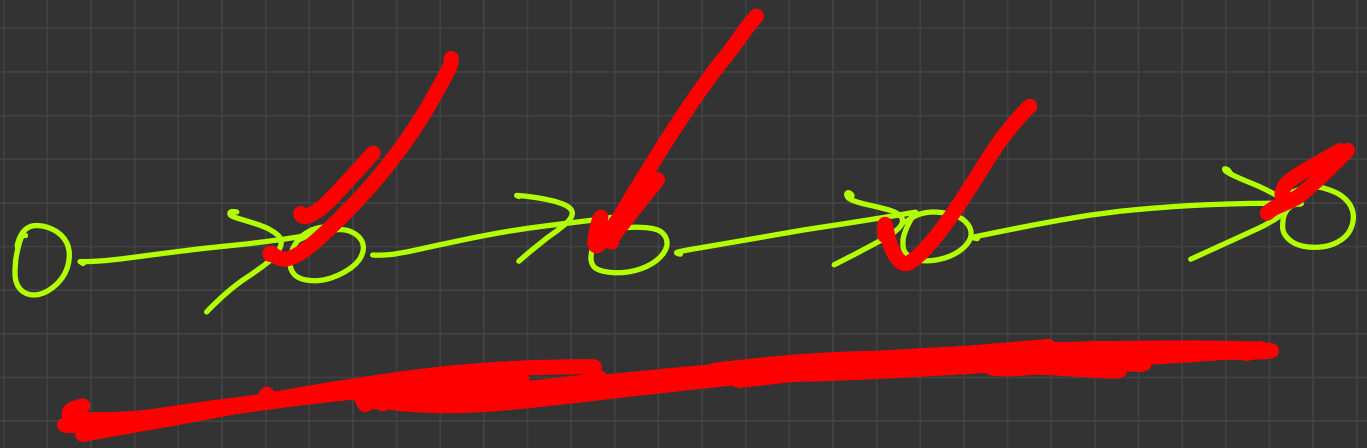
$x_{1y} = 60$

$x_{1b} = 30$

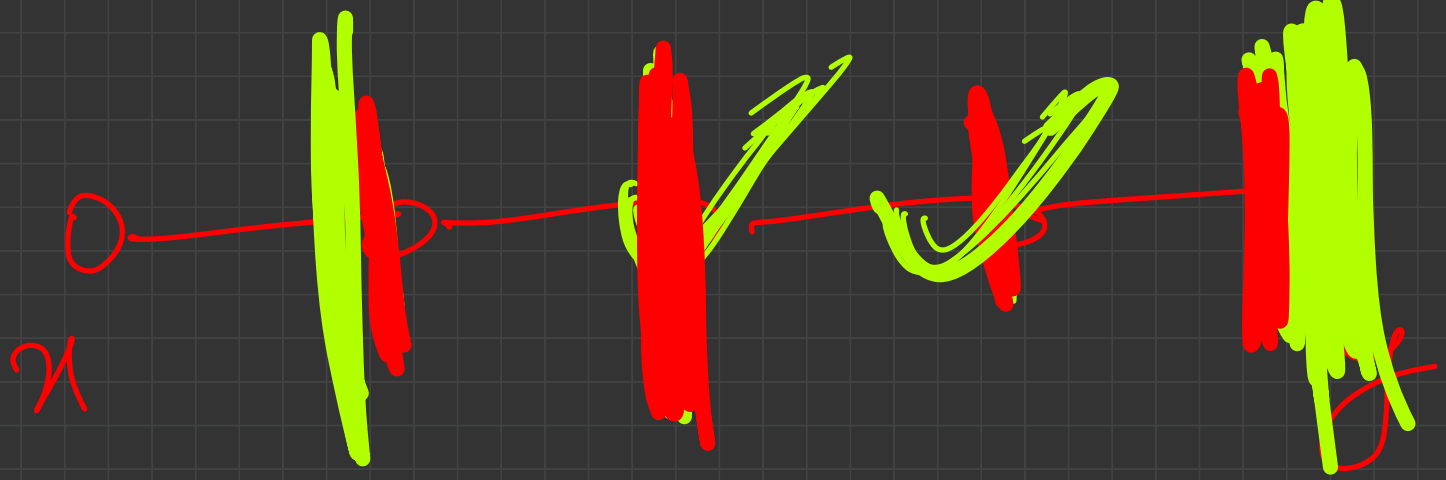
$x_{1a} = 10$

$a_{1b} = 20$

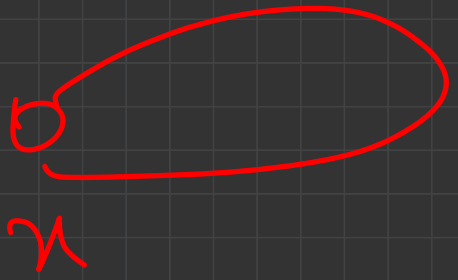
$b_{1y} = 30$

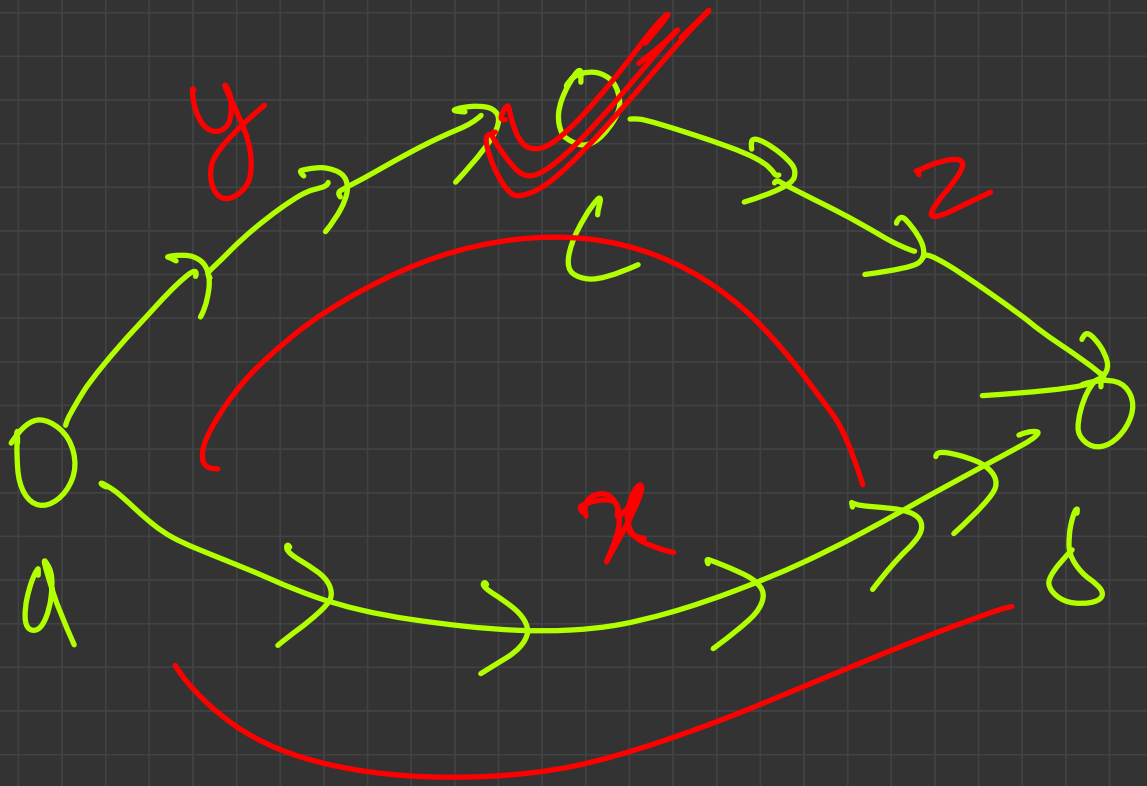


$$dp[i][j] = df[i][k] + df[k][j]$$



dist[i][i] < 0

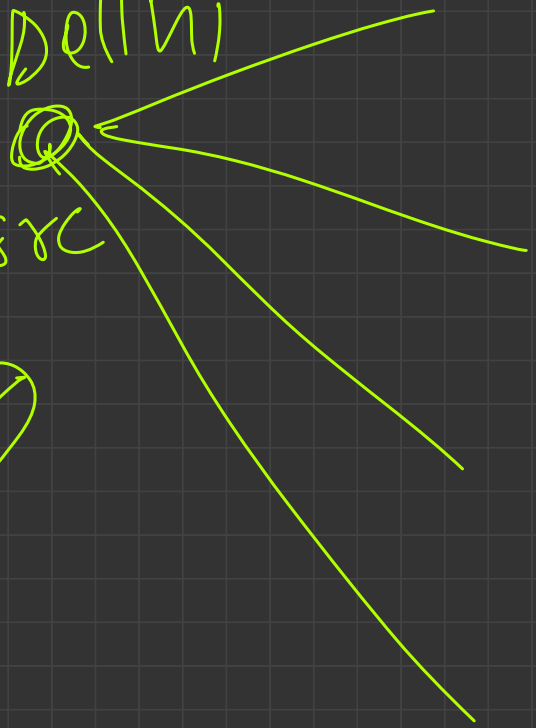




$y + z < x$

Dijkstra

Bellman ford



All shortest paths from source

Floyd Warshall

Floyd Warshall Visualization