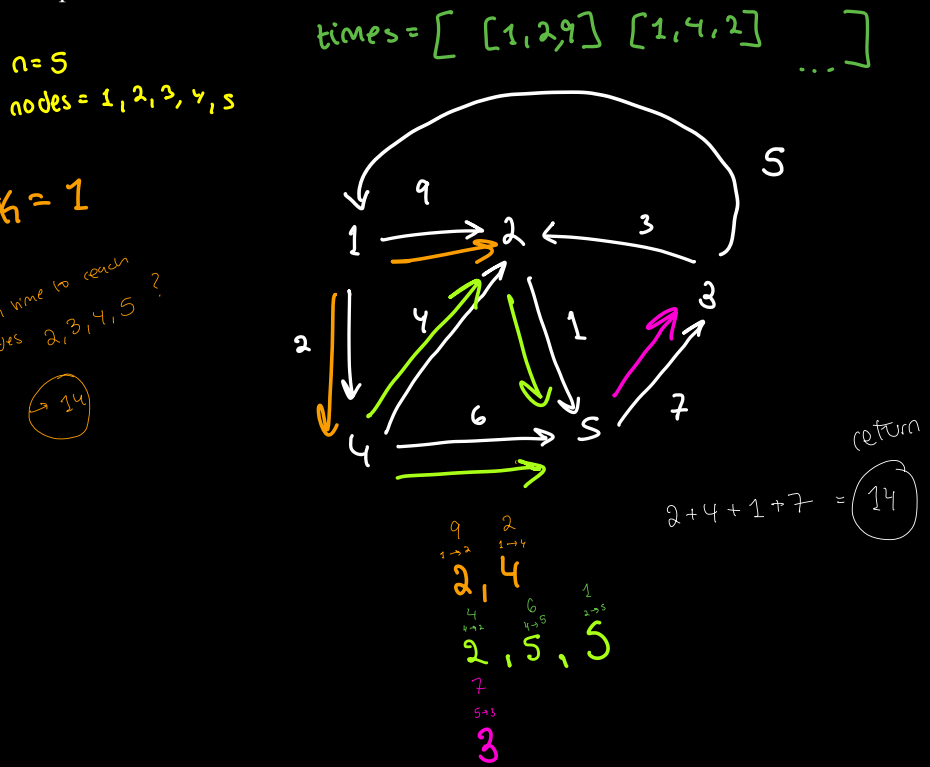
Problem: n network nodes labelled 1 to N
given a times array, containing edges represented by arrays [u, v, w]

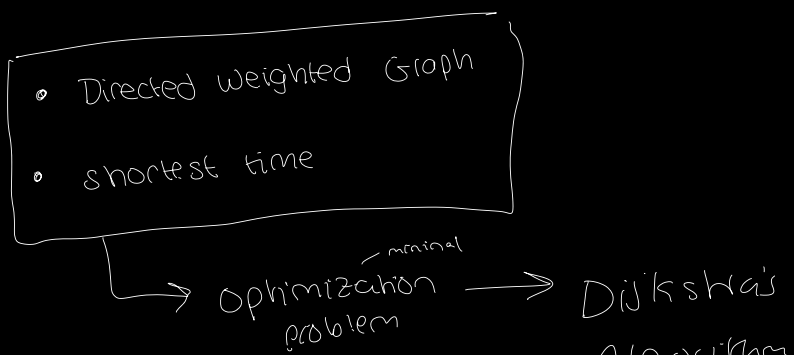
- u: src node
- v: target node
- w: weight - time taken to travel from src to target node

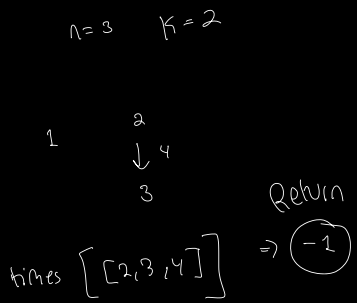
send signal from node k, return how long it takes for all nodes to receive the signal
return -1 if impossible

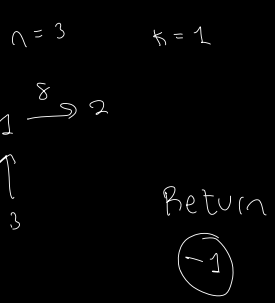


Intro:

- Verify Constraints
  - Unconnected? yes
  - negative time? no
- Create Testcases





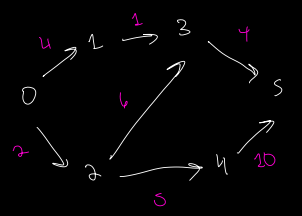


Dijkstra's Algorithm — greedy method

algorithmic paradigm

graphs
- directed
- weighted

optimization problems — max or min



closed

0 : 0

1 : ∞ 4 closed

2 : ∞ 2 closed

when making decision always pick MIN or MAX

3 : ∞ $0 \xrightarrow{2} 2 \xrightarrow{6} 3 \Rightarrow 2+6=8$ $0 \xrightarrow{4} 1 \xrightarrow{1} 3 = 5$

4 : ∞ $0 \xrightarrow{2} 2 \xrightarrow{5} 4 \Rightarrow 2+5=7$ closed

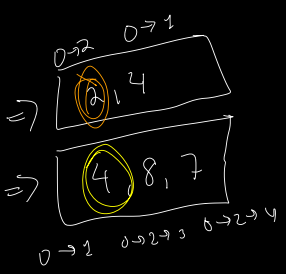
5 : ∞ $0 \xrightarrow{4} 1 \xrightarrow{1} 3 \xrightarrow{4} 5 = 4+1+4=9$

choose min!

figure out distance from 0 to every node

if travel to node → update distance

else leave as infinity



3

4

5

What's the pseudocode?

distance array → set to Integer.max value

closed boolean array

while distance contains Integer.MAX_VALUE

iterate thru children in vertex's

update distance from array

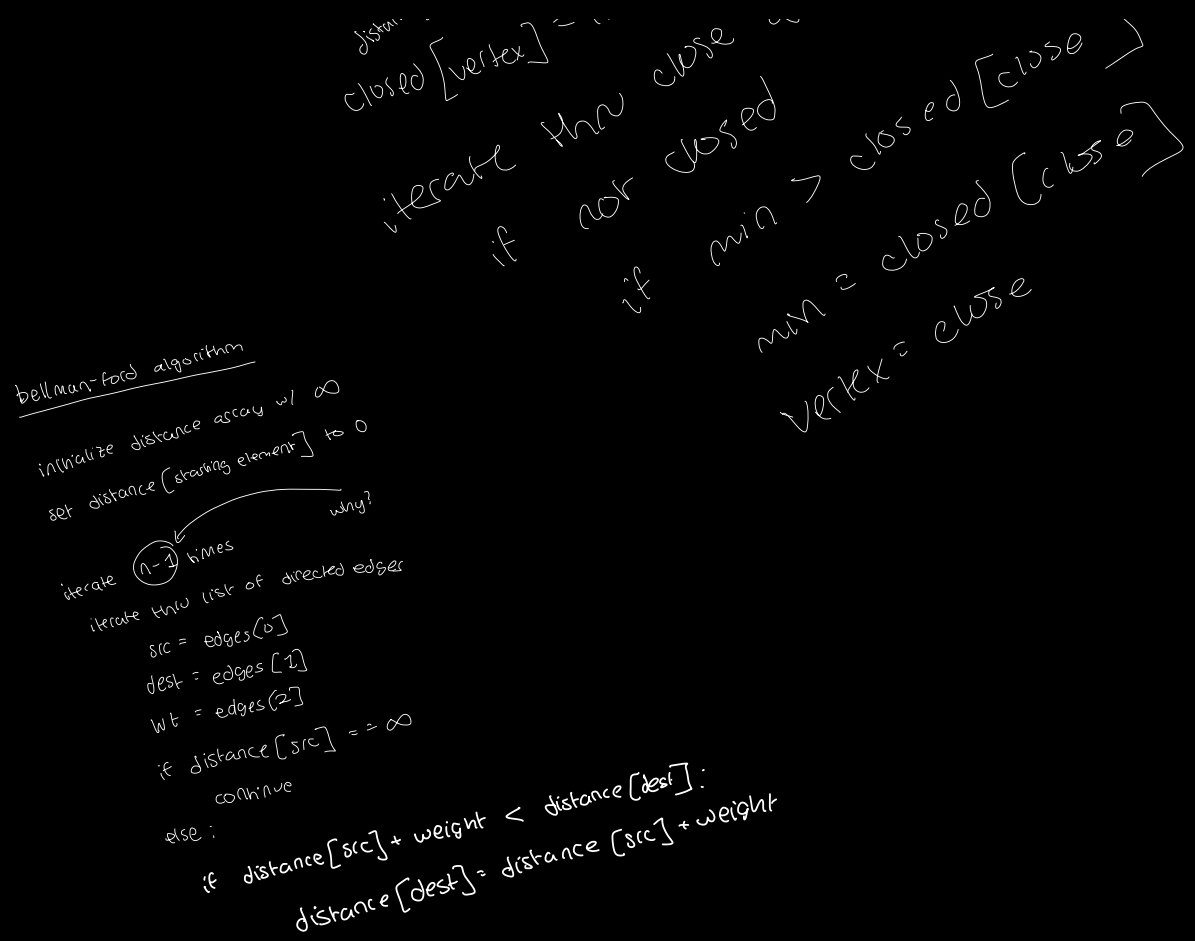
min (vertex to child, current distance)

distance[child] =

arr[vertex] + weight

- True min =

array



Brute Force:

- Brainstorming & Pattern Observations
- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity

Optimal:

- Brainstorming & Pattern Observations
- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity