

- **Weeks 1–2: informal introduction**

- network = path



- **Week 3: graph theory**

- **Weeks 4–7: models of computing**

- what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**

- what cannot be computed (efficiently)?

- **Week 12: recap**

# Week 6

- CONGEST model:  
bandwidth limitations

# CONGEST model

- **LOCAL model: arbitrarily large messages**
- **CONGEST model:  $O(\log n)$ -bit messages**

# CONGEST model

- **Any of these can be encoded in  $O(\log n)$ -bit messages:**
  - node identifier
  - number of nodes
  - number of edges
  - distance between two nodes ...

# CONGEST model

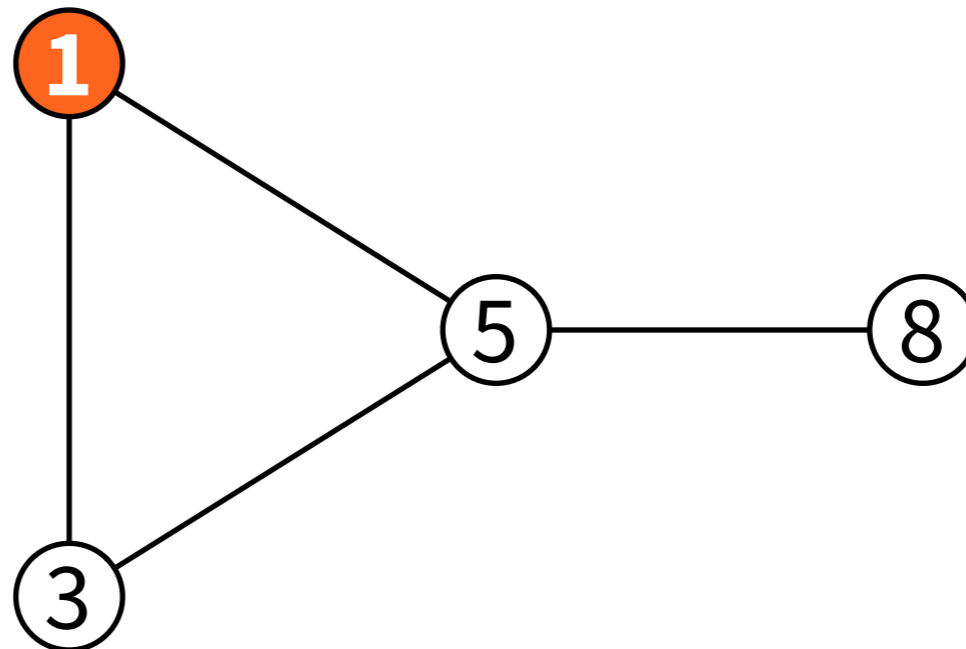
- **Many algorithms that we have seen only send small messages**
  - can be used directly in the CONGEST model
- **Exception: algorithm Gather**
  - may need to send  $O(n^2)$ -bit messages

# CONGEST model

- **$O(n)$  time trivial in the LOCAL model**
  - brute force approach: Gather + solve locally
- **$O(n)$  time non-trivial in the CONGEST model**
- **Today: how to find all-pairs shortest paths in  $O(n)$  time**

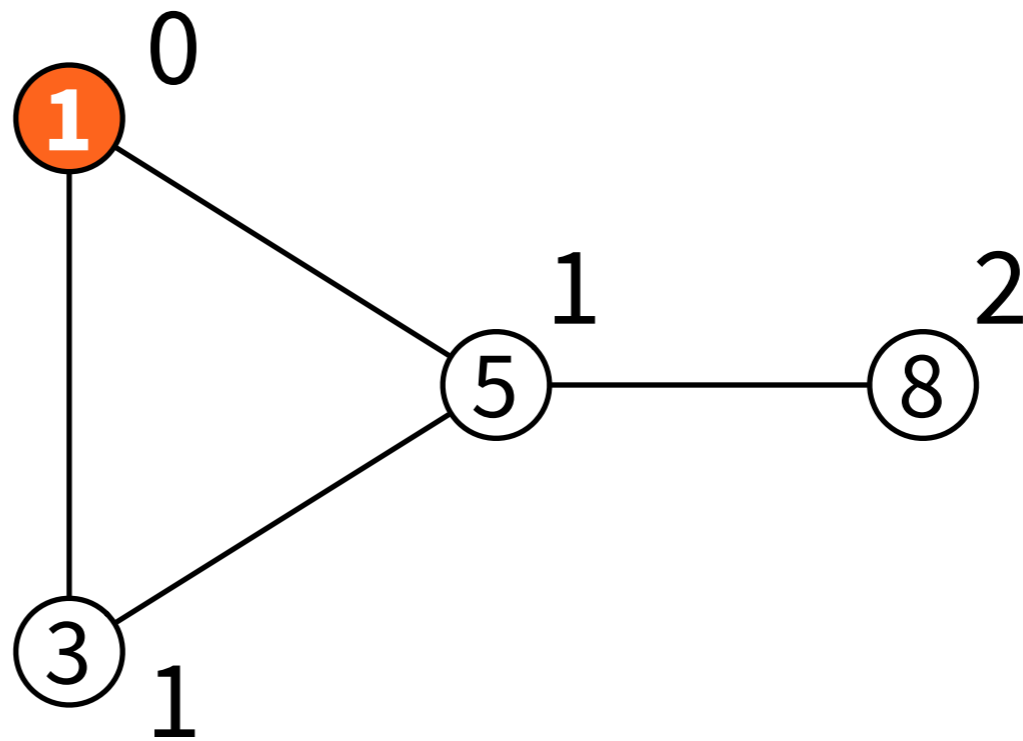
# Single-source shortest paths

**Input:**



# Single-source shortest paths

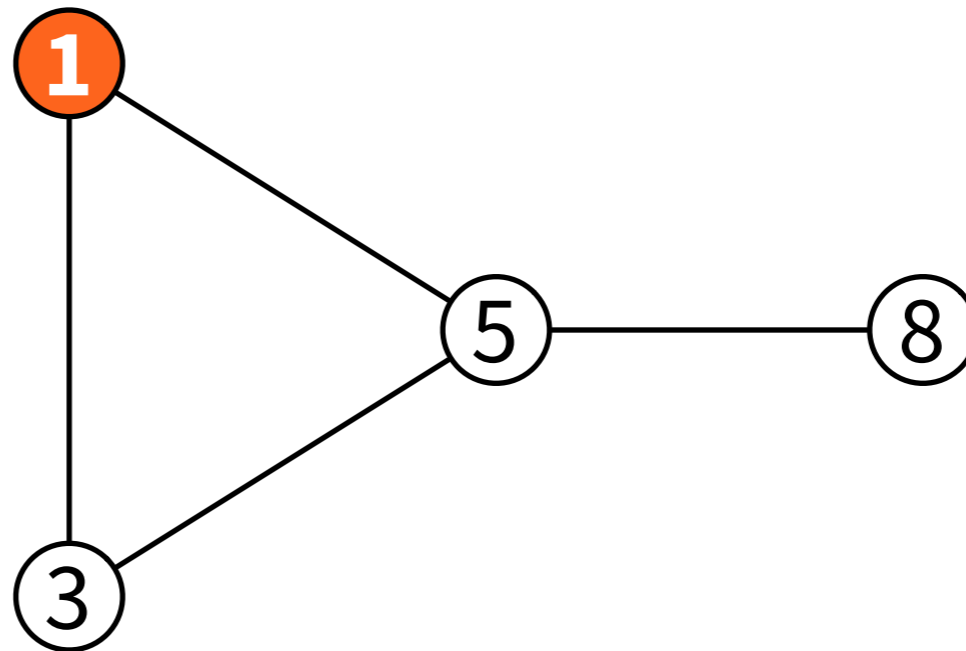
**Output:**





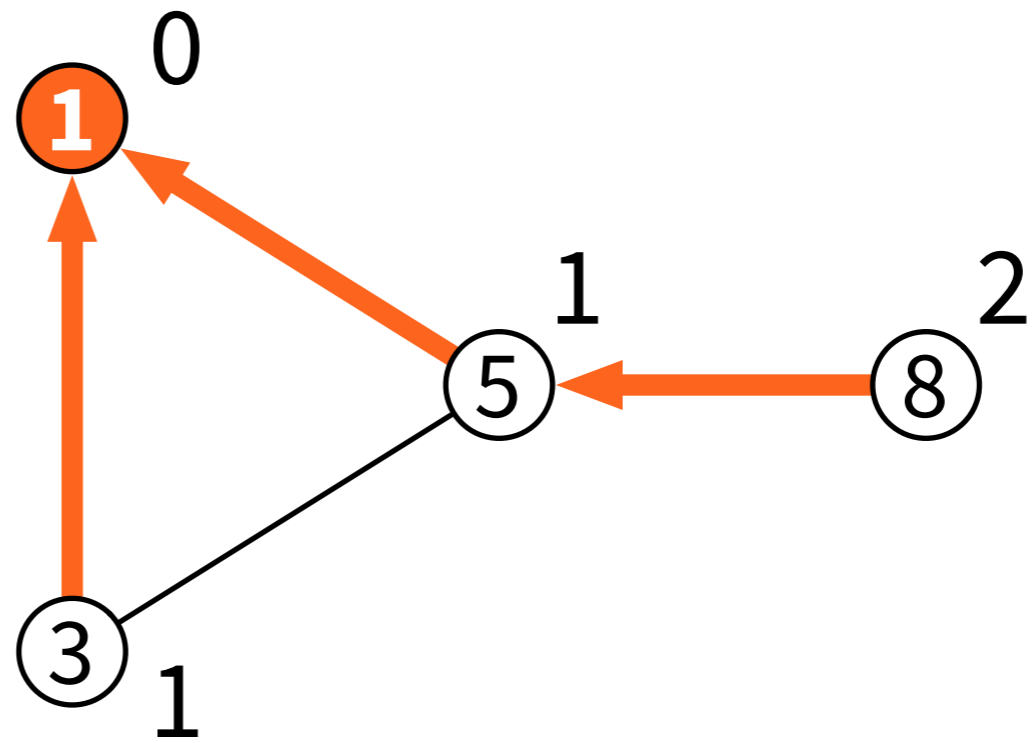
# BFS tree

**Input:**



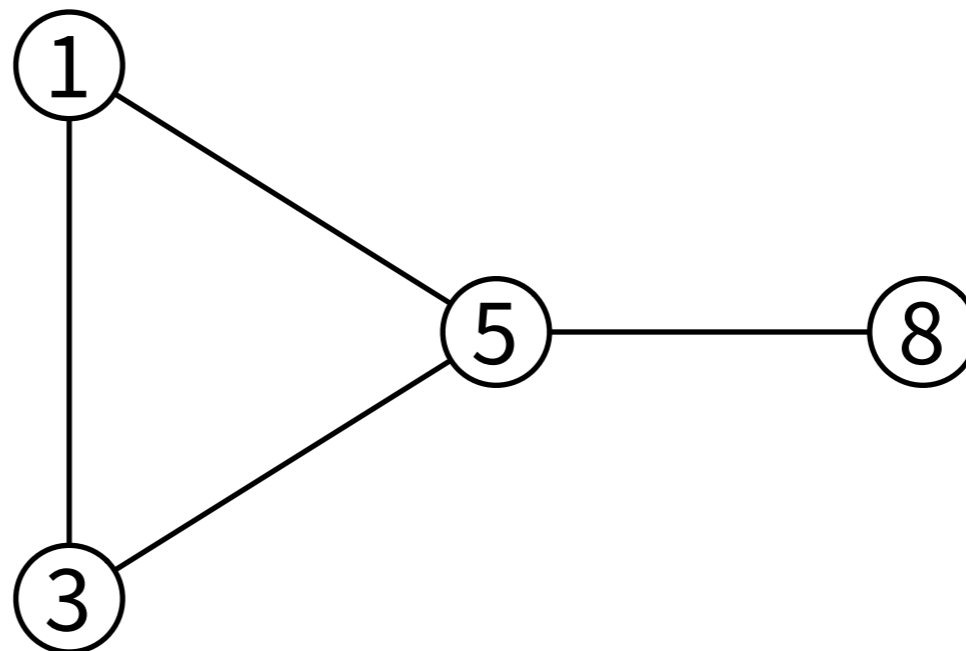
# BFS tree

**Output:**



# All-pairs shortest paths

**Input:**



# All-pairs shortest paths

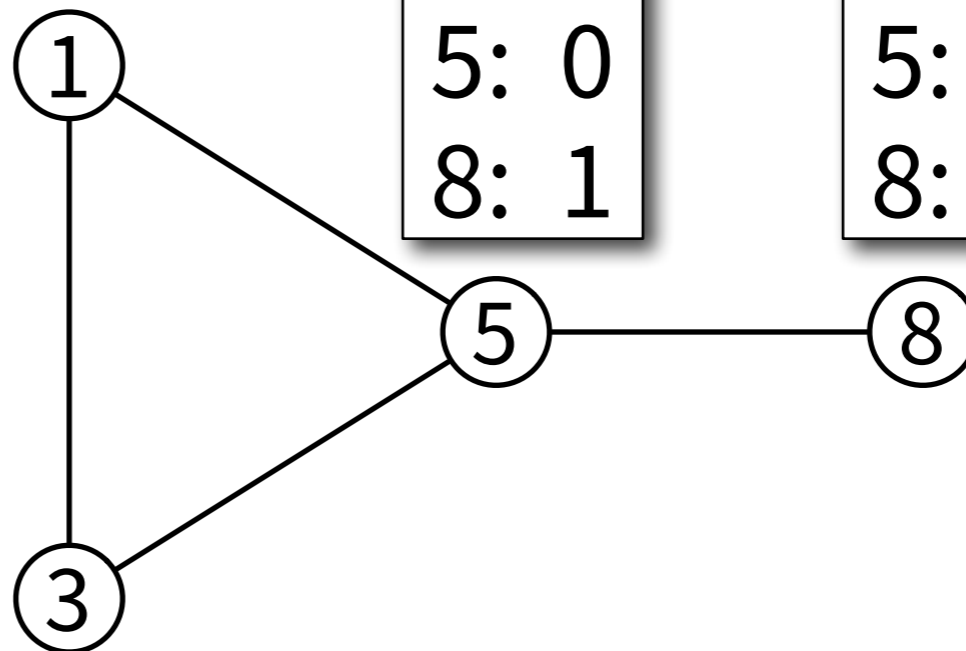
**Output:**

1:	0
3:	1
5:	1
8:	2

1:	1
3:	1
5:	0
8:	1

1:	2
3:	2
5:	1
8:	0

1:	1
3:	0
5:	1
8:	2

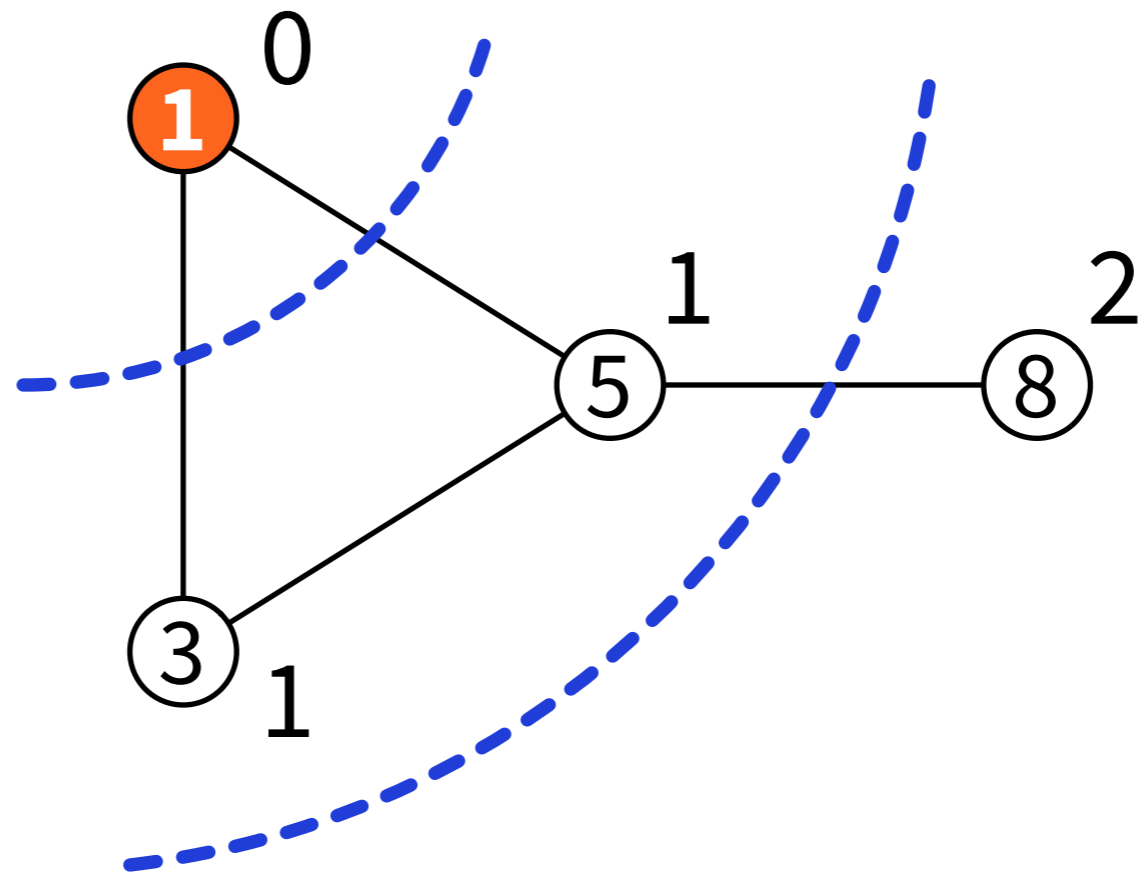


# Algorithm Wave

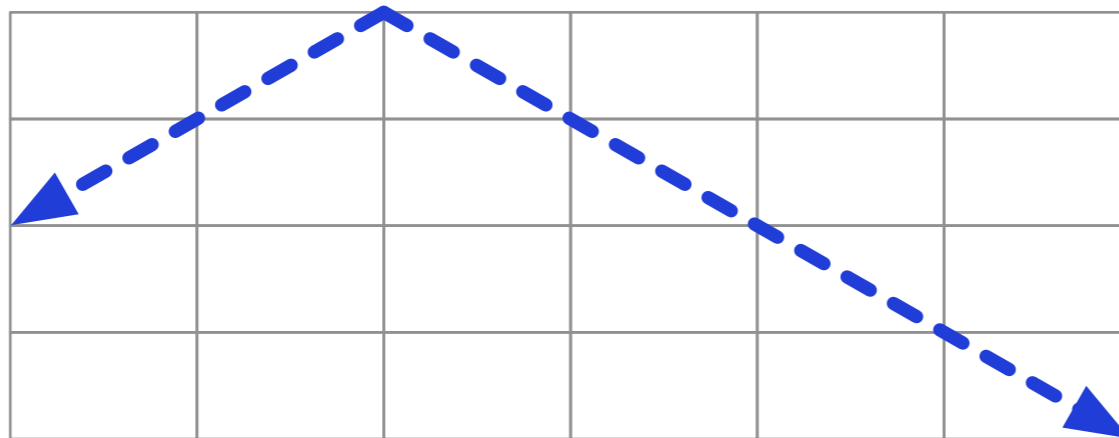
- **Solves single-source shortest paths in time  $O(\text{diam}(G))$**
- **Leader creates a ‘wave’, other nodes propagate it**
- **Wave first received in round  $t$ : distance to leader is  $t$**

# Algorithm Wave

**Output:**



# Algorithm Wave



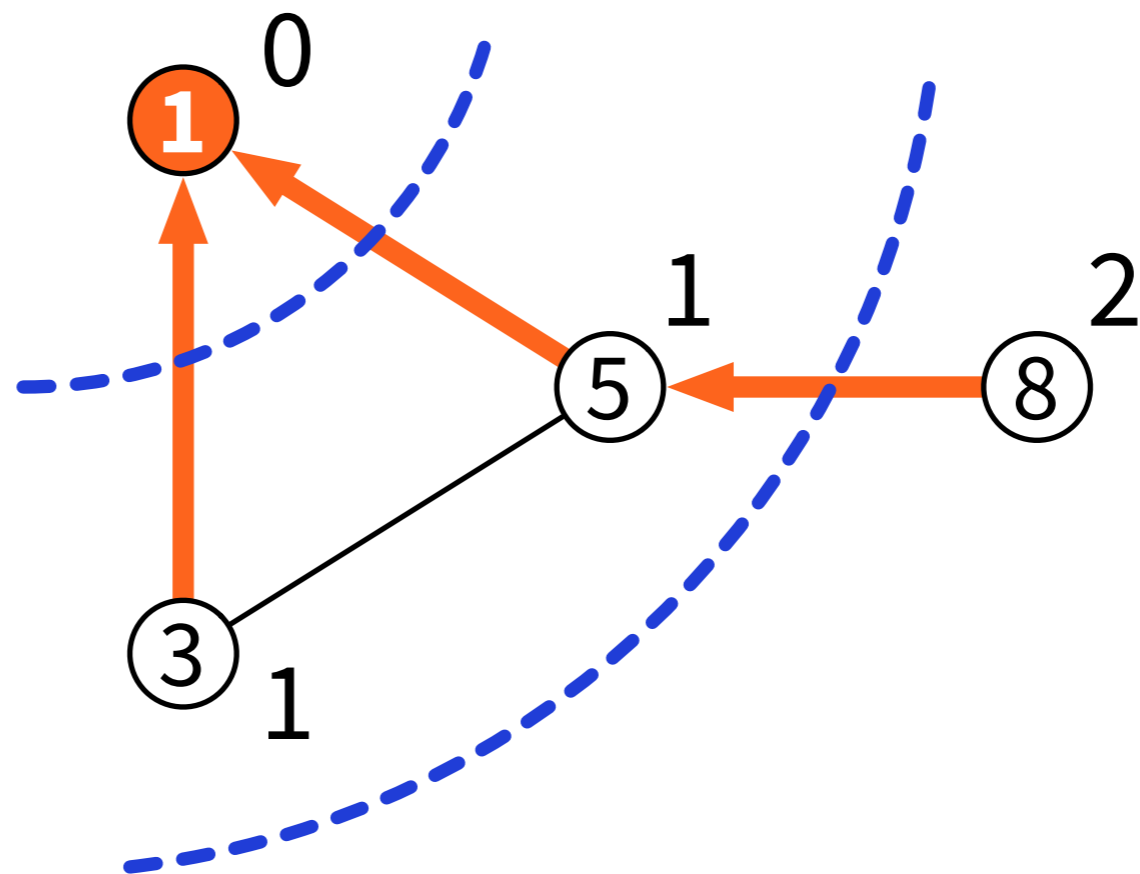
# Algorithm BFS

- **Wave + handshakes**
- **Tree construction:**
  - “proposal” + “accept”
  - everyone knows their parent & children
- **Acknowledgements back from leaf nodes**

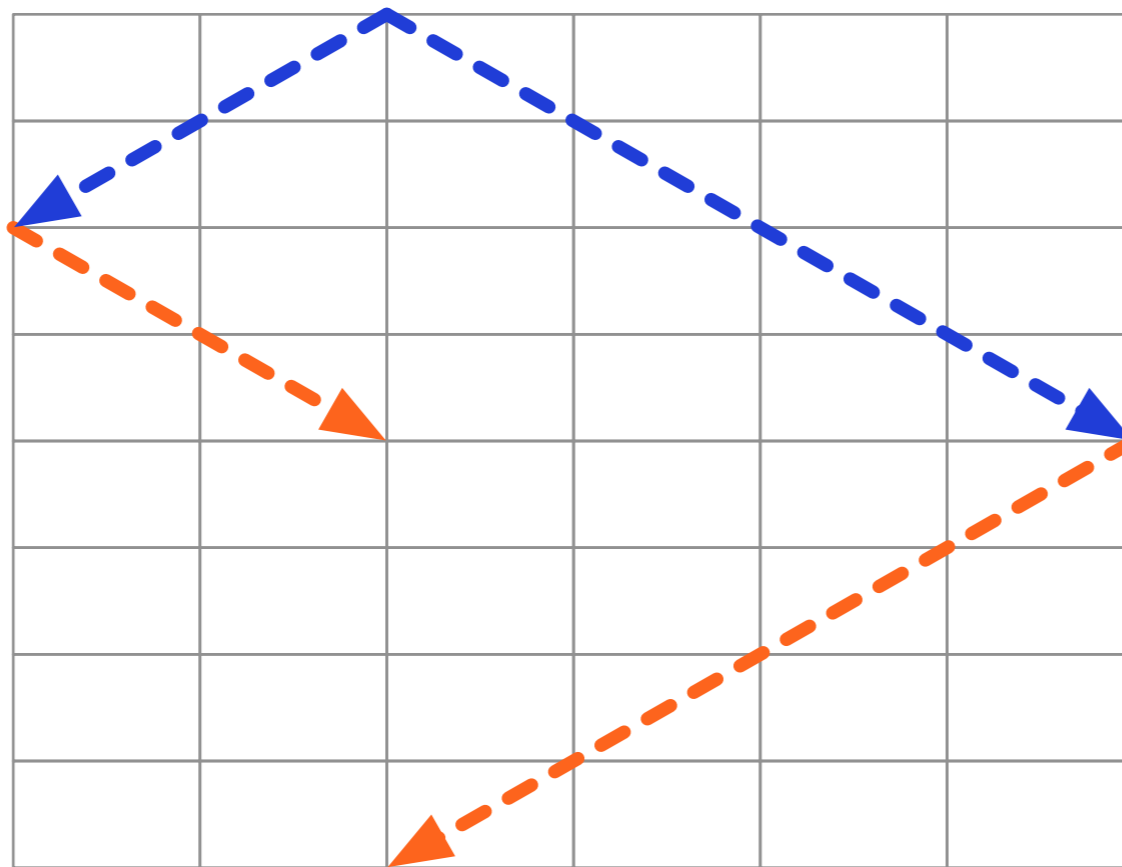


# Algorithm BFS

**Output:**



# Algorithm BFS



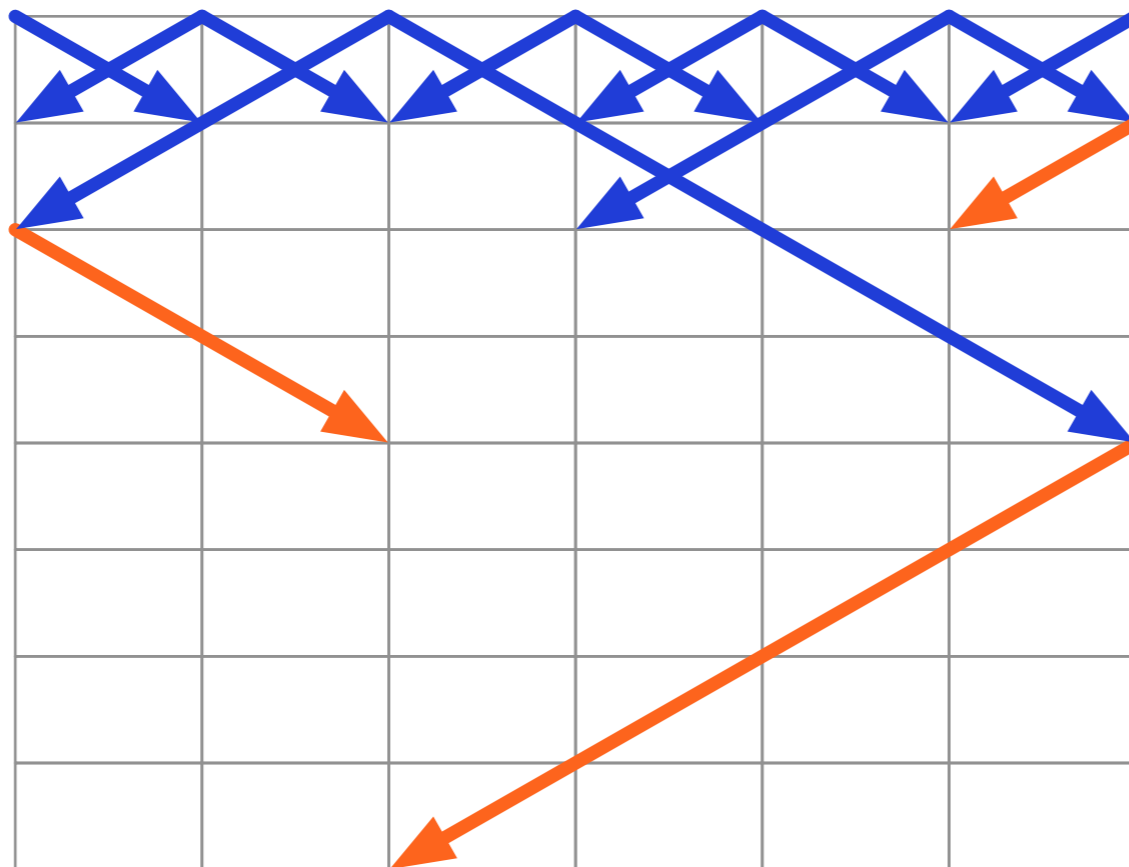
**propose,  
accept**

**ack**

# Algorithm Leader

- **Each node creates a separate BFS process**
- **When two BFS processes “collide”, the one with the smaller root “wins”**
  - each node only needs to send messages related to ***one BFS process***
- **One tree wins everyone else → leader**

# Algorithm Leader



**propose,  
accept**

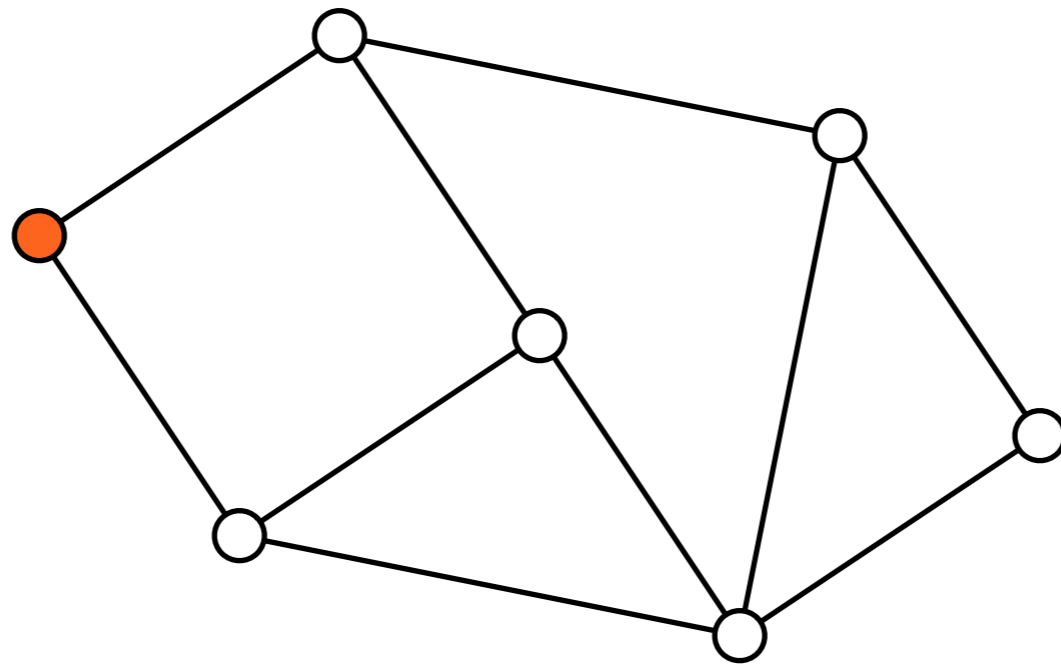
**ack**

# Algorithm APSP

- **Basic idea: run Wave from each node**
- **Challenge: congestion**
  - all waves parallel → too many bits per edge
  - all waves sequentially → takes too long
- **Solution: pipelining**

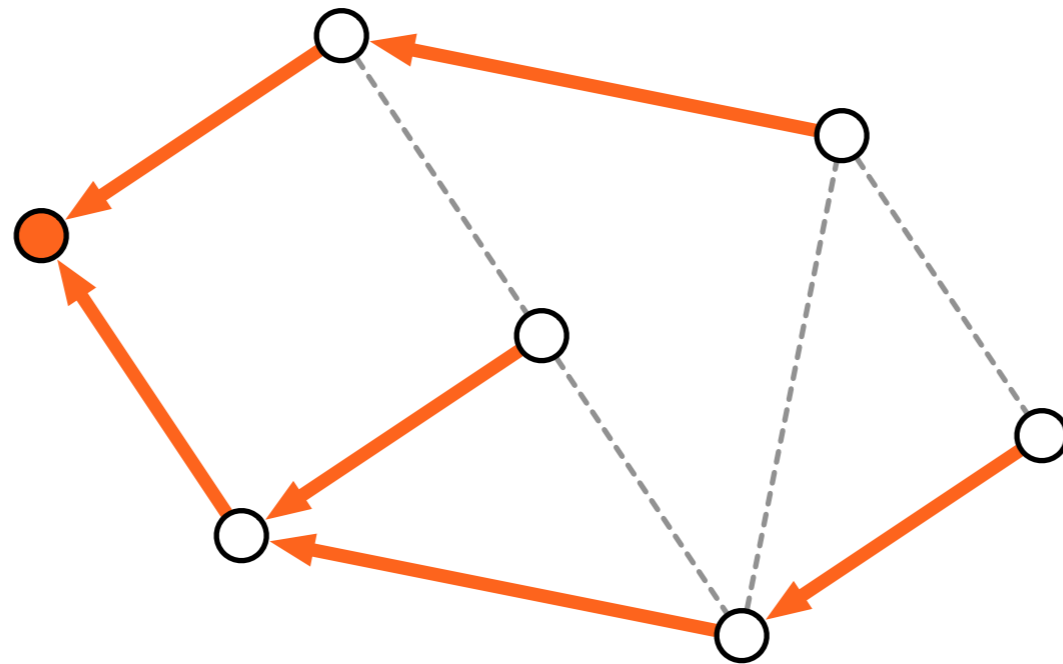
# Algorithm APSP

- **Elect leader**



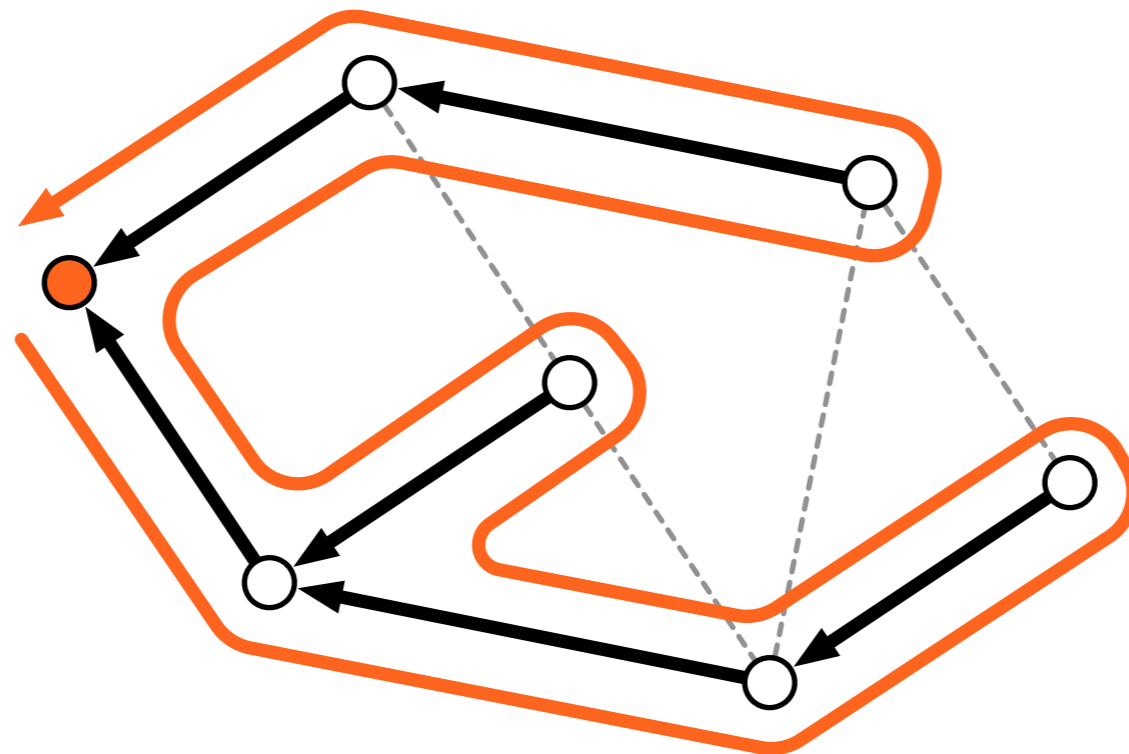
# Algorithm APSP

- **Elect leader, construct BFS tree**



# Algorithm APSP

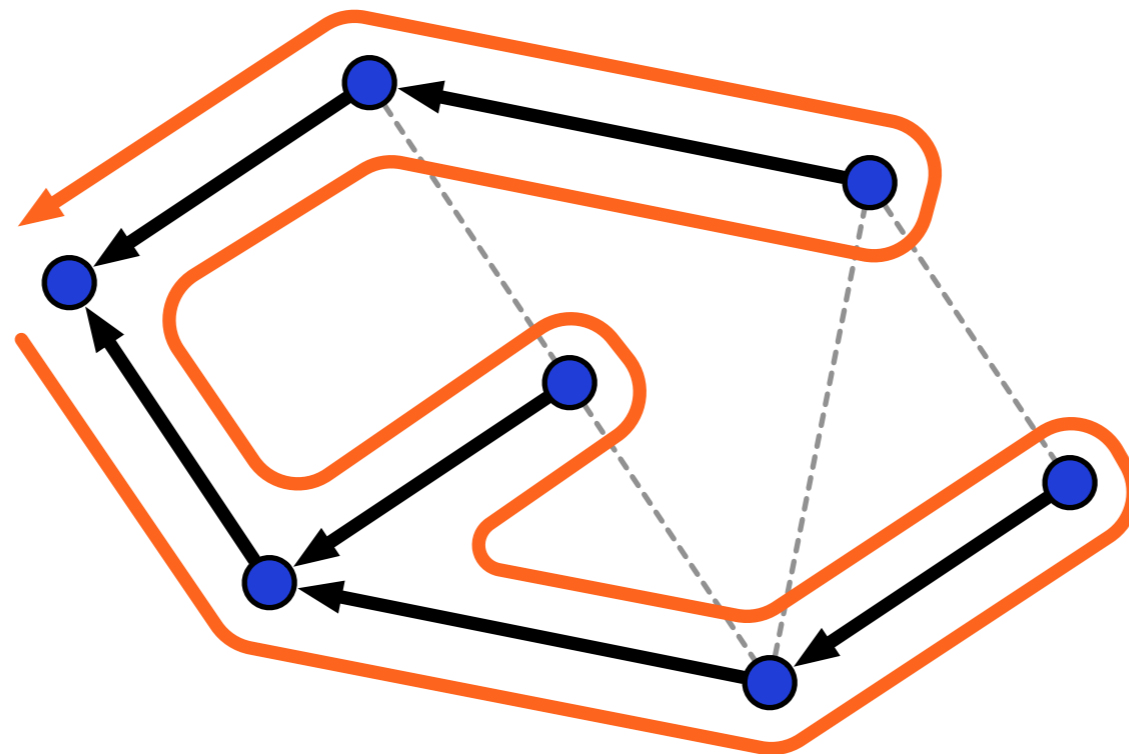
- Move token along BFS tree slowly



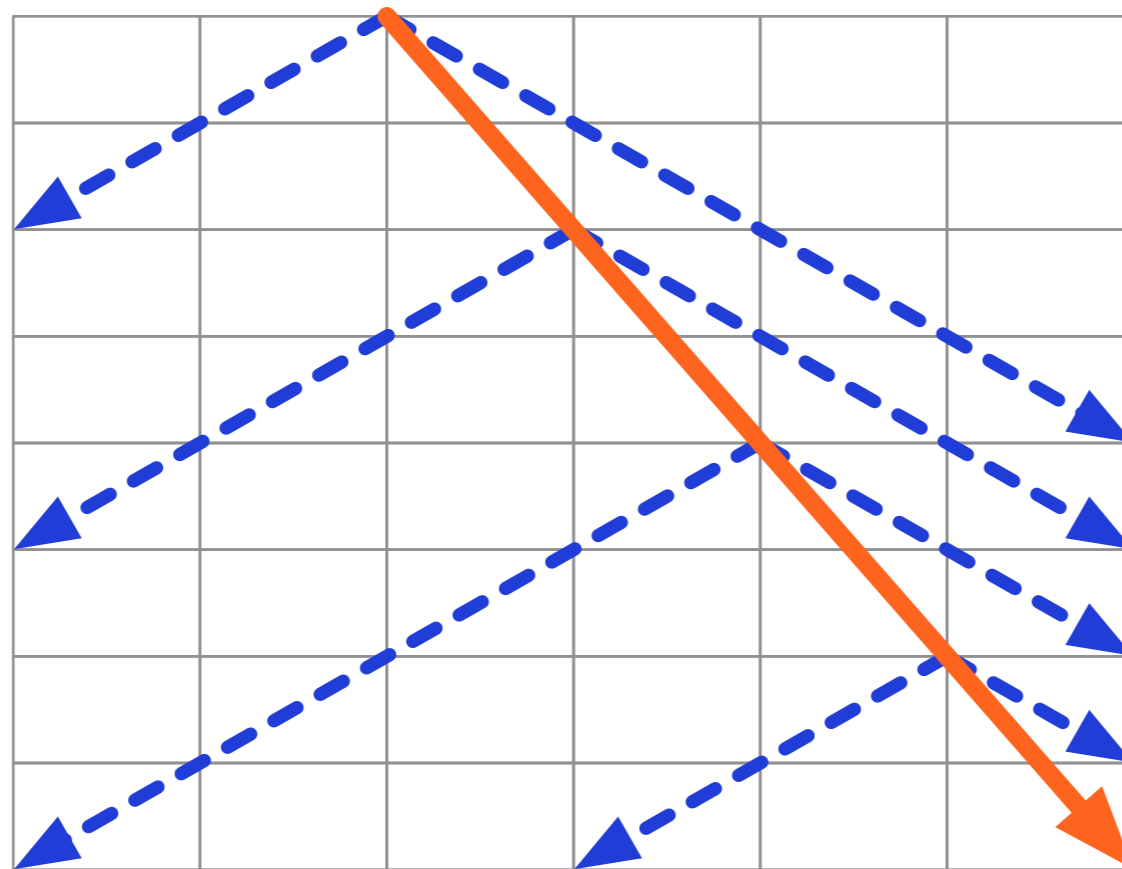


# Algorithm APSP

- Create *wave* every time we visit a new node

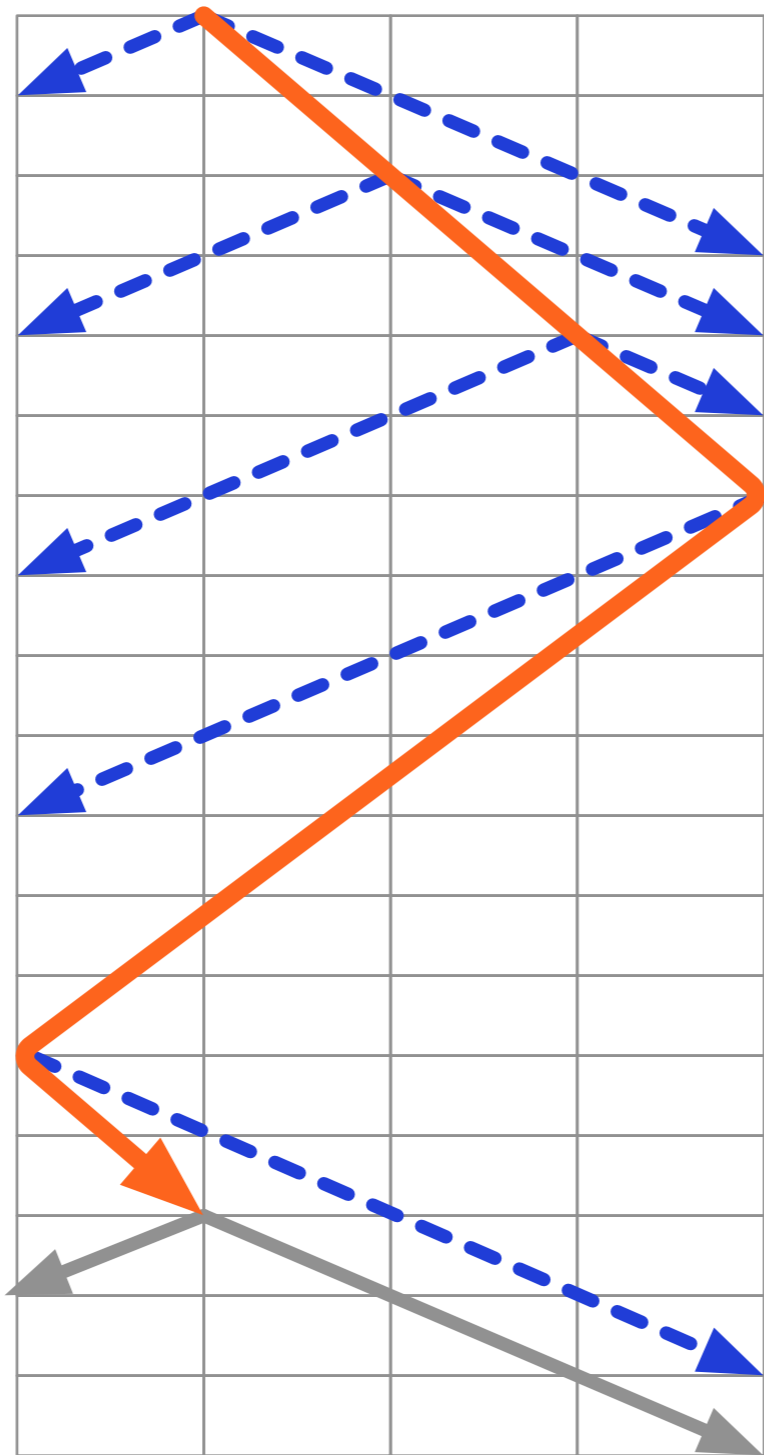


# Algorithm APSP



**wave**

**token**



**wave**

**token**

**all done**

# Algorithm APSP

- **Algorithm animation:**

<http://users.ics.aalto.fi/suomela/apsp/>

# Pipelining

- $n$  operations, each operation takes time  $t$
- **Parallel:**  $t$  rounds, bad congestion
- **Sequential:**  $nt$  rounds, no congestion
- **Pipelining:**  $n + t$  rounds, no congestion

# Summary

- **LOCAL model: unlimited bandwidth**
- **CONGEST model:  $O(\log n)$  bandwidth**
- **$O(n)$  or  $O(\text{diam}(G))$  time is no longer trivial**
- **Example: all-pairs shortest paths in time  $O(n)$ ,  
pipelining helps**

- **Weeks 1–2: informal introduction**

- network = path



- **Week 3: graph theory**

- **Weeks 4–7: models of computing**

- what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**

- what cannot be computed (efficiently)?

- **Week 12: recap**