Detecting Currency Arbitrage

Ayomide hakeem

What is Currency Arbitrage

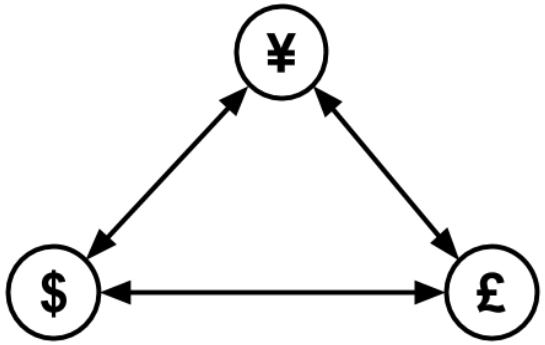
$

Currency arbitrage in simple terms involves buying and selling the same currency pair simultaneously in different markets to profit from price differences:

₹

Traders buy a currency pair at a lower exchange rate in one market and sell it at a higher exchange rate in another market. Because of the speed of trading algorithms, these opportunities can last only a fraction of a second.

What is Currency Arbitrage



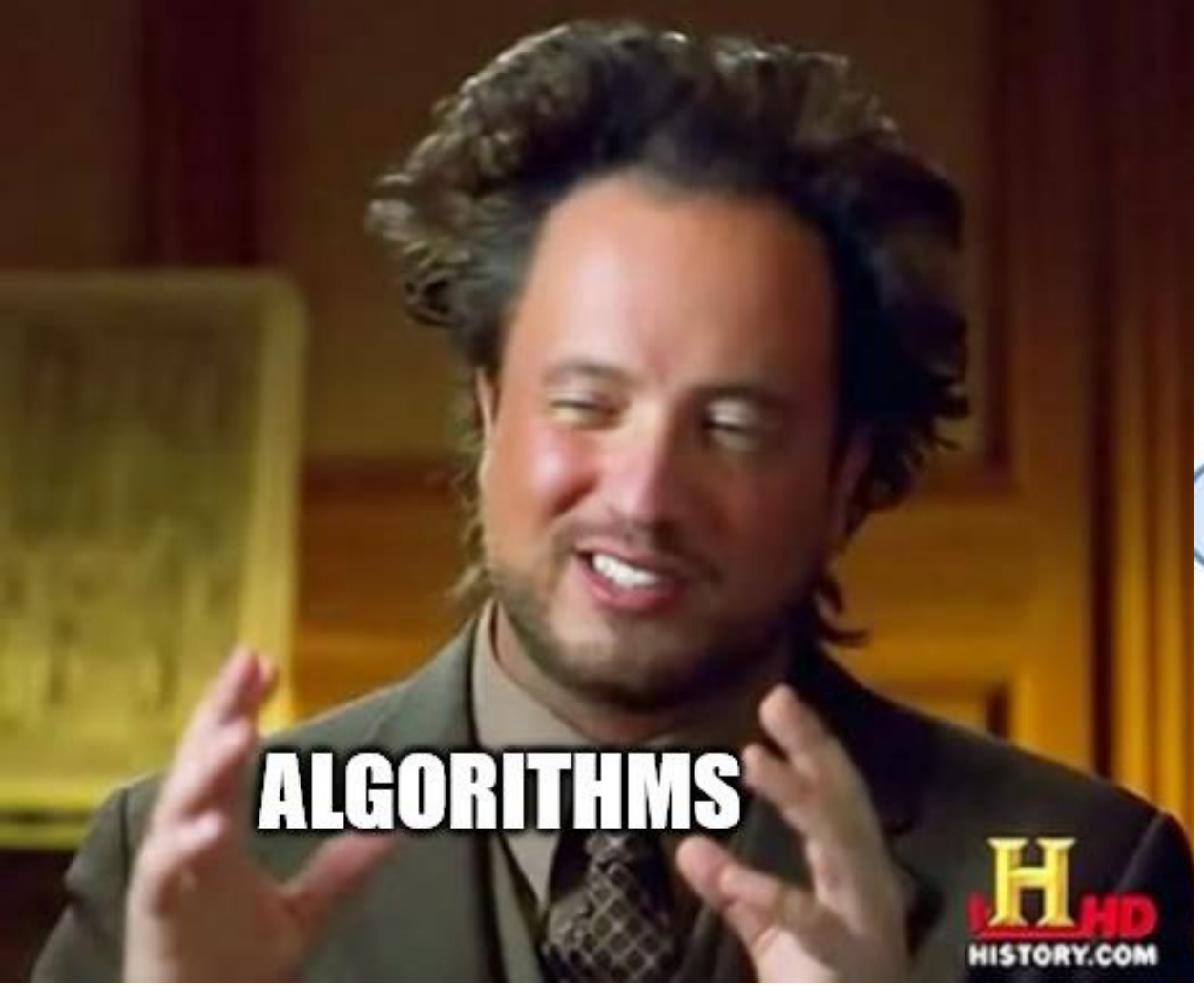
Market	Exchange Rate
pound (£) / dollar ($)	0.8 pounds/dollar
yen (¥) / pound (£)	100 yen/pound
dollar ($) / yen (¥)	0.013 dollars/yen

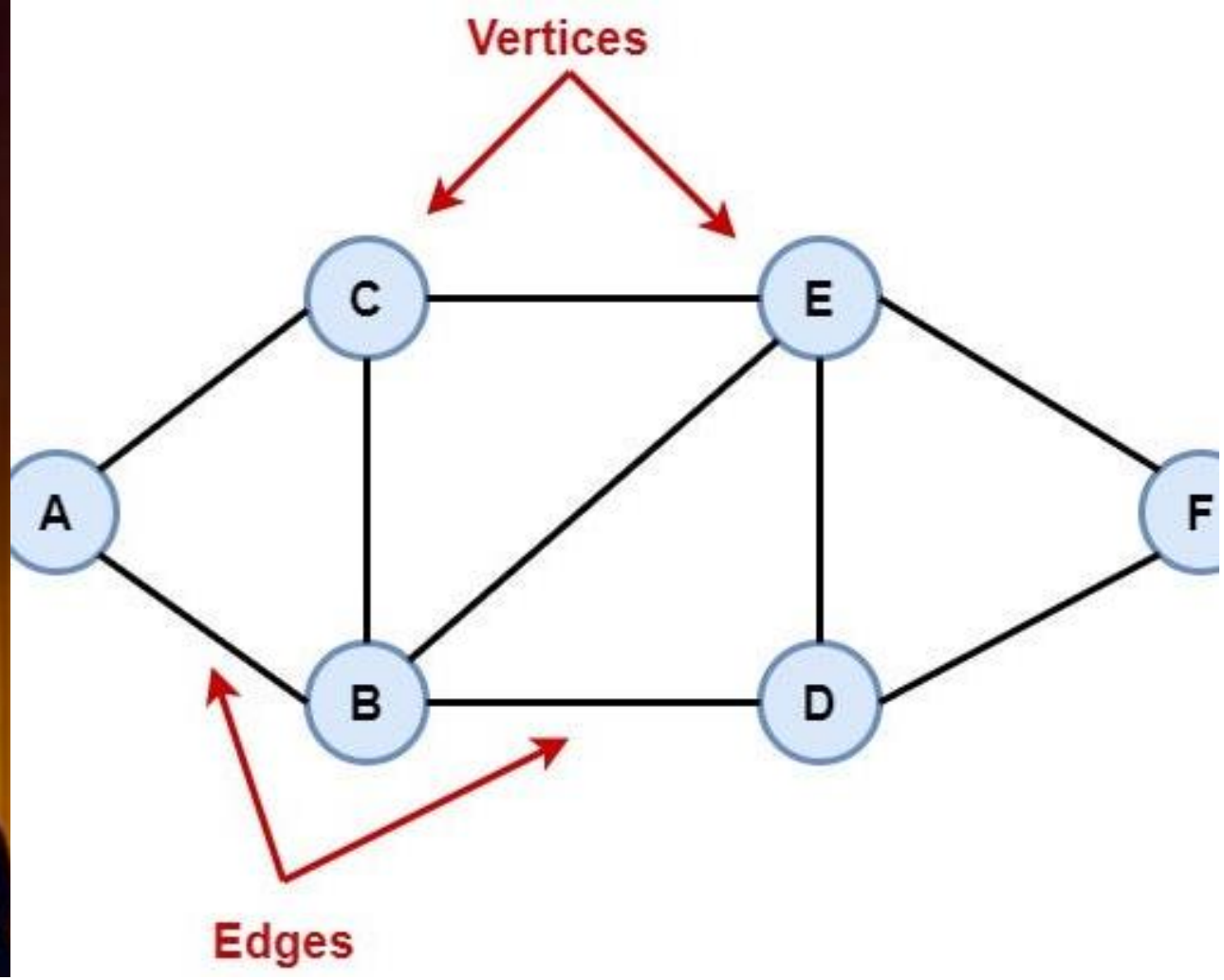
$1 for £0.8

£0.8 for ¥80

¥80 for $1.04

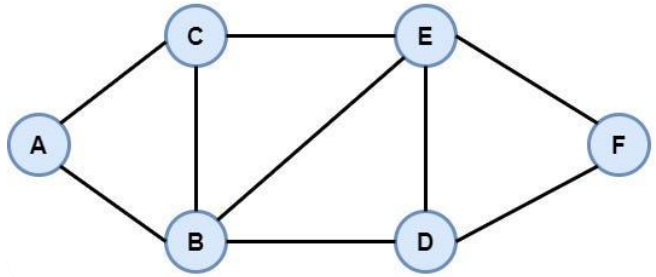
We made Profit!



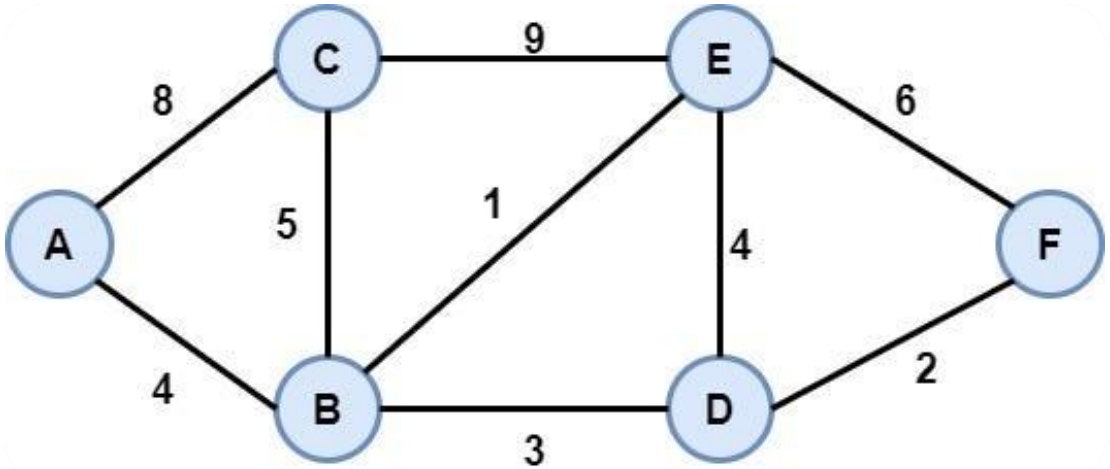


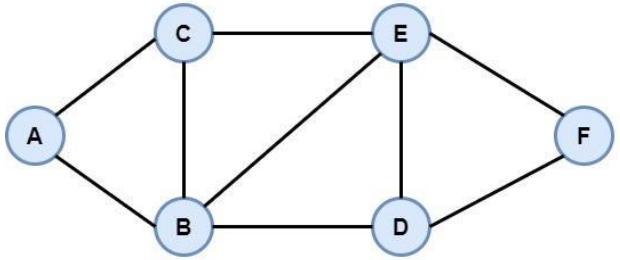
Can We Detect Currency Arbitrage Algorithmically?

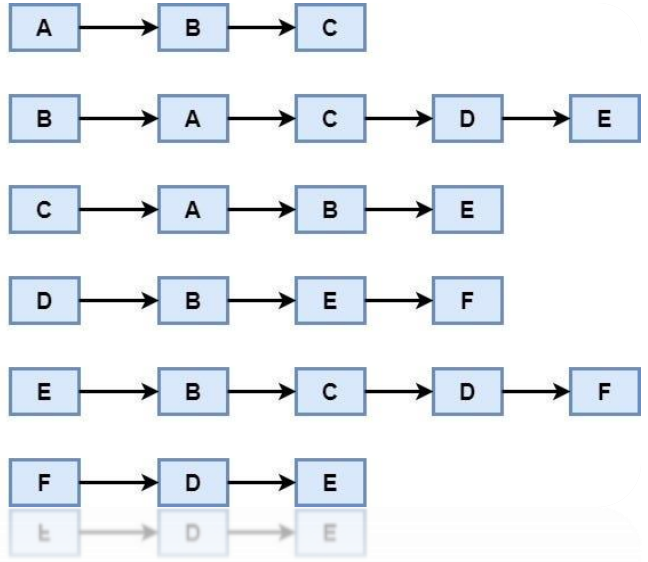
Graphs



	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	1	1	0
C	1	1	0	0	1	0
D	0	1	0	0	1	1
E	0	1	1	1	0	1
F	0	0	0	1	1	0

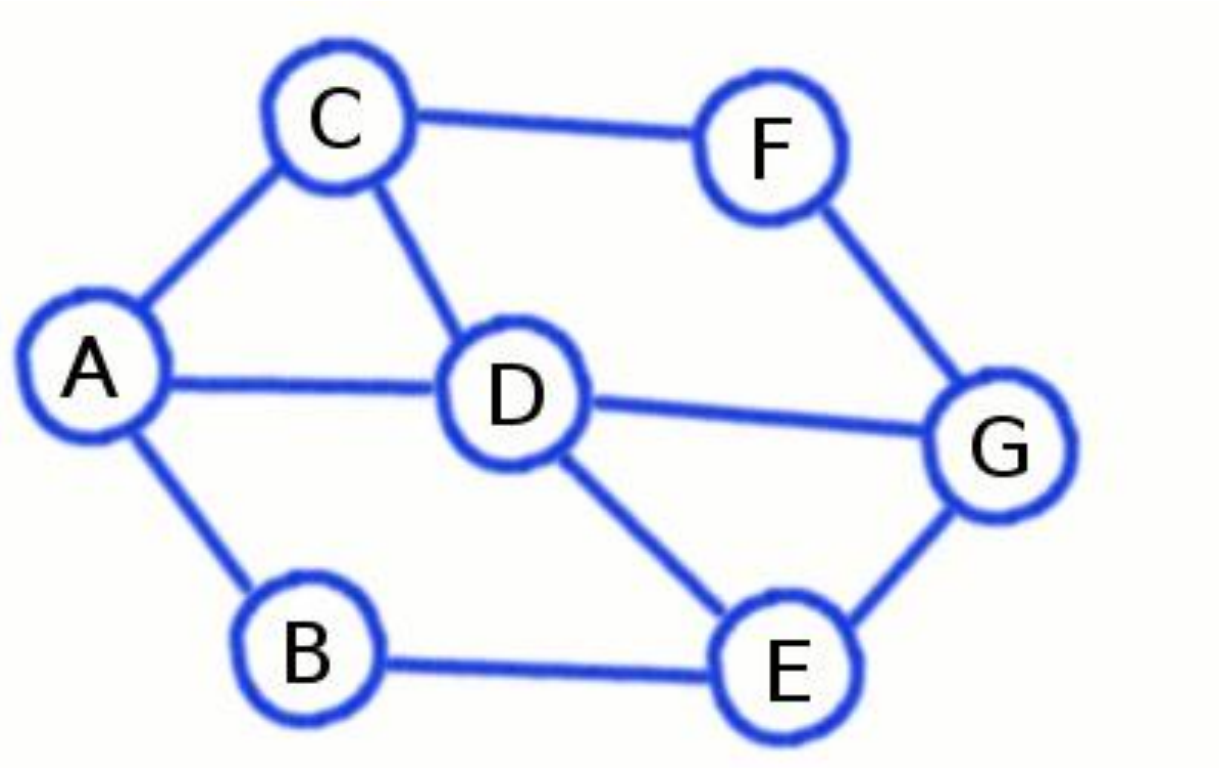




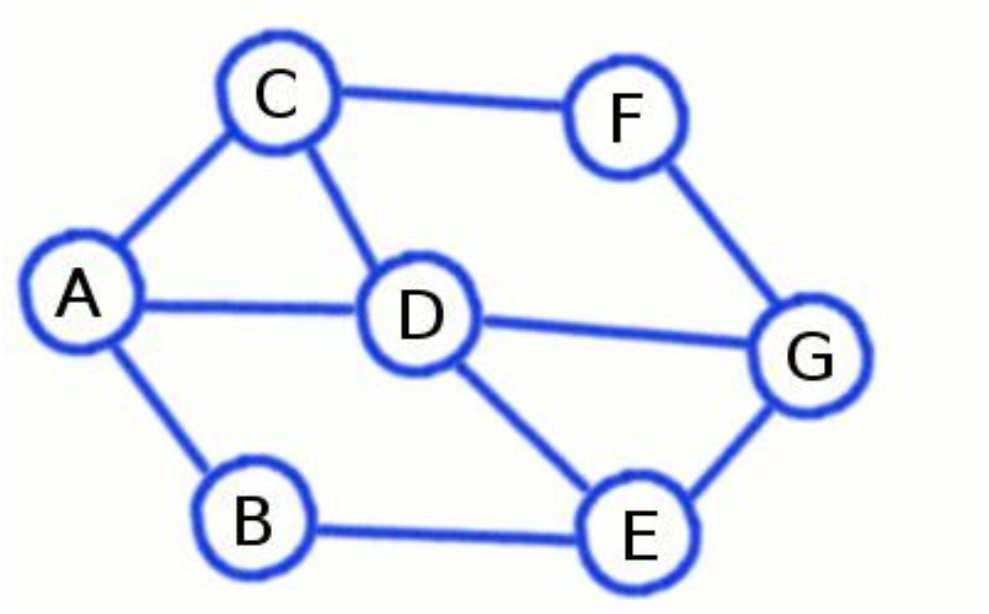


Graph Traversals

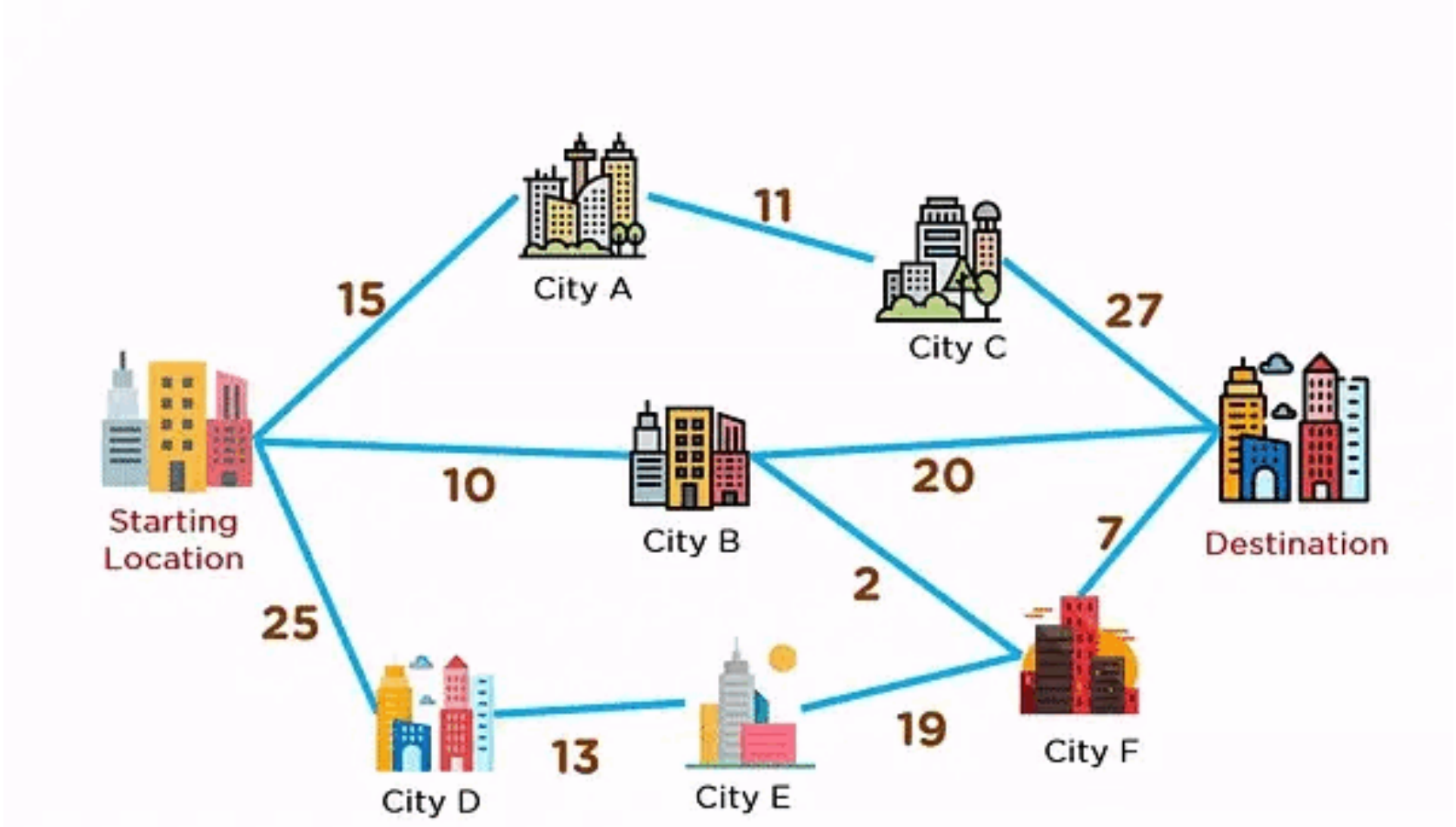
- **Depth First Search**: This is a graph traversal algorithm that explores a graph as far as possible along each branch before backtracking. It may be implemented using an explicit stack or by way of recursion for implicit use of a stack to keep track of vertices to visit.



Graph Traversals



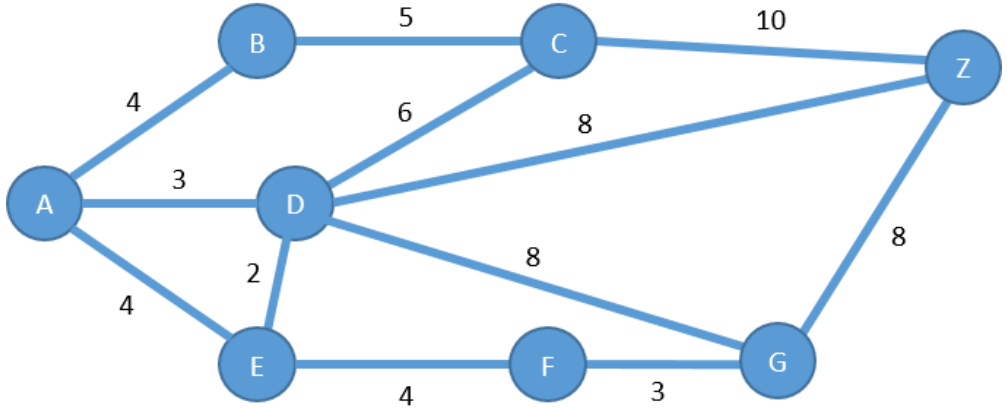
- **Breadth-First Search** or BFS is a graph traversal algorithm that shall traverse neighbors of a node level by level before traversing the next depth level. This is done by keeping tabs on visiting nodes through the linearity of a queue data structure.

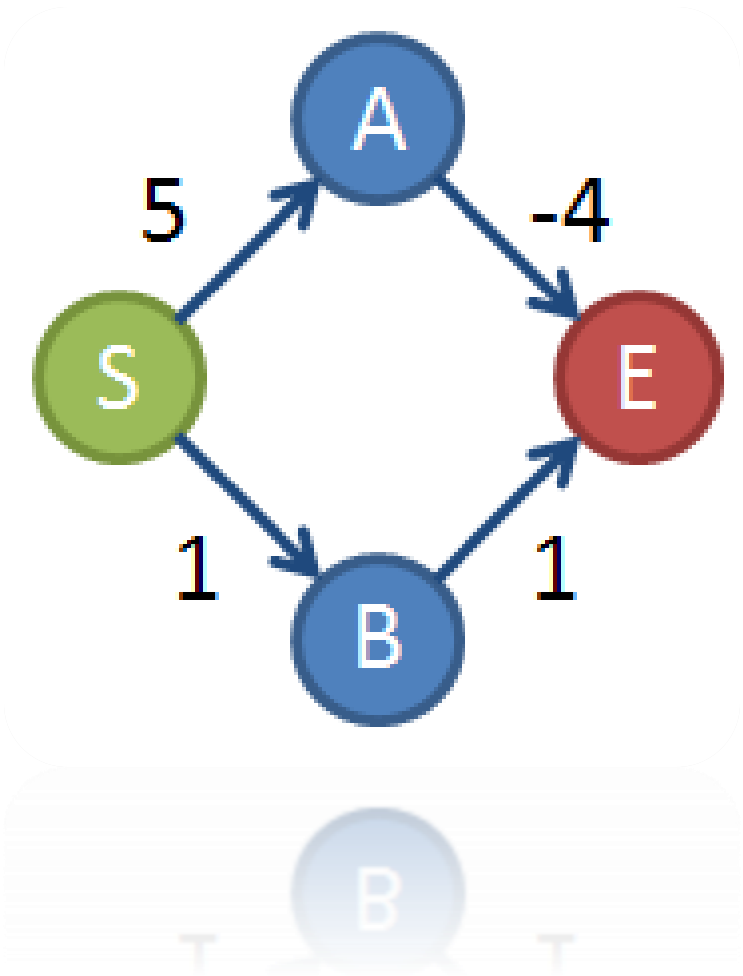


Dijkstra's Algorithm

Dijkstra's Algorithm

- **Dijkstra's algorithm** finds the shortest path from a single-source node, in a directed or undirected graph, with non-negative edge weights.





Dijkstra's Algorithm

Downsides of Dijkstra's algorithm:

Inefficient for dense graphs:

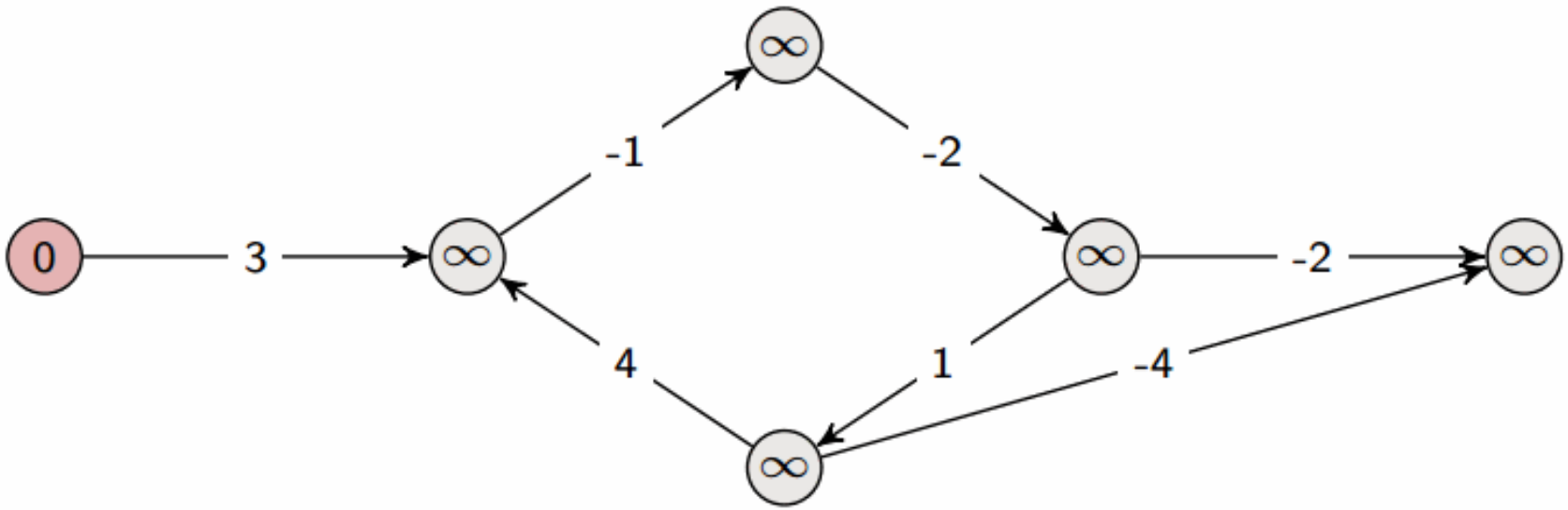
High time complexity if many edges are present.

No negative weights:

Fails with negative edge weights.

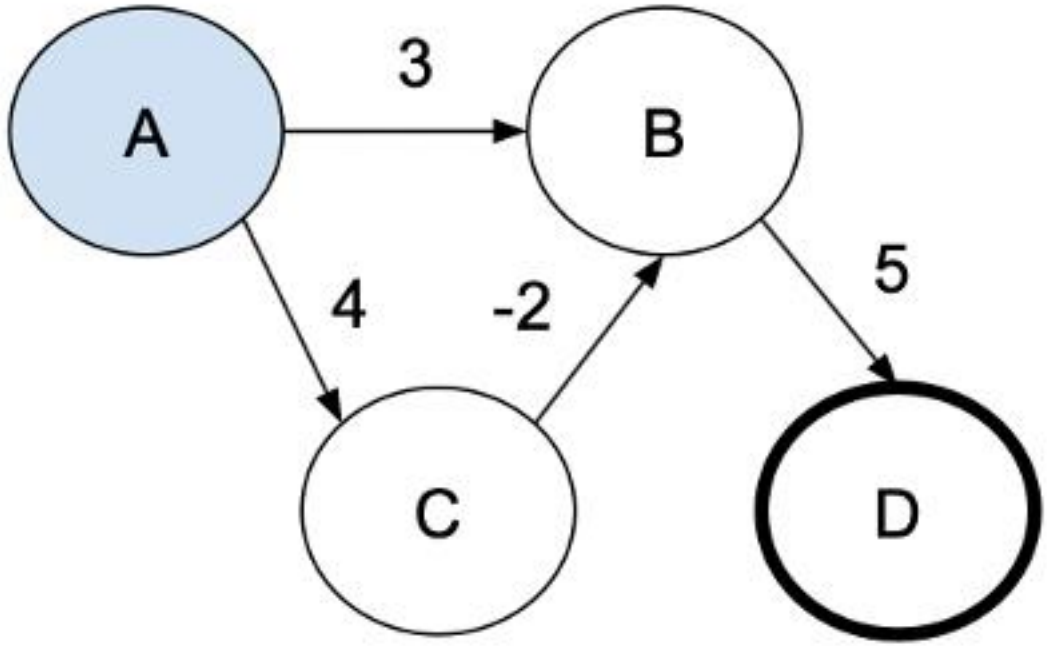
Memory-intensive:

Big graphs demand much storage.



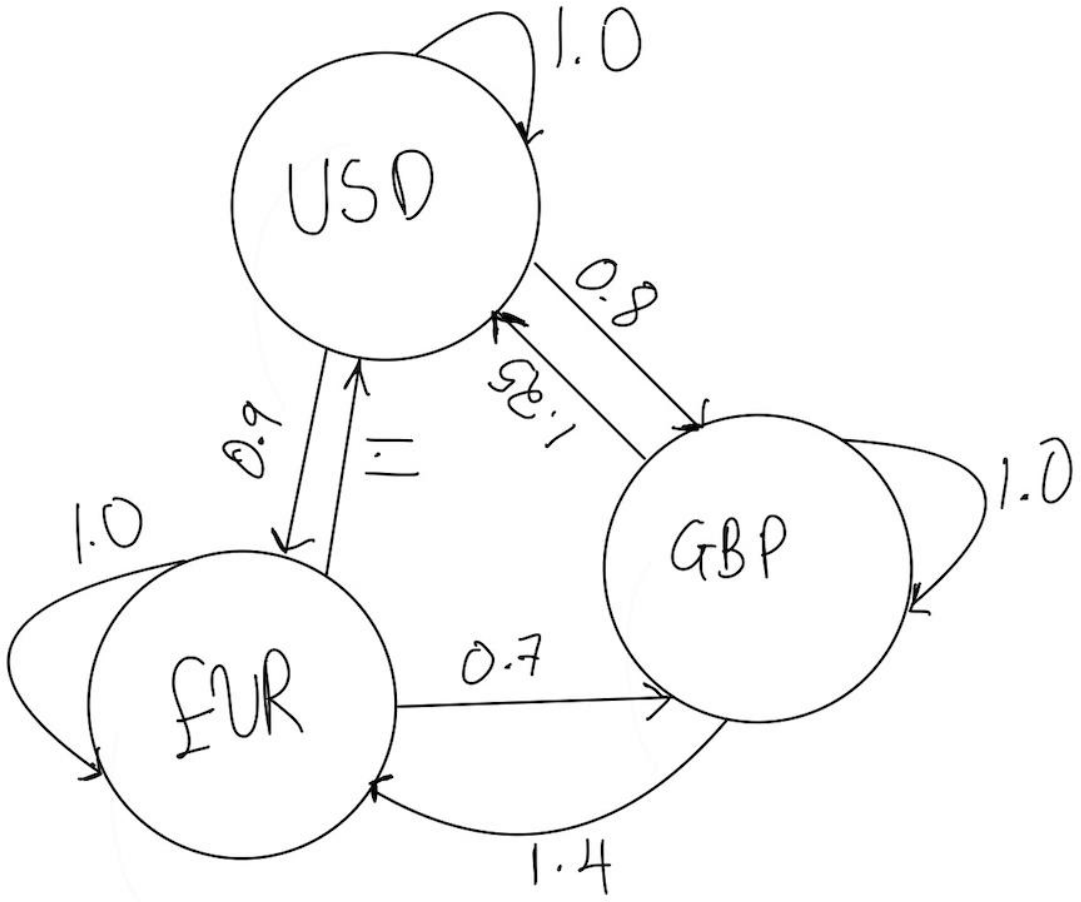
Bellman Ford Algorithm

Why Bellman Ford Algorithm?



More general, as it handles negative weights and detects negative cycles.

Addressed more abstract problems where constraints on edge weights aren't guaranteed.

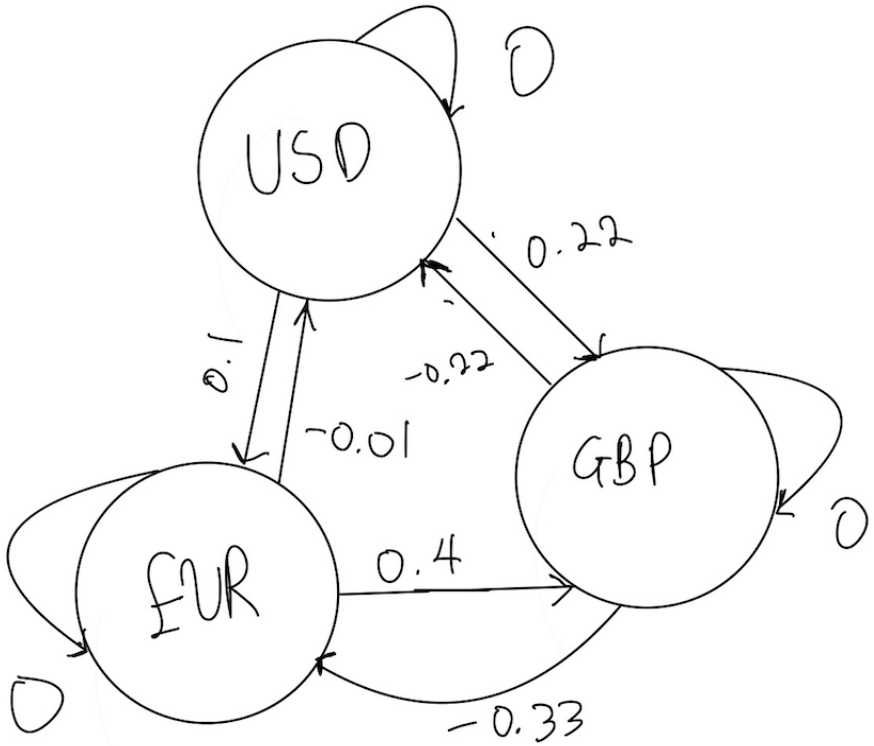


	USD	EUR	GBP
USD	1.0	0.9	0.8
EUR	1.1	1.0	0.7
GBP	1.25	1.4	1.0

Bellman Ford Algorithm and Currency Arbitrage

Bellman Ford Algorithm and Currency Arbitrage

Graph after performing –log(E)



	USD	EUR	GBP
USD	0	0.1	0.22
EUR	-0.01	0	0.40
GBP	-0.22	-0.33	0

Bellman Ford Algorithm and Currency Arbitrage

- **Distances after relaxation 1:**

  0.000000 ,-0.113329, 0.223144

- **Distances after Final relaxation :**

  -0.208639 ,-0.113329, 0.223144

- **Final distances after all relaxations:**
  -0.208639, -0.113329 , 0.223144

Notice how the distances changed

Cycle: GBP -> EUR -> USD

Starting with $1, you can end with $1.23

this can be repeated infinitely!.

Q&A

Thank you!

Scan for repo with Bellman-Ford implementation and slide.