

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

n = 5

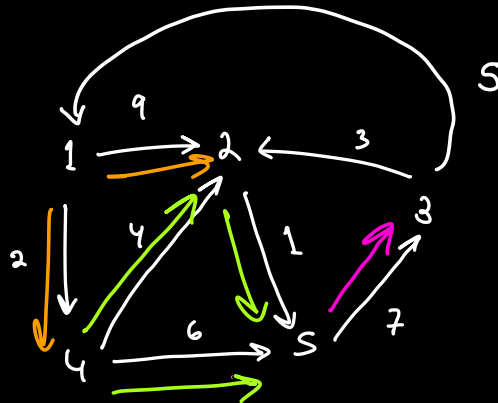
nodes = 1, 2, 3, 4, 5

k = 1

min time to reach
nodes 2, 3, 4, 5 ?

→ 14

times = [[1, 2, 9] [1, 4, 2] ...]



return
 $2 + 4 + 1 + 7 = 14$

9 2
1 → 2 1 → 4
2, 4
4 6
4 → 2 4 → 5
2, 5, 5
7
5 → 3
3

Intro:

- Verify Constraints

unconnected? yes

negative time? no

- Create Testcases

n = 3 k = 2

1 2
↓ 4
3

times [[2, 3, 14]] ⇒ Return -1

n = 3 k = 1

1 8
→ 2
3 ↑
3

Return -1

- Directed weighted Graph
- shortest time

optimization problem → Dijkstra's Algorithm

Dijkstra's Algorithm

greedy method

graphs

- directed
- weighted

algorithmic paradigm

optimization problems

max or min

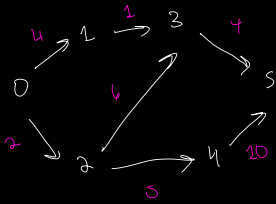


figure out distance from 0 to every node

if travel to node → update distance

else leave as infinity

0	:	0	closed
1	:	∞ 4	closed
2	:	∞ 2	closed

when making decision always pick MIN or MAX

3	:	∞	0 → 2 → 3 = 2 + 6 = 8	0 → 1 → 3 = 5
4	:	∞	0 → 2 → 4 = 2 + 5 = 7	closed
5	:	∞	0 → 1 → 3 → 5 = 4 + 1 + 4 = 9	

choose min!

	$0 \rightarrow 2$	$0 \rightarrow 1$
\Rightarrow	$(2, 4)$	
\Rightarrow	$(4, 8, 7)$	
	$0 \rightarrow 1$	$0 \rightarrow 2 \rightarrow 3$
		$0 \rightarrow 2 \rightarrow 4$

3

4

5

What's the pseudocode?

set to Integer.MAX_VALUE

distance array

closed boolean array

while distance contains

Integer.MAX_VALUE

iterate through children in vertex's

distance[child] =

update distance from array

min (vertex to child, current distance)

min [vertex] + weight

- True min = array

bellman-ford algorithm

initialize distance array w/ ∞
set distance[starting element] to 0

iterate $(n-1)$ times
iterate thru list of directed edges

src = edges[0]

dest = edges[1]

wt = edges[2]

if distance[src] == ∞

continue

else:
if distance[src] + weight < distance[dest]:
distance[dest] = distance[src] + weight

distm.
closed[vertex] = 1

iterate thru close ~
if not closed ~

if min > closed[close]

min = closed[close]

vertex = close

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

