The slides are loosely based on those of Dr Kevin Wayne, Princeton University and Prof. Mary Wootters, Stanford University
Agenda

1. Path Counting
2. Longest/Shortest Path in DAG
3. Parallel Scheduling
4. Shortest Paths Up to $k$ Hops
Given a DAG, find the number of (not necessarily shortest) paths from a source to a destination
Dynamic Programming
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

Number of paths from source to $\nu$
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

Number of paths from source to \( v \)
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

source

\[ x \] Number of paths from source to \( v \)
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

Number of paths from source to \( v \)
Dynamic Programming

Number of paths from source to $v$
Dynamic Programming

Number of paths from source to $v$
Practice

Find the number of paths from A to every other node

Nodes are no longer indexed 0, 1, ..., so what should we do?
Arbitrary Labeling Won’t Work

Let’s say we index the nodes like this
Arbitrary Labeling Won’t Work

Can’t compute value for node 1, since the value at node 2 is not ready
How To Label?

When we process a node, we want the predecessors to be ready

Topological order
Issues In Topological Ordering

Want:
- A has index 0 (base case)
- Other nodes have indices > 0 (recursive case)

Seems impossible if there are incoming edges to source
Issues In Topological Ordering

Solution: Delete incoming edges to source

Any path coming out of the source won’t return to it (as there’s no cycle) and thus won’t contain those incoming edges
Once Nodes Are Topologically Sorted

0

1

2

3
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Once Nodes Are Topologically Sorted
Agenda

1. Path Counting

2. Longest/Shortest Path in DAG

3. Parallel Scheduling

4. Shortest Paths Up to $k$ Hops
Motivation

Edges are labeled by profits

**Goal:** Find **longest paths** from the source A to maximize profits
Motivation

Edges are labeled by profits

Goal: Find longest paths from the source A to maximize profits

Longest path from A to D
Longest Path: A Pathological Case

What is the longest path from A to D?
For non-negative weights, longest paths are undefined if there are cycles.

We’ll therefore focus on DAG.

What is the longest path from A to D?

Can make profit arbitrarily large by looping $B \rightarrow C \rightarrow E$ many times.
Given a DAG with possibly negative weights, find longest paths from the source to every node
From Longest Path To Shortest Path

Find **longest** paths from A

Find **shortest** paths from A

Negate weights
Recall: Dijkstra Can’t Handle Negative Weights

Incorrect shortest path from A to C!
Shortest-Path-DAG Algorithm

**Idea:** Use dynamic programming

**Issue:** We need an order to do recursion

How to get an order out of a DAG?
Order in DAG: Topological Order

Perform a topological sort on the DAG

Want the source to have order 1:
- This should be the base case, as we already know the shortest path from source to itself
**Issue In Topological Sort**

Want the source to have order 1

Seems impossible if there are incoming edges to source

**Solution:** Delete incoming edges to source
Issue In Topological Sort

Want the source to have order 1

Seems impossible if there are incoming edges to source

Solution: Delete incoming edges to source

Any path coming out of the source won’t return to it (as there’s no cycle) and thus won’t contain those incoming edges
Dynamic Programming in DAG

Topologically sorted

dist[i]: distance from source to node i
pred[i]: predecessor in the shortest path from source to node i
Recurrence

To reach node $i$, must go to one incoming neighbor of $i$ first

$d[u] = 50$

$d[v] = 45$

$d[i] = \min\{50 + 7, 45 + 15\}$

predecessor of $X$ is $Y$

\[
d[i] = \begin{cases} 
0 & \text{if } i = 1 \\
\min_{(u,i) \in E} \{d[u] + w(u, i)\} & \text{otherwise}
\end{cases}
\]
Shortest-Path-DAG via An Example
Shortest-Path-DAG via An Example

![Diagram showing a directed acyclic graph with nodes and edges labeled with weights. The graph includes nodes 0, 1, 2, 3, 4, and 5, with weights indicated on the edges.]
Shortest-Path-DAG via An Example

predecessor of X is Y
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG via An Example

predecessor of $X$ is $Y$
Shortest-Path-DAG: Implementation

**procedure** Shortest-Path-DAG($V, E, w, s$)

Delete incoming edges to $s$. Let $n \leftarrow |V|$

Topologically sort the nodes, with $s$ of order 1.

```plaintext
for $i \leftarrow 1$ to $n$ do $d[v] \leftarrow \infty$; pred[$v$] $\leftarrow$ NULL  // Initialization

$d[1] \leftarrow 0$

for $i \leftarrow 2$ to $n$ do //

  for $(u, i) \in E$ do // Look at incoming neighbors

    if $d[i] > d[u] + w(i, u)$ then // If new dist is smaller, update

      $d[i] \leftarrow d[u] + w(u, i)$; pred[$i$] $\leftarrow u$
```

**Time:** $O(m + n)$ ← $n$ nodes and $m$ edges
Agenda

1. Path Counting

2. Longest/Shortest Path in DAG

3. Parallel Scheduling

4. Shortest Paths Up to $k$ Hops
Parallel Scheduling

Time to execute job A is 15

Given $n$ jobs and unlimited number of cores, how to schedule them to minimize total latency?

Job C must finish before you can run job D
Parallel Scheduling

Total latency: 85
Parallel Scheduling

Time to execute job A is 15

Given $n$ jobs and unlimited number of cores, how to schedule them to minimize total latency?

Job C must finish before you can run job D
Assigning Weights

10

B

A

15

C

35

D

E

20

50

10

B

A

15

C

35

D

E

35

35

35

10

58
Adding Source and Destination

Starting time

Finishing time
Bounding Time Via Paths

This path means that the running time must be at least 15 + 10 + 20 = 45
The **longest path** from \( s \) to \( t \) is the running time for the best scheduling.
Finding Longest Paths in DAG

- Longest cost to \( v \) is \( x \)
- Predecessor of \( X \) is \( Y \)
Parallel Scheduling

Total latency: 85
Agenda

1. Path Counting

2. Longest/Shortest Path in DAG

3. Parallel Scheduling

4. Shortest Paths Up to $k$ Hops
Give a graph without negative cycles, find the shortest path from the source to the destination using at most \( k \) edges
Intuition: Replicate Graph

Dashed edges have weight 0

Only consider up to $k$ hops away
Intuition: Replicate Graphs

Shortest path in the original graph becomes a shortest path in a DAG

How to carry out this intuition (without the expensive graph transform)?
Dynamic Programming:

\[ d[i, v] : \text{distance from A to } v \text{ using at most } i \text{ edges} \]

\[ \text{pred}[v] : \text{predecessor of } v \text{ in the shortest path from A to } v \text{ found so far} \]
Shortest Paths Up To $\kappa$ Hops

- **Source:** A
- **Destinations:** B, C, D, E

**Table: Shortest Distances**

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shortest Paths Up To $\kappa$ Hops

![Graph Diagram]

### Table: Shortest Path Distances

<table>
<thead>
<tr>
<th>Round</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Predecessor of X is Y
Shortest Paths Up To $k$ Hops

Source: A

<table>
<thead>
<tr>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shortest Paths Up To $k$ Hops

![Diagram showing a graph with nodes A, B, C, D, and E. The edges are labeled with weights: A-B: 4, A-C: 3, B-D: 20, B-E: 8, C-D: 2, C-E: -5, D-E: ∞, A-D: -2. The table shows the shortest distances from the source to each node in rounds 0, 1, and 2.]

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shortest Paths Up To $k$ Hops

![Graph with nodes and edges labeled with distances and predecessors highlighted]

- **Source**: A
- **Destinations**: B, C, D, E
- **Distances**:
  - A to B: 4
  - A to C: -5
  - A to D: 20
  - A to E: ∞
  - B to C: 3
  - B to D: 2
  - B to E: 8
  - C to D: -2

**Table: Shortest Paths**

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Predecessor Relationship**:
  - Predecessor of X is Y

---

73
Shortest Paths Up To $k$ Hops

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Shortest Paths Up To $k$ Hops

The diagram illustrates a weighted graph with vertices labeled A, B, C, D, and E. The weights are represented by arrows indicating the cost of moving from one vertex to another. The table below shows the shortest paths for each destination vertex up to 2 hops.

### Table: Shortest Paths Up To 2 Hops

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the graph, the predecessor of vertex X is indicated as Y.
Shortest Paths Up To $k$ Hops

Y ← X  predecessor of X is Y

<table>
<thead>
<tr>
<th>Round</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shortest Paths Up To $k$ Hops

<table>
<thead>
<tr>
<th>Round</th>
<th>Dest</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>-3</td>
<td></td>
</tr>
</tbody>
</table>
Shortest Paths Up To $k$ Hops

The graph shows a network of nodes labeled A, B, C, D, E, with directed edges between them. The table below represents the shortest paths up to $k$ hops, where:

- **Round 0**: Source node A is set as the starting point with distances to itself as 0. All other nodes are set to infinity ($\infty$).
- **Round 1**: The predecessor of node B is set as node A with a distance of 4. Node C is reached via node A with a distance of 4, making the predecessor of C node A.
- **Round 2**: Node D is reached via node C with a distance of -5, and E is reached via node B with a distance of 12. The predecessor of E is set as node B.

The table is structured as follows:

<table>
<thead>
<tr>
<th>Round</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>12</td>
</tr>
</tbody>
</table>

The predecessor of X is Y.
Shortest Paths Up To $k$ Hops

Using up to 2 hops, this is the shortest path from A to E