# CMPSC 465: LECTURE XVIII

## All Pairs Shortest Paths
## Maximum Flow

Ke Chen

October 10, 2025

# All pairs shortest path

Recall that:

- We want to fill the $dist[\cdot, \cdot]$ matrix with correct pairwise shortest distances.

# All pairs shortest path

Recall that:

- We want to fill the $dist[\cdot, \cdot]$ matrix with correct pairwise shortest distances.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.

# All pairs shortest path

Recall that:

- We want to fill the $dist[\cdot, \cdot]$ matrix with correct pairwise shortest distances.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.
- Repeating Bellman-Ford fills $dist$ row by row. $O\left(|V|^2 \cdot |E|\right)$.

# All pairs shortest path

Recall that:

- We want to fill the $dist[\cdot, \cdot]$ matrix with correct pairwise shortest distances.
- For simplicity, assume vertices are labeled by $\{1, 2, \ldots, n\}$.
- Repeating Bellman-Ford fills $dist$ row by row. $O\left(|V|^2 \cdot |E|\right)$.
- Can we use information from other rows to speed things up?

# A recursive formulation

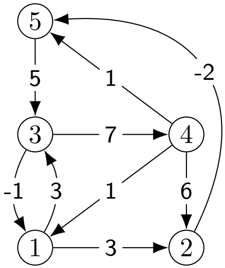
**Initially:** $$dist_0[i,j] = \begin{cases} 0 & \text{if } i = j \\ \ell(i,j) & \text{if } (i,j) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$$

This is the **adjacency matrix** of the weighted graph $G$.

# A recursive formulation

**Initially:** $dist_0[i,j] = \begin{cases} 0 & \text{if } i = j \\ \ell(i,j) & \text{if } (i,j) \in E \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$

This is the **adjacency matrix** of the weighted graph $G$.



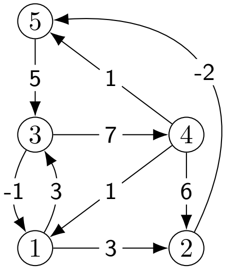

$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.



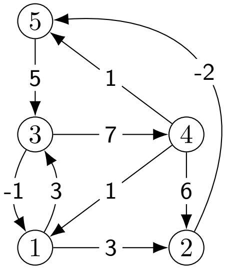

$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.
- $dist_1[i,j] = \min\left\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\right\}$.




$dist_0$ :

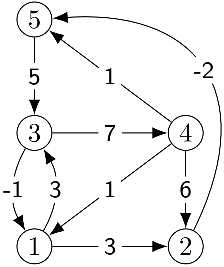
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.

- $dist_1[i,j] = \min\Big\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\Big\}.$




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

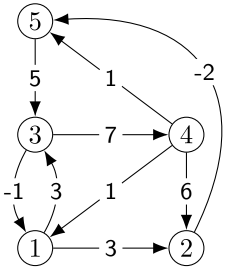
$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.
- $dist_1[i,j] = \min\left\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\right\}$.




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_1$ :

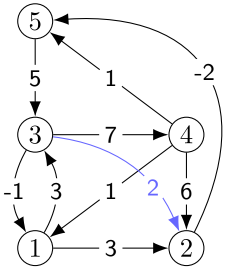
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.

- $dist_1[i,j] = \min\left\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\right\}$.




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_1$ :

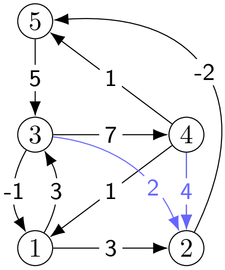
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.

- $dist_1[i,j] = \min\left\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\right\}.$




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_1$ :

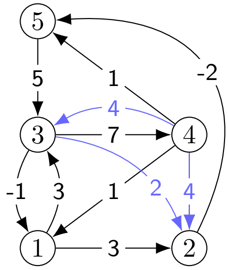
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

Consider a simpler question:

- Fill $dist_1[\cdot,\cdot]$ with pairwise shortest distances using only node 1 as an intermediate node.

- $dist_1[i,j] = \min\left\{dist_0[i,j],\ dist_0[i,1] + dist_0[1,j]\right\}.$

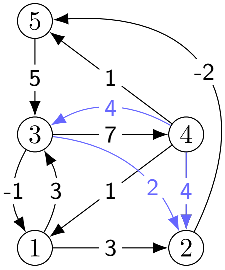



$$dist_0 : \begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$$dist_1 : \begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

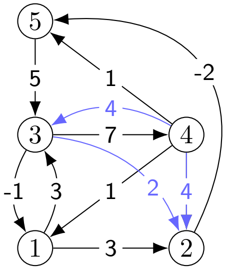
$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?

- Fill $dist_2[\cdot,\cdot]$ with pairwise shortest distances using only nodes in $\{1,2\}$ as intermediate nodes.




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_1$ :

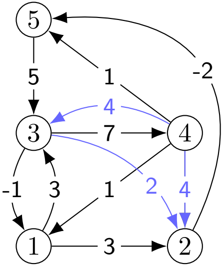
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?

- Fill $dist_2[\cdot,\cdot]$ with pairwise shortest distances using only nodes in $\{1,2\}$ as intermediate nodes.

- $dist_2[i,j] = \min\left\{dist_1[i,j],\ dist_1[i,2] + dist_1[2,j]\right\}.$




$dist_0$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & \infty & 0 & 7 & \infty \\ 1 & 6 & \infty & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

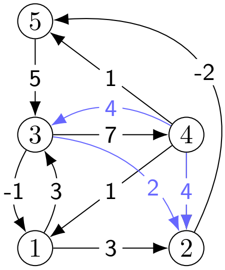
$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?

- Fill $dist_2[\cdot,\cdot]$ with pairwise shortest distances using only nodes in $\{1,2\}$ as intermediate nodes.
- $dist_2[i,j] = \min\Big\{dist_1[i,j],\ dist_1[i,2] + dist_1[2,j]\Big\}.$




$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_2$ :

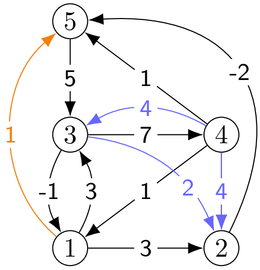
$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?

- Fill $dist_2[\cdot,\cdot]$ with pairwise shortest distances using only nodes in $\{1,2\}$ as intermediate nodes.

- $dist_2[i,j] = \min\Big\{dist_1[i,j],\ dist_1[i,2] + dist_1[2,j]\Big\}.$




$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_2$ :

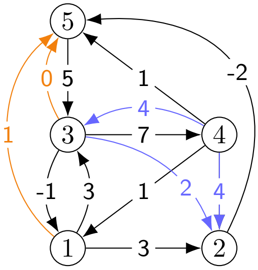
$$\begin{bmatrix} 0 & 3 & 3 & \infty & 1 \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

How about also including node 2 as an intermediate node?

- Fill $dist_2[\cdot,\cdot]$ with pairwise shortest distances using only nodes in $\{1,2\}$ as intermediate nodes.

- $dist_2[i,j] = \min\left\{dist_1[i,j],\ dist_1[i,2] + dist_1[2,j]\right\}.$




$dist_1$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & \infty \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & \infty \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

$dist_2$ :

$$\begin{bmatrix} 0 & 3 & 3 & \infty & 1 \\ \infty & 0 & \infty & \infty & -2 \\ -1 & 2 & 0 & 7 & 0 \\ 1 & 4 & 4 & 0 & 1 \\ \infty & \infty & 5 & \infty & 0 \end{bmatrix}$$

# A recursive formulation

In general:

- Start with the weighted adjacency matrix $dist_0$ (shortest distances without any intermediate nodes).

# A recursive formulation

In general:

- Start with the weighted adjacency matrix $dist_0$ (shortest distances without any intermediate nodes).

- Suppose shortest distances only using intermediate nodes in $\{1, \ldots, k-1\}$ have been correctly computed in $dist_{k-1}$.

# A recursive formulation

In general:

- Start with the weighted adjacency matrix $dist_0$ (shortest distances without any intermediate nodes).
- Suppose shortest distances only using intermediate nodes in $\{1, \ldots, k-1\}$ have been correctly computed in $dist_{k-1}$.
- Can compute shortest distances only using intermediate nodes in $\{1, \ldots, k-1, k\}$ by

$$dist_k[i,j] = \min \begin{Bmatrix} dist_{k-1}[i,j], \\ dist_{k-1}[i,k] + dist_{k-1}[k,j] \end{Bmatrix}.$$

# A recursive formulation

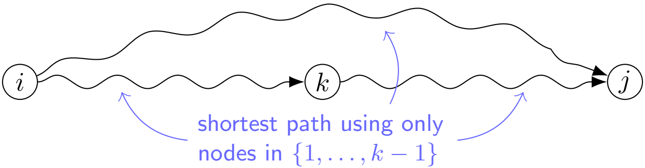
In general:

- Start with the weighted adjacency matrix $dist_0$ (shortest distances without any intermediate nodes).

- Suppose shortest distances only using intermediate nodes in $\{1, \ldots, k-1\}$ have been correctly computed in $dist_{k-1}$.

- Can compute shortest distances only using intermediate nodes in $\{1, \ldots, k-1, k\}$ by

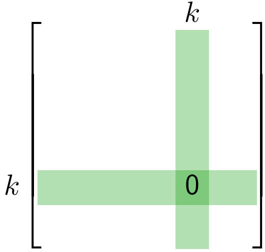
$$dist_k[i,j] = \min \left\{ \begin{array}{l} dist_{k-1}[i,j], \\ dist_{k-1}[i,k] + dist_{k-1}[k,j] \end{array} \right\}.$$



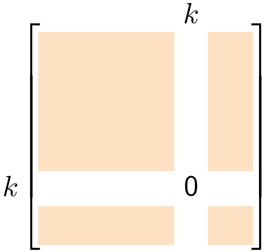
# Implementation

$$\boxed{dist_k[i,j]} = \min \left\{ \begin{matrix} dist_{k-1}[i,j], \\ \boxed{dist_{k-1}[i,k] + dist_{k-1}[k,j]} \end{matrix} \right\}.$$

$dist_{k-1}$ :



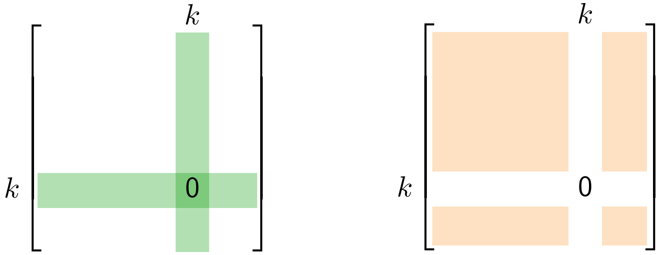

$dist_k$ :



# Implementation

$$\boxed{dist_k[i,j]} = \min \left\{ \begin{array}{l} dist_{k-1}[i,j], \\ \boxed{dist_{k-1}[i,k] + dist_{k-1}[k,j]} \end{array} \right\}.$$

$dist_{k-1}$ : $\qquad\qquad\qquad\qquad$ $dist_k$ :




The values we need from $dist_{k-1}$ will not change when computing $dist_k$, so we can simply do updates in place.

# Implementation

$$\boxed{dist_k[i,j]} = \min \left\{ \begin{array}{l} dist_{k-1}[i,j], \\ \boxed{dist_{k-1}[i,k] + dist_{k-1}[k,j]} \end{array} \right\}.$$

**Floyd-Warshall**($G = (V, E, \ell)$)

```
n = |V|
foreach (i, j) ∈ {1, ..., n}^2 do
    if i == j then dist[i, j] = 0
    else if (i, j) ∈ E then dist[i, j] = ℓ(i, j)
    else dist[i, j] = ∞

for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if dist[i, j] > dist[i, k] + dist[k, j] then
                dist[i, j] = dist[i, k] + dist[k, j]
```

# Implementation

$$\boxed{dist_k[i,j]} = \min \left\{ \begin{array}{l} dist_{k-1}[i,j], \\ \boxed{dist_{k-1}[i,k] + dist_{k-1}[k,j]} \end{array} \right\}.$$

Floyd-Warshall($G = (V, E, \ell)$)

Time complexity?

```
n = |V|
foreach (i, j) ∈ {1, ..., n}^2 do
    if i == j then dist[i, j] = 0
    else if (i, j) ∈ E then dist[i, j] = ℓ(i, j)
    else dist[i, j] = ∞

for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if dist[i, j] > dist[i, k] + dist[k, j] then
                dist[i, j] = dist[i, k] + dist[k, j]
```

# Implementation

$$dist_k[i,j] = \min \begin{Bmatrix} dist_{k-1}[i,j], \\ dist_{k-1}[i,k] + dist_{k-1}[k,j] \end{Bmatrix}.$$

**Floyd-Warshall**($G = (V, E, \ell)$)

Time complexity? $O\left(|V|^3\right)$

```
n = |V|
foreach (i, j) ∈ {1, ..., n}^2 do
    if i == j then dist[i, j] = 0
    else if (i, j) ∈ E then dist[i, j] = ℓ(i, j)
    else dist[i, j] = ∞

for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            if dist[i, j] > dist[i, k] + dist[k, j] then
                dist[i, j] = dist[i, k] + dist[k, j]
```

# Final notes on shortest paths

- What does it mean if during Floyd-Warshall, some diagonal entry $dist[i, i]$ becomes negative?

# Final notes on shortest paths

- What does it mean if during Floyd-Warshall, some diagonal entry $dist[i, i]$ becomes negative? Negative cycle detected!

# Final notes on shortest paths

- What does it mean if during Floyd-Warshall, some diagonal entry $dist[i, i]$ becomes negative? Negative cycle detected!

- We have covered four different algorithms for shortest paths:

| Scenario | Algorithm | Time complexity |
|---|---|---|
| unweighted | BFS | $O(\|V\| + \|E\|)$ |
| nonnegative weights | Dijkstra | $O((\|V\| + \|E\|) \log \|V\|)$ |
| no negative cycles | Bellman-Ford | $O(\|V\| \cdot \|E\|)$ |
| no negative cycles | Floyd-Warshall | $O(\|V\|^3)$ |

# Final notes on shortest paths

- What does it mean if during Floyd-Warshall, some diagonal entry $dist[i, i]$ becomes negative? Negative cycle detected!

- We have covered four different algorithms for shortest paths:

| Scenario | Algorithm | Time complexity |
|---|---|---|
| unweighted | BFS | $O(\|V\| + \|E\|)$ |
| nonnegative weights | Dijkstra | $O((\|V\| + \|E\|) \log \|V\|)$ |
| no negative cycles | Bellman-Ford | $O(\|V\| \cdot \|E\|)$ |
| no negative cycles | Floyd-Warshall | $O(\|V\|^3)$ |

- There are many more:

| | | |
|---|---|---|
| no negative cycles | Johnson | $O(\|V\|^2 \log \|V\| + \|V\|\|E\|)$ |
| no negative cycles | Pettie (2004) | $O(\|V\|^2 \log \log \|V\| + \|V\|\|E\|)$ |

$\vdots$

# Final notes on shortest paths

- What does it mean if during Floyd-Warshall, some diagonal entry $dist[i, i]$ becomes negative? Negative cycle detected!

- We have covered four different algorithms for shortest paths:

| | Scenario | Algorithm | Time complexity |
|---|---|---|---|
| SS | unweighted | BFS | $O(\|V\| + \|E\|)$ |
| SS | nonnegative weights | Dijkstra | $O((\|V\| + \|E\|) \log \|V\|)$ |
| SS | no negative cycles | Bellman-Ford | $O(\|V\| \cdot \|E\|)$ |
| AP | no negative cycles | Floyd-Warshall | $O(\|V\|^3)$ |

- There are many more:

| | Scenario | Algorithm | Time complexity |
|---|---|---|---|
| AP | no negative cycles | Johnson | $O(\|V\|^2 \log \|V\| + \|V\|\|E\|)$ |
| AP | no negative cycles | Pettie (2004) | $O(\|V\|^2 \log \log \|V\| + \|V\|\|E\|)$ |

$\vdots$

# Flow network

One particularly useful application of graphs is to model transportation networks, where edges allow some sort of traffic and nodes act as switches or hubs.

# Flow network

One particularly useful application of graphs is to model transportation networks, where edges allow some sort of traffic and nodes act as switches or hubs.

Examples

- Highways and interchanges
- Fluid networks (pipes and junctions)
- Links and routers

# Flow network

One particularly useful application of graphs is to model transportation networks, where edges allow some sort of traffic and nodes act as switches or hubs.

Examples

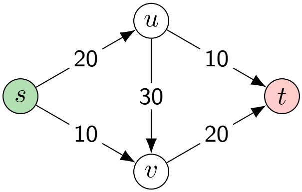
- Highways and interchanges
- Fluid networks (pipes and junctions)
- Links and routers

Definition A Flow Network is a directed graph $G = (V, E)$ s.t.:

1. Each edge $e \in E$ has a nonnegative capacity $c(e)$.
2. There is a unique source node $s \in V$.
3. There is a unique sink node $t \in V$.

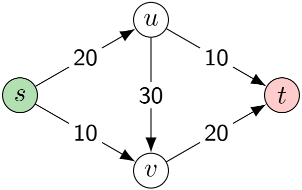
# Flow

Example A flow network $G$:



# Flow

Example A flow network $G$:



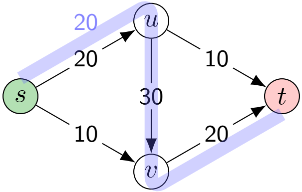

Definition An $s-t$ flow in a flow network is a function $f$ that maps each edge to a nonnegative real number ($f : E \to \mathbb{R}_{\geq 0}$) satisfying:

1. [Capacity Condition] $0 \leq f(e) \leq c(e) \quad \forall e \in E$
2. [Flow Conservation] For all $v \in V$, $v \notin \{s, t\}$:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e).$$

# Flow

Example A flow network $G$:



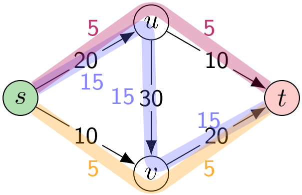

Definition An $s - t$ flow in a flow network is a function $f$ that maps each edge to a nonnegative real number ($f : E \to \mathbb{R}_{\geq 0}$) satisfying:

1. [Capacity Condition] $0 \leq f(e) \leq c(e) \quad \forall e \in E$
2. [Flow Conservation] For all $v \in V$, $v \notin \{s, t\}$:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e).$$

# Flow

Example A flow network $G$:




Definition An $s - t$ flow in a flow network is a function $f$ that maps each edge to a nonnegative real number ($f : E \to \mathbb{R}_{\geq 0}$) satisfying:

1. [Capacity Condition] $0 \leq f(e) \leq c(e) \quad \forall e \in E$
2. [Flow Conservation] For all $v \in V$, $v \notin \{s, t\}$:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e).$$

# The maximum flow problem

The value of a flow $f$ is $v(f) = \sum_{e \text{ out of } s} f(e)$.

# The maximum flow problem

The value of a flow $f$ is $v(f) = \sum_{e \text{ out of } s} f(e)$.

Note that $v(f) = \sum_{e \text{ into } t} f(e)$ (why?).

# The maximum flow problem

The value of a flow $f$ is $v(f) = \sum_{e \text{ out of } s} f(e)$.

Note that $v(f) = \sum_{e \text{ into } t} f(e)$ (why?).

Max Flow Problem Given a flow network, find the flow of maximum value.