Plain `algo` and code

title: "Floyd-Warshall" (algo only)

parameters: ("V", "E", "w") (algo only)

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3      $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5      $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                  $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

```python
def floyd_warshall(G):
    # let G be an adjacency matrix
    dist = G

    for k in range(len(G)):
        for i in range(len(G)):
            for j in range(len(G)):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist
```

Basic styling parameters

fill:	none
stroke:	2pt + black
radius:	10pt
row-gutter:	8pt
column-gutter:	8pt
inset:	15pt
indent-size:	12pt (algo only)
indent-guides:	1pt + gray
indent-guides-offset:	4pt
comment-prefix:	[#sym.triangle] (algo only)

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

```
 1  Let dist[u, v] ← ∞ for u, v in V
 2  For (u, v) in E:
 3  │  dist[u, v] ← w(u, v)                        ▷ edge weights
 4  For v in V:
 5  │  dist[v, v] ← 0                              ▷ base case
 6
 7  For k ← 1 to |V|:
 8  │  For i ← 1 to |V|:
 9  │  │  For j ← 1 to |V|:
10  │  │  │  ▷ if new path is shorter, reduce distance
11  │  │  │  If dist[i, j] > dist[i, k] + dist[k, j]:
12  │  │  │  │  dist[i, j] ← dist[i, k] + dist[k, j]
13
14  Return dist
```

```python
 1  def floyd_warshall(G):
 2      # let G be an adjacency matrix
 3      dist = G
 4
 5      for k in range(len(G)):
 6        for i in range(len(G)):
 7          for j in range(len(G)):
 8            if dist[i][j] > dist[i][k] + dist[k][j]:
 9              dist[i][j] = dist[i][k] + dist[k][j]
10
11      return dist
```

Empty bodies

code with empty raw block

1

Indent guides with line wrapping

indent-guides: 1pt + black

$\textsc{Floyd-Warshall}(V, E, w)$:

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3      $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5      $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12              $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13   blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah

14

15 **Return** dist

```python
def floyd_warshall(G):

    # let G be an adjacency matrix

    dist = G


    for k in range(len(G)):

        for i in range(len(G)):

            for j in range(len(G)):

                if dist[i][j] > dist[i][k] + dist[k][j]:

                    dist[i][j] = dist[i][k] + dist[k][j]

                    blah blah blah blah blah blah blah blah blah blah blah blah blah
blah blah blah

```

code indent guides with custom tab size

indent-guides: 1pt + black

tab-size: 2

```
 1   def floyd_warshall(
 2   │   │   G
 3   │   ):
 4   │   # let G be an adjacency matrix
 5   │   dist = G
 6   │
 7   │   for k in range(len(G)):
 8   │   │   for i in range(len(G)):
 9   │   │   │   for j in range(len(G)):
10   │   │   │   │   if dist[i][j] > dist[i][k] + dist[k][j]:
11   │   │   │   │   │   dist[i][j] = dist[i][k] + dist[k][j]
12   │
13   │   return dist
```

No line numbers

line-numbers: false

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

Let $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

For (u, v) **in** E:

    $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

For v **in** V:

    $\text{dist}[v, v] \leftarrow 0$ // base case

For $k \leftarrow 1$ **to** $|V|$:

    For $i \leftarrow 1$ **to** $|V|$:

        For $j \leftarrow 1$ **to** $|V|$:

            // if new path is shorter, reduce distance

            If $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

                $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

Return dist

```python
def floyd_warshall(G):
    # let G be an adjacency matrix
    dist = G

    for k in range(len(G)):
        for i in range(len(G)):
            for j in range(len(G)):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist
```

algo without keywords

strong-keywords: false

<u>Floyd-Warshall</u>(V, E, w):

1   Let $\text{dist}[u, v] \leftarrow \infty$ for u, v in V

2   For (u, v) in E:

3       $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4   For v in V:

5       $\text{dist}[v, v] \leftarrow 0$ // base case

6

7   For $k \leftarrow 1$ to $|V|$:

8   For $i \leftarrow 1$ to $|V|$:

9   For $j \leftarrow 1$ to $|V|$:

10   // if new path is shorter, reduce distance

11   If $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                  $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 Return dist

`algo` with custom keywords

keywords: ("in", "to")

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

1   Let $\text{dist}[u, v] \leftarrow \infty$ for u, v **in** V

2   For (u, v) **in** E:

3       $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4   For v **in** V:

5       $\text{dist}[v, v] \leftarrow 0$ // base case

6

7   For $k \leftarrow 1$ **to** $|V|$:

8   For $i \leftarrow 1$ **to** $|V|$:

9   For $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   If $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                  $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 Return dist

`algo` without title

title: none

```
(V, E, w):
 1   Let dist[u, v] ← ∞ for u, v in V
 2   For (u, v) in E:
 3       dist[u, v] ← w(u, v)                              // edge weights
 4   For v in V:
 5       dist[v, v] ← 0                                    // base case
 6
 7   For k ← 1 to |V|:
 8       For i ← 1 to |V|:
 9           For j ← 1 to |V|:
10               // if new path is shorter, reduce distance
11               If dist[i, j] > dist[i, k] + dist[k, j]:
12                   dist[i, j] ← dist[i, k] + dist[k, j]
13
14   Return dist
```

algo without parameters

parameters: ()

<u>Floyd-Warshall</u>():

1 **Let** $\mathrm{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3      $\mathrm{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5      $\mathrm{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\mathrm{dist}[i, j] > \mathrm{dist}[i, k] + \mathrm{dist}[k, j]$:

12                 $\mathrm{dist}[i, j] \leftarrow \mathrm{dist}[i, k] + \mathrm{dist}[k, j]$

13

14 **Return** dist

`algo` without header

title: none

parameters: ()

1	**Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V
2	**For** (u, v) **in** E**:**
3	$\quad \text{dist}[u, v] \leftarrow w(u, v)$ — // edge weights
4	**For** v **in** V**:**
5	$\quad \text{dist}[v, v] \leftarrow 0$ — // base case
6	
7	**For** $k \leftarrow 1$ **to** $|V|$**:**
8	$\quad$ **For** $i \leftarrow 1$ **to** $|V|$**:**
9	$\quad\quad$ **For** $j \leftarrow 1$ **to** $|V|$**:**
10	$\quad\quad\quad$ // if new path is shorter, reduce distance
11	$\quad\quad\quad$ **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$**:**
12	$\quad\quad\quad\quad \text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$
13	
14	**Return** dist

algo with content-type parameters

parameters: ([#text(blue, [V])], [#text(red, [E])], [#text(green, [w])])

FLOYD-WARSHALL(V, E, w):

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3     $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5     $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

`algo` with content-type title

title: [#set text(red);Floyd-Warshall]

Floyd-Warshall():

1   **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2   **For** (u, v) **in** E:

3       $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4   **For** v **in** V:

5       $\text{dist}[v, v] \leftarrow 0$ // base case

6

7   **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12              $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

`algo` with custom header

Floyd-Warshall Algorithm

     Inputs: graph $G = (V, E)$
                      weight function $w : E \rightarrow \mathbb{R}$
    Outputs: distance matrix dist

---

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E**:**

3      $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V**:**

5      $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$**:**

8   **For** $i \leftarrow 1$ **to** $|V|$**:**

9   **For** $j \leftarrow 1$ **to** $|V|$**:**

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$**:**

12               $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

Text styling

main-text-styles: (fill: green)

line-number-styles: (fill: red)

comment-styles: (fill: blue) (algo only)

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3      $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5      $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                  $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

```python
1  def floyd_warshall(G):
2    # let G be an adjacency matrix
3    dist = G
4
5    for k in range(len(G)):
6      for i in range(len(G)):
7        for j in range(len(G)):
8          if dist[i][j] > dist[i][k] + dist[k][j]:
9            dist[i][j] = dist[i][k] + dist[k][j]
10
11    return dist
```

Indent guides with big main text

indent-guides: 1pt + black

main-text-styles: (size: 15pt)

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

```
1   Let dist[u, v] ← ∞ for u, v in V
2   For (u, v) in E:
3   │   dist[u, v] ← w(u, v)                          // edge weights
4   For v in V:
5   │   dist[v, v] ← 0                                // base case
6
7   For k ← 1 to |V|:
8   │   For i ← 1 to |V|:
9   │   │   For j ← 1 to |V|:
10  │   │   │       // if new path is shorter, reduce distance
11  │   │   │   If dist[i, j] > dist[i, k] + dist[k, j]:
12  │   │   │   │   dist[i, j] ← dist[i, k] + dist[k, j]
13
14  Return dist
```

```python
1   def floyd_warshall(G):
2   │     # let G be an adjacency matrix
3   │     dist = G
4
5   │     for k in range(len(G)):
6   │     │   for i in range(len(G)):
7   │     │   │   for j in range(len(G)):
8   │     │   │   │   if dist[i][j] > dist[i][k] + dist[k][j]:
9   │     │   │   │   │   dist[i][j] = dist[i][k] + dist[k][j]
10
11  │     return dist
```

Indent guides with big line numbers

indent-guides: 1pt + black

line-number-styles: (size: 15pt)

FLOYD-WARSHALL(V, E, w):

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3 $\quad$ $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5 $\quad$ $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8 $\quad$ **For** $i \leftarrow 1$ **to** $|V|$:

9 $\quad\quad$ **For** $j \leftarrow 1$ **to** $|V|$:

10 $\quad\quad\quad$ // if new path is shorter, reduce distance

11 $\quad\quad\quad$ **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12 $\quad\quad\quad\quad$ $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

```python
def floyd_warshall(G):
    # let G be an adjacency matrix
    dist = G

    for k in range(len(G)):
        for i in range(len(G)):
            for j in range(len(G)):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist
```

`algo` indent guides with big comments

indent-guides: 1pt + black

comment-styles: (size: 15pt)

FLOYD-WARSHALL(V, E, w):

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3 | $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5 | $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8 | **For** $i \leftarrow 1$ **to** $|V|$:

9 | | **For** $j \leftarrow 1$ **to** $|V|$:

10 | | | // if new path is shorter, reduce distance

11 | | | **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12 | | | | $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14 **Return** dist

Alignment

indent-guides: 1pt + black

block-align: bottom + right

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

1   **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2   **For** (u, v) **in** E:

3      $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4   **For** v **in** V:

5      $\text{dist}[v, v] \leftarrow 0$ // base case

6

7   **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10   // if new path is shorter, reduce distance

11   **If** $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$:

12                $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$

13

14   **Return** dist

```python
def floyd_warshall(G):
    # let G be an adjacency matrix
    dist = G

    for k in range(len(G)):
        for i in range(len(G)):
            for j in range(len(G)):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist
```

Breakable

indent-guides: 1pt + black

breakable: true

$\underline{\text{Floyd-Warshall}}(V, E, w)$:

1 **Let** $\text{dist}[u, v] \leftarrow \infty$ **for** u, v **in** V

2 **For** (u, v) **in** E:

3     $\text{dist}[u, v] \leftarrow w(u, v)$ // edge weights

4 **For** v **in** V:

5     $\text{dist}[v, v] \leftarrow 0$ // base case

6

7 **For** $k \leftarrow 1$ **to** $|V|$:

8   **For** $i \leftarrow 1$ **to** $|V|$:

9   **For** $j \leftarrow 1$ **to** $|V|$:

10				// if new path is shorter, reduce distance
11				**If** $\text{dist}[i,j] > \text{dist}[i,k] + \text{dist}[k,j]$:
12				$\quad \text{dist}[i,j] \leftarrow \text{dist}[i,k] + \text{dist}[k,j]$
13				
14	**Return** dist			

```python
def floyd_warshall(G):
    # let G be an adjacency matrix
    dist = G

    for k in range(len(G)):
        for i in range(len(G)):
            for j in range(len(G)):
```

```
8               if dist[i][j] > dist[i][k] + dist[k][j]:
9                   dist[i][j] = dist[i][k] + dist[k][j]
10
11    return dist
```