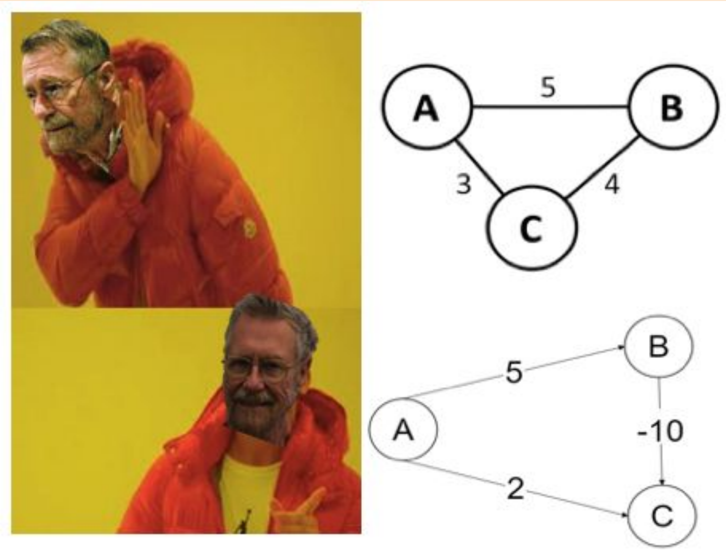
Bellman–Ford Algorithm

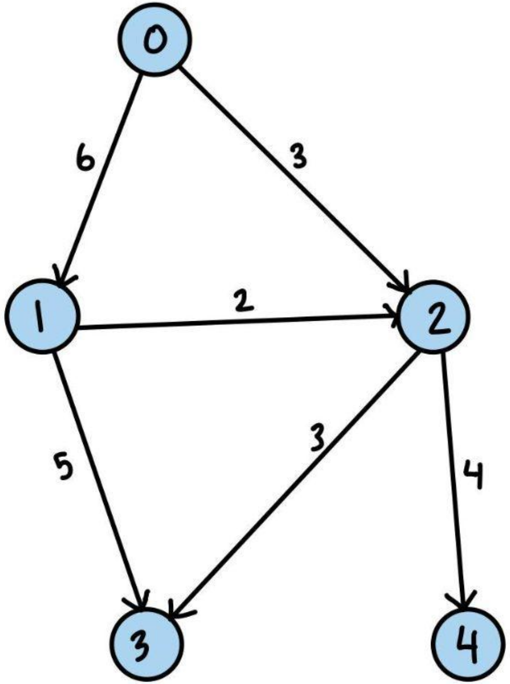


Presented by Group 4:
Zadie, Yan, Peng

Algorithm Introduction

- Shortest path algorithm
- Performs shortest path traversal on a source node from a weighted digraph
- Edges can have negative weights and the algorithm can detect negative weight cycles
- Real life application: Negative weights can exist in an transportation graph or a discount or rebate is offered for a specific flight or route

Review



Node	MinDist	lastStep
0	0	[0,0]
1	6	[0, 1]
2	3	[0, 2]
3	6	[2, 3]
4	7	[2,4)

Dijkstra's Algorithm

BMF:

1) Initalize all weights to ∞

2) Source = 0

3) Relax all edges - Inner loop runs E times

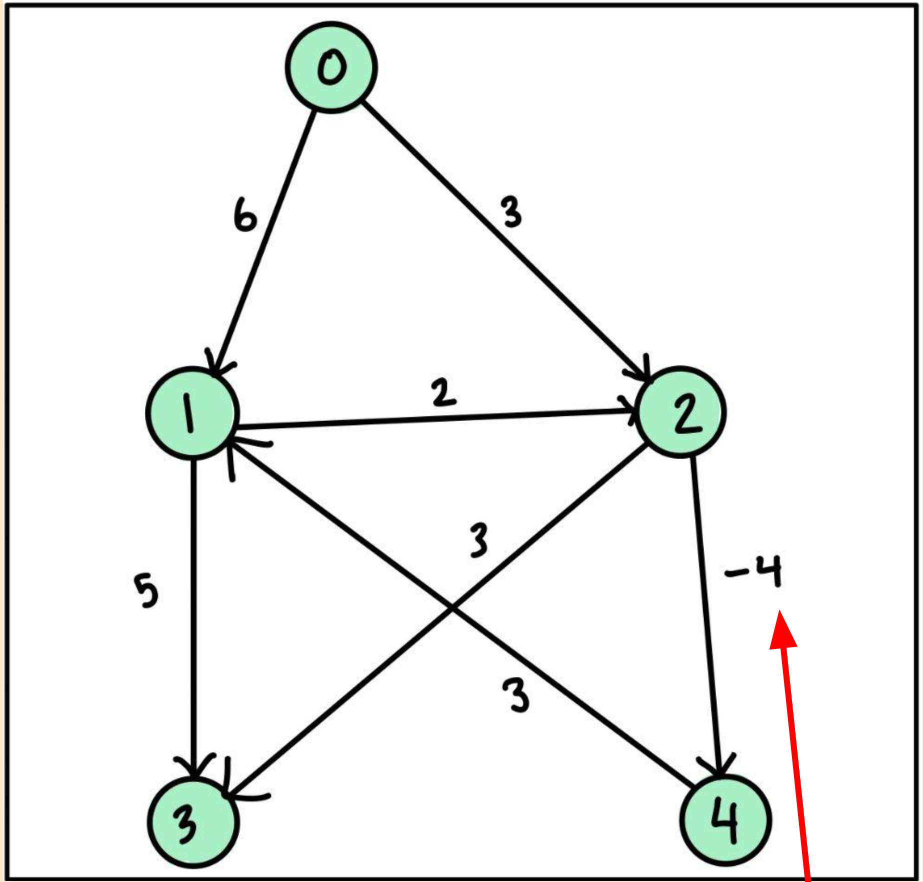
4) Outer loop runs V-1 times

5) Negative weight check

Algorithm should detect shortest path in V-1 loops

Otherwise:

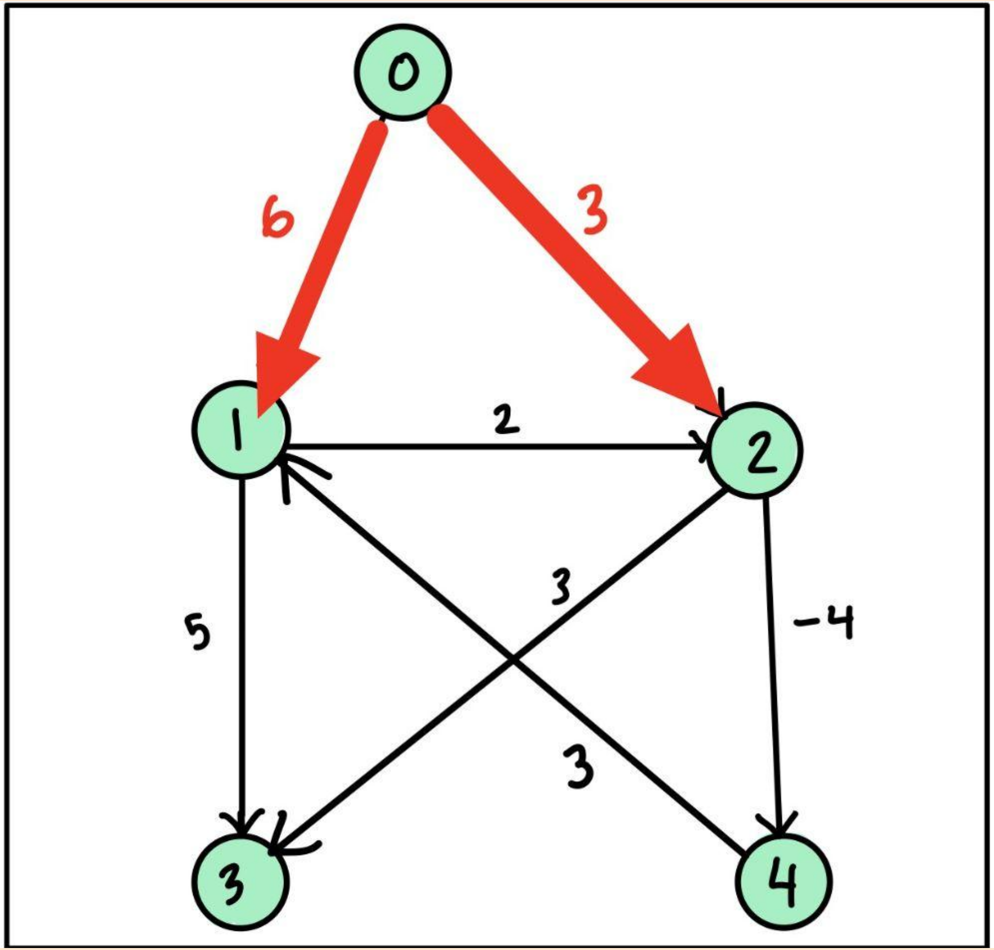
A negative cycle exists



New: Handles negative edge weights

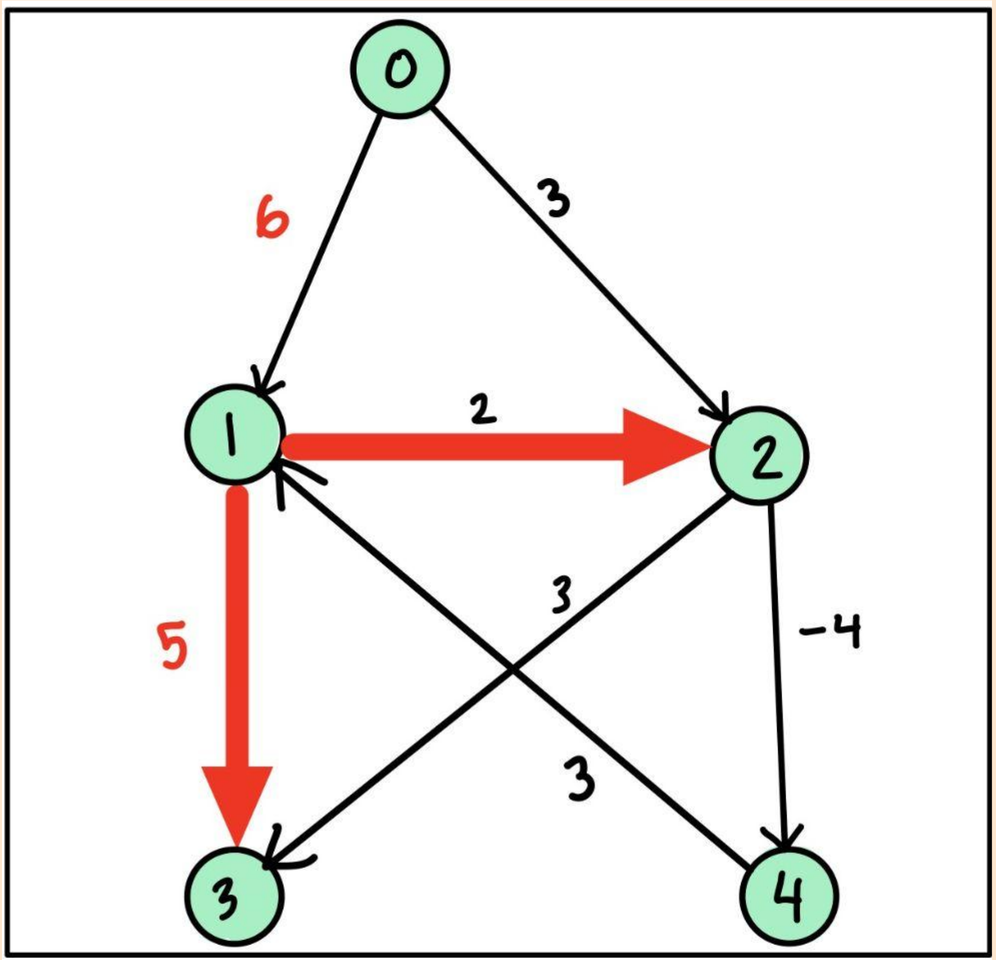
Initialization

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	∞
[0,2] [1,2]	2	∞
[1,3] [2,3]	3	∞
[2,4]	4	∞



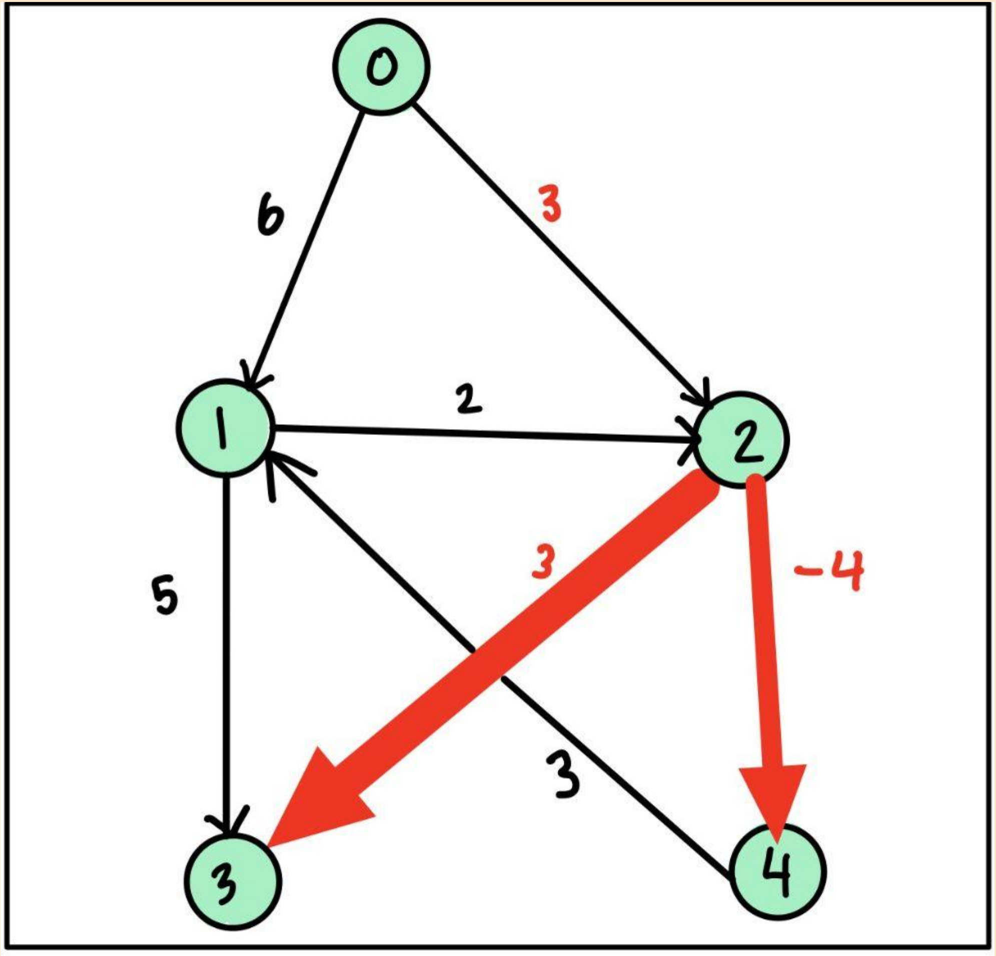
1st Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	6
[0,2] [1,2]	2	3
[1,3] [2,3]	3	∞
[2,4]	4	∞



1st Iteration

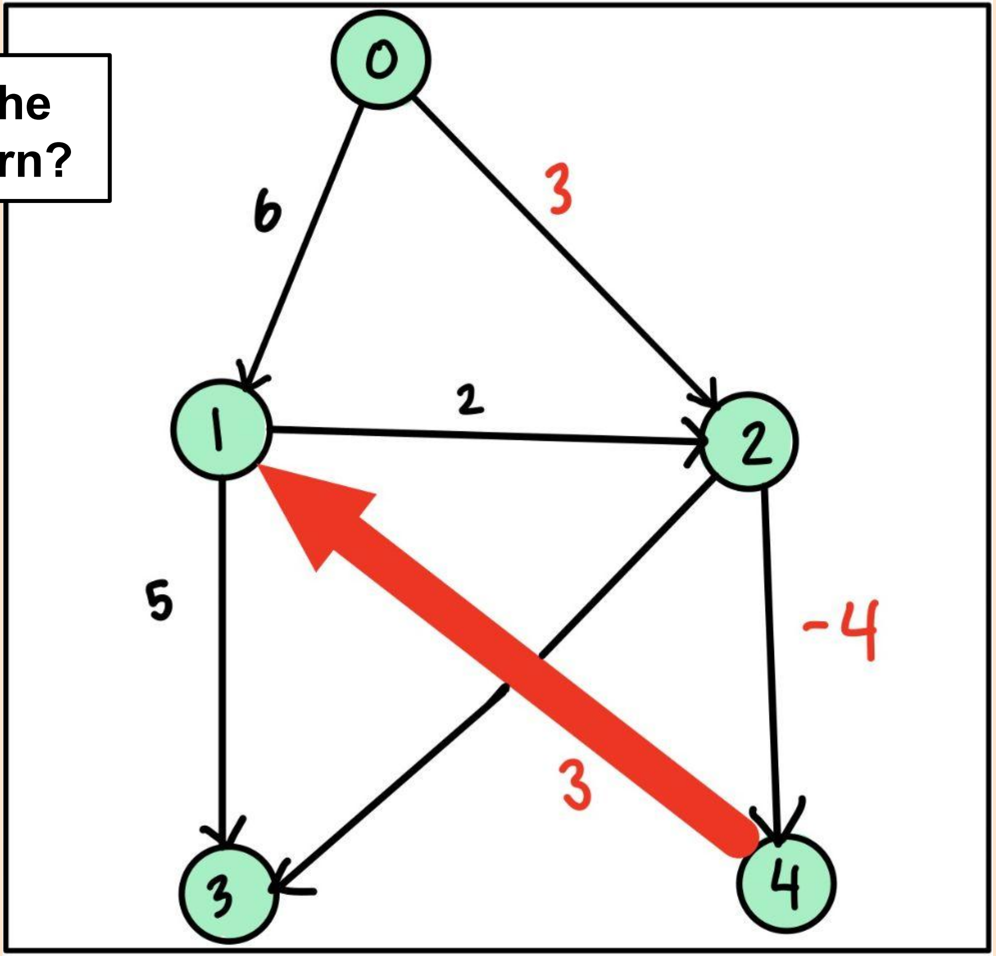
Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	6
[0,2] [1,2]	2	3
[1,3] [2,3]	3	11
[2,4]	4	∞



1st Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	6
[0,2] [1,2]	2	3
[1,3] [2,3]	3	6
[2,4]	4	-1

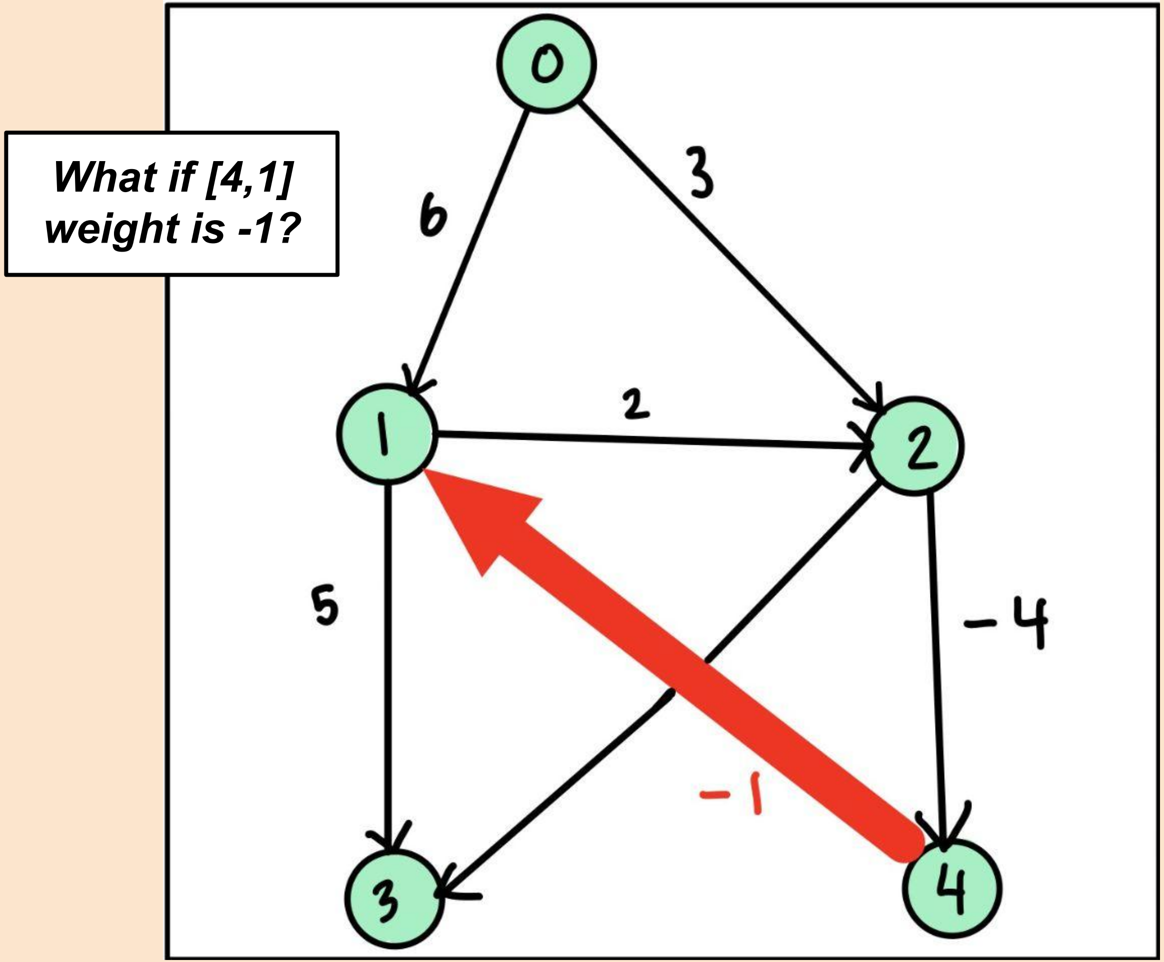
See the pattern?



1st Iteration

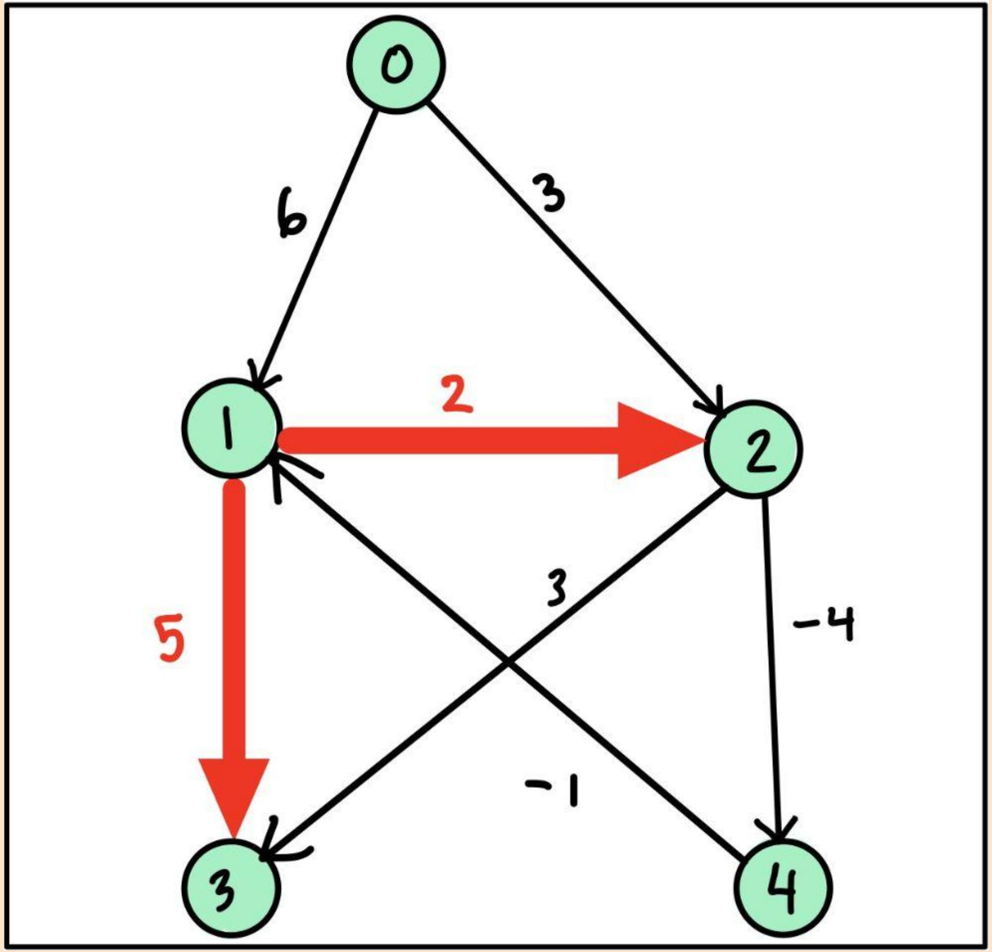
Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	2
[0,2] [1,2]	2	3
[1,3] [2,3]	3	6
[2,4]	4	-1

Now let's look at a <u>bad</u> implementation…



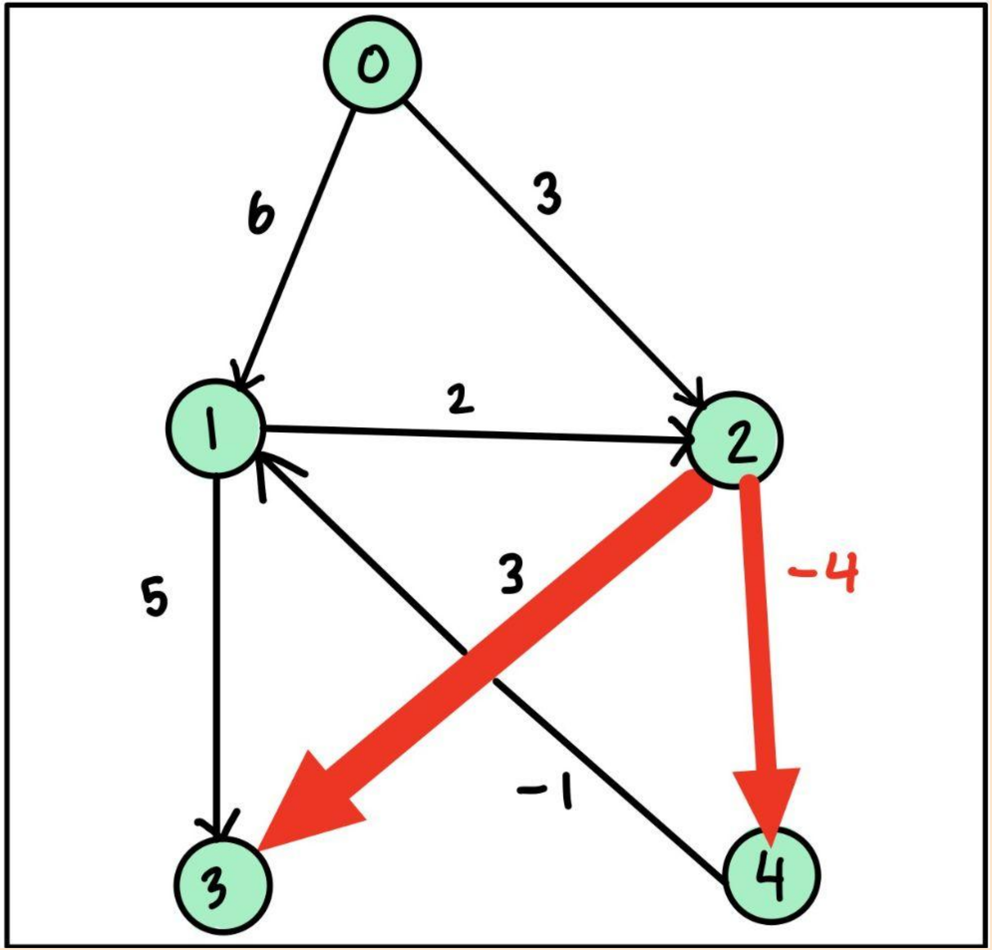
1st Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	-2
[0,2] [1,2]	2	3
[1,3] [2,3]	3	6
[2,4]	4	-1



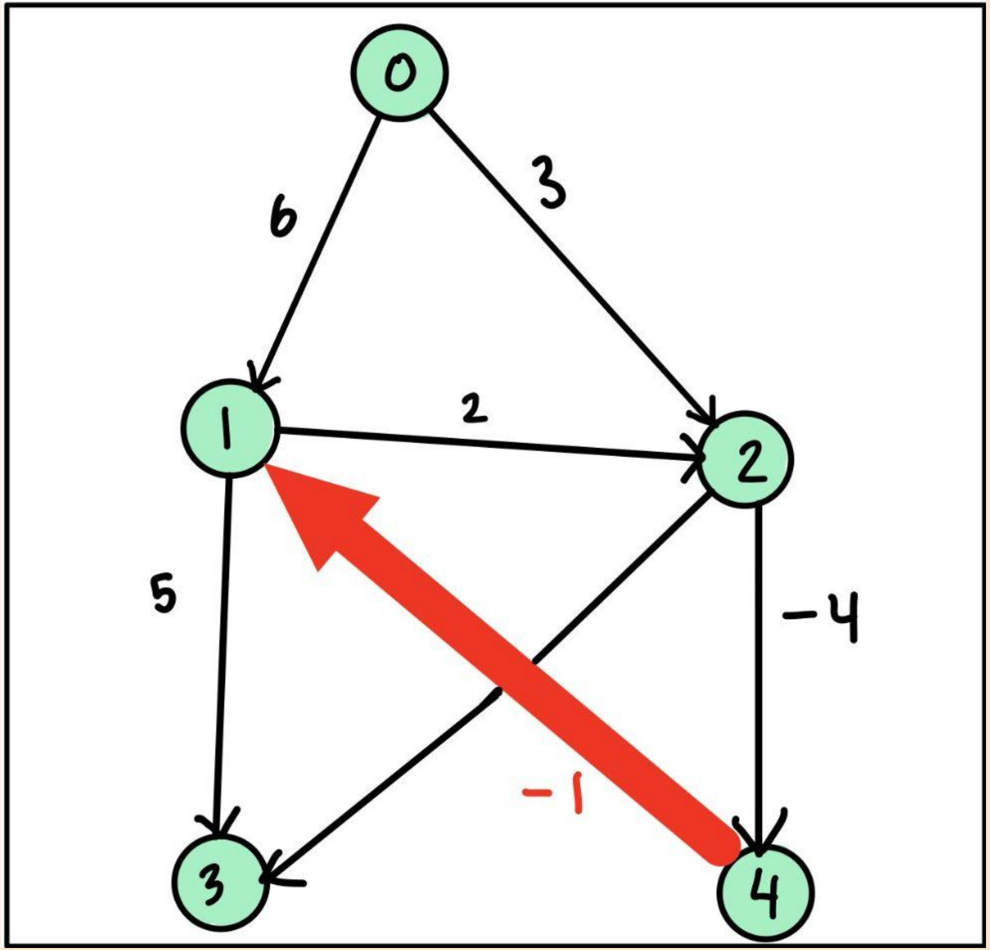
2nd Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	-2
[0,2] [1,2]	2	0
[1,3] [2,3]	3	3
[2,4]	4	-1



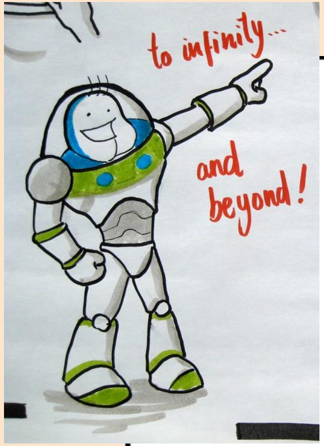
2nd Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	-2
[0,2] [1,2]	2	0
[1,3] [2,3]	3	3
[2,4]	4	-4

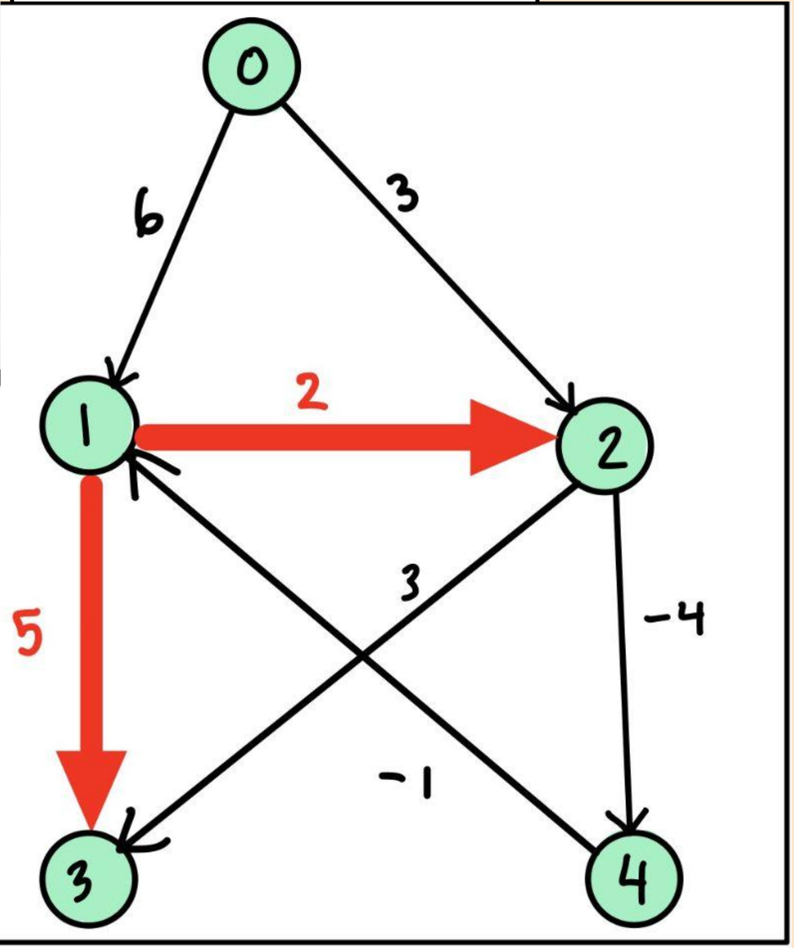


2nd Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	-5
[0,2] [1,2]	2	0
[1,3] [2,3]	3	3
[2,4]	4	-4



negative weight cycle



3rd Iteration

Edges	Node	Min Dist
	0	0
[0,1] [4,1]	1	-5
[0,2] [1,2]	2	-3
[1,3] [2,3]	3	0
[2,4]	4	-4

Bellman-Ford Pseudocode:

```
function BellmanFord(vertices, edges, source):
    // Step 1: Initialize distances from source to all other vertices as infinity
    dist = {}
    for each vertex in vertices:
        dist[vertex] = infinity
    dist[source] = 0

    // Step 2: Relax edges repeatedly
    for i from 1 to |vertices|-1:
        for each edge (u, v, w) in edges:
            if dist[u] + w < dist[v]:
                dist[v] = dist[u] + w

    // Step 3: Check for negative-weight cycles
    for each edge (u, v, w) in edges:
        if dist[u] + w < dist[v]:
            throw "Graph contains a negative-weight cycle"

    return dist
```

Time and Space Complexity

- Worst case time complexity: **O (V^3)**

- Average case time complexity: **O(|V| * |E|)**

- Best case time complexity: **O(E)**

- Space Complexity: **O(V + E) for all cases**

Thanks for learning with us!

References

1) A. Shimbel, "Structure in communication nets," in Proceedings of the Symposium on Information Networks, New York, NY, USA: Polytechnic Press of the Polytechnic Institute of Brooklyn, pp. 199-203 1955.
2) R. Bellman, “On a routing problem,” Quarterly of Applied Mathematics, vol. 16, no. 1, pp. 87–90, 1958.
3) L. R. Ford Jr., "Network Flow Theory," Paper P-923, Santa Monica, CA, USA: RAND Corporation, 1956.
4) E. F. Moore, "The shortest path through a maze," in Proc. Internat. Sympos. Switching Theory 1957, Part II, Cambridge, MA, USA: Harvard Univ. Press, 1959, pp. 285-292.
5) J. Y. Yen, “An algorithm for finding shortest routes from all source nodes to a given destination in general networks,” Quarterly of Applied Mathematics, vol. 27, no. 4, pp. 526–530, 1970.
6) M. J. Bannister and D. Eppstein, “Randomized speedup of the bellman–Ford algorithm,” 2012 Proceedings of the Ninth Workshop on Analytic Algorithmics and Combinatorics (ANALCO), 2012.
7) “Bellman–Ford algorithm,” Wikipedia, 27-Feb-2023. [Online]. Available: https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm. [Accessed: 23-Mar-2023].