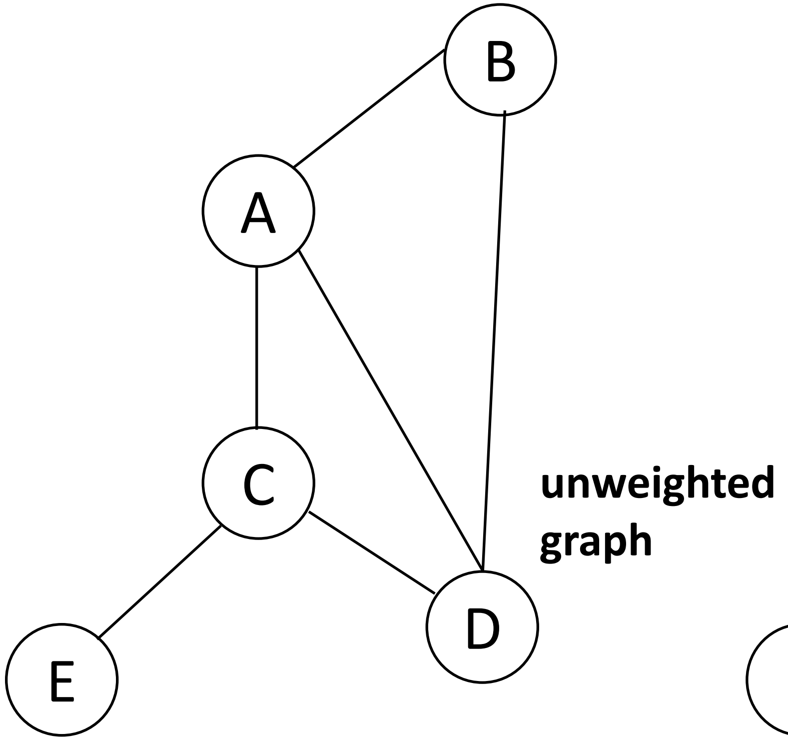
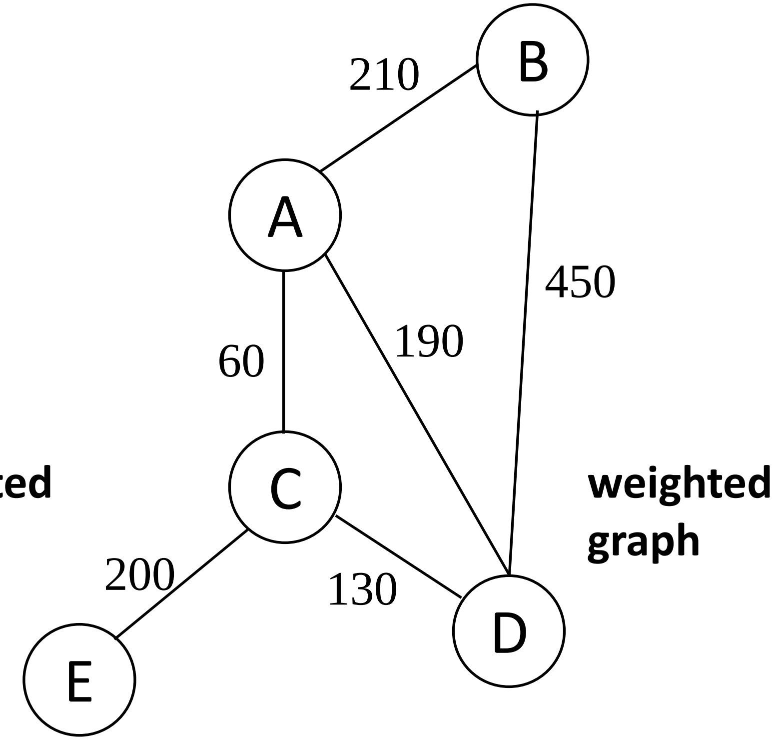
All Pairs Shortest Path

Floyd-Warshall Algorithm

Shortest Path Problems

- **What is shortest path ?**
  - shortest length between two vertices for an unweighted graph:
  - smallest cost between two vertices *for* a weighted graph:





Shortest Path Problems

- How can we find the shortest route between two points on a map?
- Model the problem as a graph problem:
  - Road map is a weighted graph:

    vertices = cities

    edges = road segments between cities

    edge weights = road distances
  - Goal: find a shortest path between two vertices (cities)

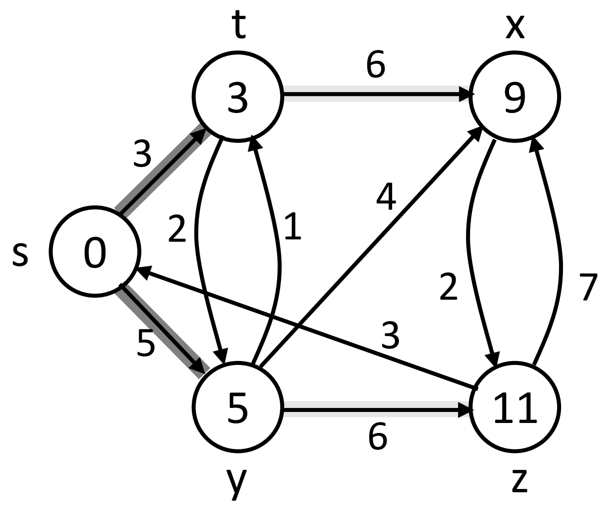
Shortest Path Problems

- **Input:**
  - Directed graph G = (V, E)
  - Weight function w : E → **R**
- **Weight of path** p = $\langle v_0, v_1, \ldots, v_k \rangle$

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- **Shortest-path weight** from u to v:

$$\delta(u, v) = \begin{cases} \min \{ w(p) : u \overset{p}{\rightsquigarrow} v \} & \text{if there exists a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$



- Shortest path u to v is any path p such that $w(p) = \delta(u, v)$

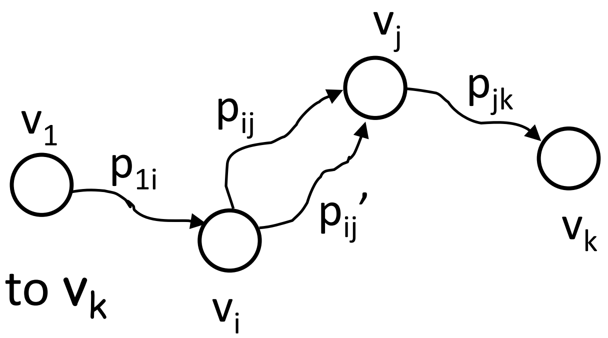
Variants of Shortest Paths

- **Single-source shortest path**
  - G = (V, E) $\Rightarrow$ find a shortest path from a given source vertex **s** to each vertex v $\in$ V
- **Single-destination shortest path**
  - Find a shortest path to a given destination vertex **t** from each vertex v
  - Reverse the direction of each edge $\Rightarrow$ single-source
- **Single-pair shortest path**
  - Find a shortest path from u to v for given vertices u and v
  - Solve the single-source problem
- **All-pairs shortest-paths**
  - Find a shortest path from u to v for every pair of vertices u and v

Optimal Substructure of Shortest Paths

Given:

- A weighted, directed graph $G = (V, E)$
- A weight function $w: E \rightarrow \mathbf{R}$,
- A shortest path $p_{1k} = \langle v_1, v_2, \ldots, v_k \rangle$ from v_1 to v_k
- A subpath of p: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \leq i \leq j \leq k$



Then: p_{ij} is a shortest path from v_i to v_j

Proof: $p = v_1 \overset{p_{1i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$

$$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$$

Assume $\exists\, p_{ij}'$ from v_i to v_j with $w(p_{ij}') < w(p_{ij})$

$\Rightarrow w(p') = w(p_{1i}) + w(p_{ij}') + w(p_{jk}) < w(p)$ contradiction!

What can we use?

Use Dijkstra's $|V|$ times!!!

- **If all the weights are non-negative.**
- Dijkstra has $O\,(E \log V)$ complexity. For all pairs, it becomes $O\,(VE \log V)$
- Which is equal $O\,(V^{3} \log V)$ in the case of $E = O(V^{2})$.

Use Bellman-Ford $|V|$ times!!!

- **If negative weights are allowed.**
- Then, we have $O\,(V^{2}E)$.
- Which is equal $O\,(V^{4})$ in the case of $E = O(V^{2})$.

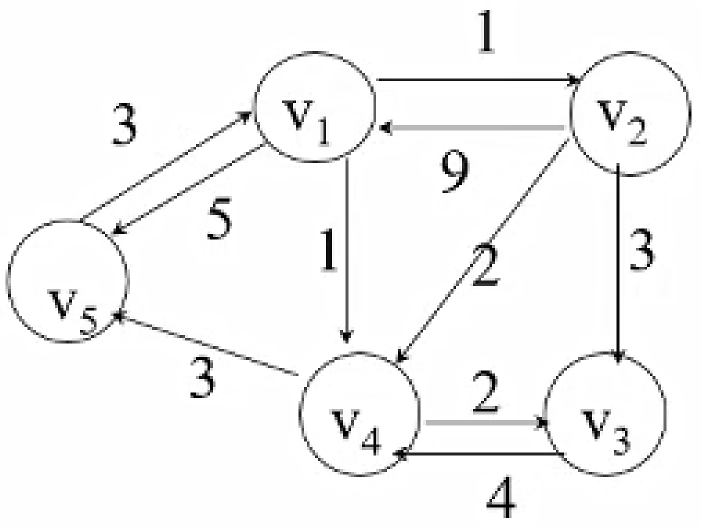
All Pairs Shortest Path

- *The problem*: find the shortest path between every pair of vertices of a graph
  - Expensive using a brute-force approach
- *The graph* may contain negative edges but no negative cycles
- *Representation*: a weight matrix where
  W(i,j)=0 if i=j.
  W(i,j)=∞ if there is no edge between i and j.
  W(i,j)="weight of edge"
- Note: we have shown principle of optimality applies to shortest path problems

Weight Matrix

	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

Adjacency Matrix



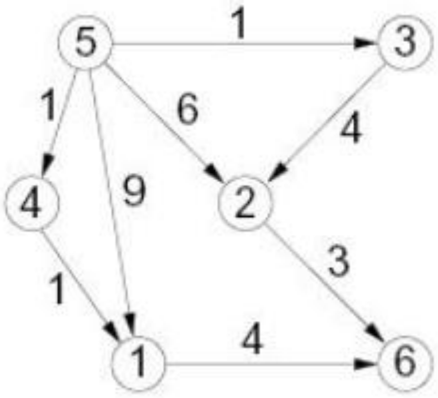
The Shortest Path Structure

Intermediate Vertex

For a path $p = \langle v_1, v_2, \ldots, v_l \rangle$, an **intermediate vertex** is any vertex of p other than v_1 or v_l.

Define

$d_{ij}^{(k)}$ =weight of a shortest path between i and j with all intermediate vertices are in the set $\{1, 2, \ldots, k\}$.



$d_{5,6}^{(0)}$ = INF (no path)

$d_{5,6}^{(1)}$ = 13 (5,1,6)

$d_{5,6}^{(2)}$ = 9 (5,2,6)

$d_{5,6}^{(3)}$ = 8 (5,3,2,6)

$d_{5,6}^{(4)}$ = 6 (5,4,1,6)

The Sub-problems

- How can we define the shortest distance $d_{i,j}$ in terms of "smaller" problems?
- One way is to restrict the paths to only include vertices from a restricted subset.
- Initially, the subset is empty.
- Then, it is incrementally increased until it includes all the vertices.

The Sub-problems

- Let $D^{(k)}[i,j]$=weight of a shortest path from v_i to v_j using only vertices from $\{v_1,v_2,\ldots,v_k\}$ as intermediate vertices in the path
  - $D^{(0)}=W$
  - $D^{(n)}=D$ which is the goal matrix
- How do we compute $D^{(k)}$ from $D^{(k-1)}$?

The Sub-problems

- $d_{ij}^{(k)}$ is the length of the shortest path from i to j such that all intermediate vertices on the path (if any) are in the set $\{1, 2, \ldots, k\}$.
- Let $D^{(k)}$ be the $n \times n$ matrix $[d_{ij}^{(k)}]$.
- Subproblems: compute $D^{(k)}$ for $k = 0, 1, \cdots, n$.
- Original Problem: $D = D^{(n)}$, i.e. $d_{ij}^{(n)}$ is the shortest distance from i to j

The Recursive Idea

Simply look at the following cases cases

- **Case I** k is not an intermediate vertex, then a shortest path from i to j with all intermediate vertices $\{1, ..., k-1\}$ is a shortest path from i to j with intermediate vertices $\{1, ..., k\}$.

$$\Longrightarrow d_{ij}^{(k)} = d_{ij}^{(k-1)}$$

- **Case II** if k is an intermediate vertice. Then, $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$ and we can make the following statements:
  - p_1 is a shortest path from i to k with all intermediate vertices in the set $\{1, ..., k-1\}$.
  - p_2 is a shortest path from k to j with all intermediate vertices in the set $\{1, ..., k-1\}$.

$$\Longrightarrow d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$

The Graphical Idea

Consider

All possible intermediate vertices in $\{1, 2, ..., k\}$

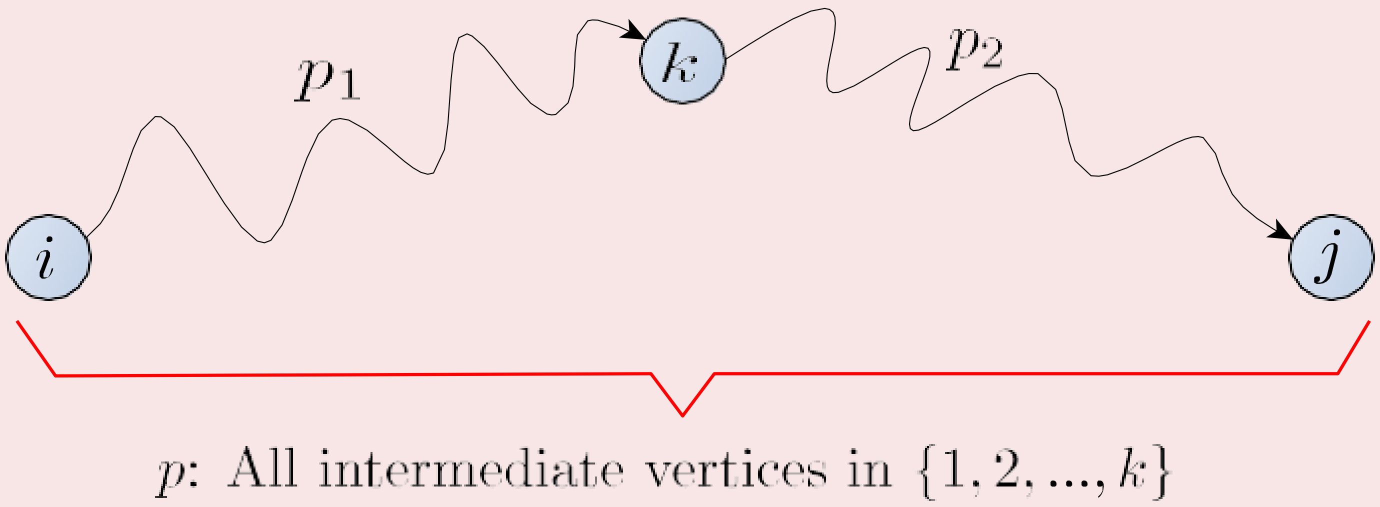


Figure: The Recursive Idea

The Recursive Solution

The Recursion

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 \end{cases}$$

Final answer when $k = n$

We recursively calculate $D^{(n)} = \left(d_{ij}^{(n)}\right)$ or $d_{ij}^{(n)} = \delta\left(i, j\right)$ for all $i, j \in V$.

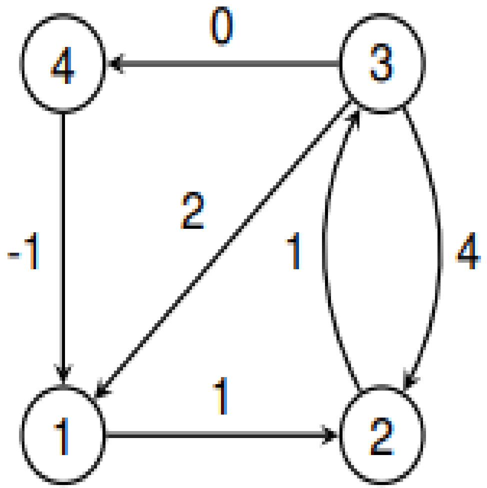
Floyd-Warshall Algorithm

```
Floyd//Computes shortest distance between all pairs of
   //nodes, and saves P to enable finding shortest paths
   D^0 ← W     // initialize D array to W [ ]
   P ← 0       // initialize P array to [0]
   for k ← 1 to n
       for i ← 1 to n
           for j ← 1 to n
               if (D^(k-1)[ i, j ] > D^(k-1) [ i, k ] + D^(k-1) [ k, j ] )
                   then  D^k[ i, j ] ← D^(k-1) [ i, k ] + D^(k-1) [ k, j ]
                         P[ i, j ] ← k;
               else D^k [ i, j ] ← D^(k-1) [ i, j ]
```

$O(V^3)$

Example

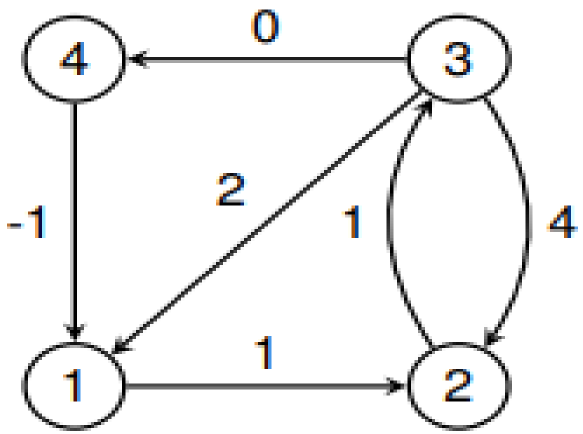
Consider the following graph and its corresponding adjacency matrix:



$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 4 & 0 & 0 \\ -1 & \infty & \infty & 0 \end{pmatrix}$$

Example

Consider the following graph and its corresponding adjacency matrix:

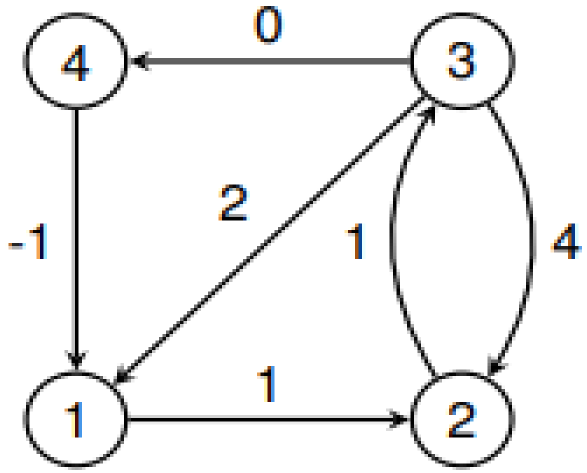


$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 4 & 0 & 0 \\ -1 & \infty & \infty & 0 \end{pmatrix}$$

$$d^{(1)} = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 3 & 0 & 0 \\ -1 & 0 & \infty & 0 \end{pmatrix}$$

Example

Consider the following graph and its corresponding adjacency matrix:

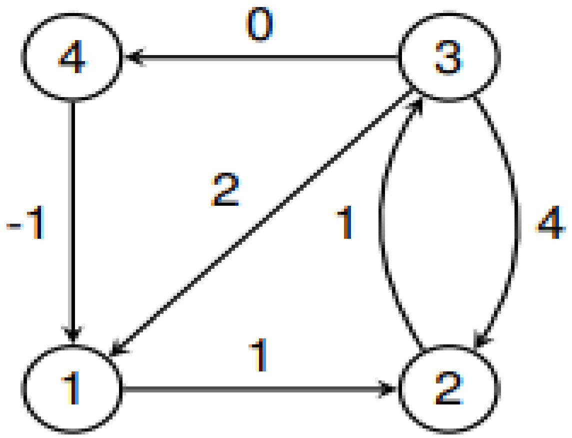


$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 4 & 0 & 0 \\ -1 & \infty & \infty & 0 \end{pmatrix}$$

$$d^{(1)} = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 3 & 0 & 0 \\ -1 & 0 & \infty & 0 \end{pmatrix}, \quad d^{(2)} = \begin{pmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 3 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix}$$

Example

Consider the following graph and its corresponding adjacency matrix:

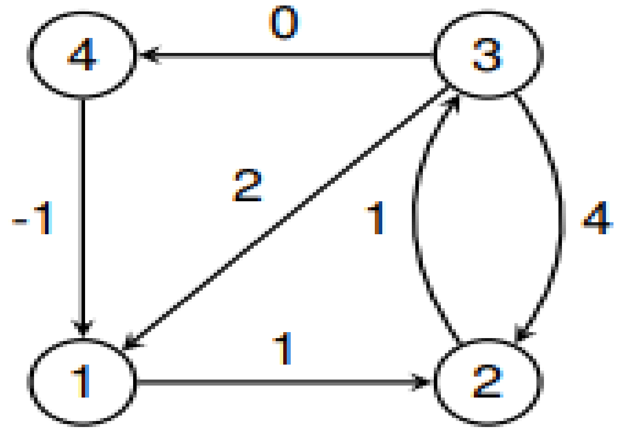


$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 4 & 0 & 0 \\ -1 & \infty & \infty & 0 \end{pmatrix}$$

$$d^{(3)} = \begin{pmatrix} 0 & 1 & 2 & \color{blue}{2} \\ \color{blue}{3} & 0 & 1 & \color{blue}{1} \\ 2 & 3 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix}$$

Example

Consider the following graph and its corresponding adjacency matrix:



$$\begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 2 & 4 & 0 & 0 \\ -1 & \infty & \infty & 0 \end{pmatrix}$$

$$d^{(3)} = \begin{pmatrix} 0 & 1 & 2 & 2 \\ 3 & 0 & 1 & 1 \\ 2 & 3 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix}, \quad d^{(4)} = \begin{pmatrix} 0 & 1 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{pmatrix}.$$