Floyd-Warshall algorithm

![Graph diagram](image)

\[
\begin{bmatrix}
  s & t & x & y & z \\ 
  s & 0 & 10 & \infty & 5 & \infty \\ 
  t & \infty & 0 & 1 & 2 & \infty \\ 
  x & \infty & \infty & 0 & \infty & 4 \\ 
  y & \infty & 3 & 9 & 0 & 2 \\ 
  z & 2 & \infty & 6 & \infty & 0 
\end{bmatrix}
\]
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex

Xiaojuan Cai, School of Software, SJTU.
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that passes none vertex
Floyd-Warshall algorithm

The shortest path from u to v that may passes s
Floyd-Warshall algorithm

The shortest path from u to v that may passes s
Floyd-Warshall algorithm

The shortest path from u to v that may passes s
The shortest path from u to v that may passes s.
Floyd-Warshall algorithm

The shortest path from u to v that may passes s
The shortest path from u to v that may passes s, t
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t
**Floyd-Warshall algorithm**

The shortest path from u to v that may pass through s, t
Floyd-Warshall algorithm

The shortest path from u to v that may pass s, t
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t

The situation just before the first iteration of the while loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5.

(b)–(f) The situation after each successive iteration of the while loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d values and predecessors shown in part (f) are the final values.

Because Dijkstra’s algorithm always chooses the “lightest” or “closest” vertex in V\S to add to set S, we always use a greedy strategy. Chapter 16 explains greedy strategies in detail, but you need not have read that chapter to understand Dijkstra’s algorithm. Greedy strategies do not always yield optimal results in general, but as the following theorem and its corollary show, Dijkstra’s algorithm does indeed compute shortest paths. The key is to show that each time it adds a vertex u to set S, we have u: d_D(u) \leq Dı.(s; u).

Theorem 24.6 (Correctness of Dijkstra’s algorithm)
Dijkstra’s algorithm, run on a weighted, directed graph G=(V; E) with non-negative weight function w and source s, terminates with u: d_D(u) for all vertices u \in V.
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t, x
The shortest path from u to v that may passes s, t, x
The shortest path from \( u \) to \( v \) that may pass \( s, t, x \)
The shortest path from u to v that may passes s, t, x
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t, x
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t, x
Floyd-Warshall algorithm

The shortest path from $u$ to $v$ that may passes $s, t, x$
### Floyd-Warshall algorithm

The shortest path from $u$ to $v$ that may pass $s, t, x$

![Graph Diagram]

<table>
<thead>
<tr>
<th></th>
<th>$s$</th>
<th>$t$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>0</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>$t$</td>
<td>$\infty$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$x$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
<td>$\infty$</td>
<td>4</td>
</tr>
<tr>
<td>$y$</td>
<td>$\infty$</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$z$</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t, x, y
The shortest path from u to v that may passes s, t, x, y
Floyd-Warshall algorithm

The shortest path from u to v that may pass s, t, x, y

Xiaojuan Cai, School of Software, SJTU.
Floyd-Warshall algorithm

The shortest path from $u$ to $v$ that may passes $s, t, x, y$
Floyd-Warshall algorithm

The shortest path from $u$ to $v$ that may passes $s, t, x, y$
Floyd-Warshall algorithm

The shortest path from u to v that may pass through s, t, x, y,
Floyd-Warshall algorithm

The shortest path from u to v that may passes s, t, x, y
Floyd-Warshall Algorithm

The shortest path from \( u \) to \( v \) that may pass \( s, t, x, y \)
The shortest path from u to v that may passes s, t, x, y, z