

**COMP20003**  
**Algorithms and Data Structures**  
**All Pairs Shortest Paths**

Nir Lipovetzky  
Department of Computing and  
Information Systems  
University of Melbourne  
Semester 2



**Shortest Paths:**  
**Single Source vs. All Pairs**



- **Single source:**
  - Shortest paths **from one** vertex **to all** others
  - Dijkstra's algorithm:  $O((V+E)\log V)$
- **All pairs:**
  - Shortest paths **from every** vertex **to every** other vertex

Why not run Dijkstra's algorithm once for every vertex?

COMP 20003 Algorithms and Data Structures

1-2

**Shortest Paths:**  
**Single Source vs. All Pairs**



Using Dijkstra's multiple times:

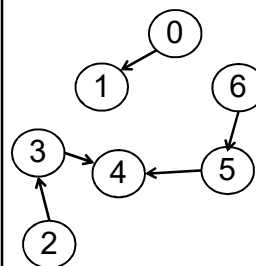
- **Dijkstra's** algorithm:  $O((V+E) \log V)$
- **Once for every vertex:**  $O(\quad)$
- $O(\quad)$  for dense graphs.

Can we do better?

COMP 20003 Algorithms and Data Structures

1-3

**Transitive closure:**  
**Unconnected directed graph**



COMP 20003 Algorithms and Data Structures

1-4

### Transitive closure: Unconnected directed graph

```

WARSHALL ALGORITHM
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

COMP 20003 Algorithms and Data Structures 1-5

### Transitive closure: Unconnected directed graph

```

WARSHALL ALGORITHM
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0

1-6

### Transitive closure: Unconnected directed graph

```

WARSHALL ALGORITHM
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0

1-7

### Transitive closure: Unconnected directed graph

```

WARSHALL ALGORITHM
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	0	1	0

i=0 Column 0 all 0, so no A[s][i]  
 Row 1 all 0 so no A[i][t]  
 i=1  
 i=2 Column 2 all 0, so no A[s][i]  
 i=3 A[2][3]&& A[3][4], so A[2][4]  
 i=4 Row 4 all 0, so no A[i][t]  
 i=5 A[6][5]&& A[5][4], so A[6][4]  
 i=6 Column 6 all 0, so no A[s][i]

**Transitive closure:  
Unconnected directed graph**

**WARSHALL ALGORITHM**

```

for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	1	1	0

*i*=0 Column 0 all 0, so no  $A[s][i]$   
*i*=1 Row 1 all 0 so no  $A[i][t]$   
*i*=2 Column 2 all 0, so no  $A[s][i]$   
*i*=3  $A[2][3]$  &&  $A[3][4]$ , so  $A[2][4]$   
*i*=4 Row 4 all 0, so no  $A[i][t]$   
*i*=5  $A[6][5]$  &&  $A[5][4]$ , so  $A[6][4]$   
*i*=6 Column 6 all 0, so no  $A[s][i]$ <sup>1-9</sup>

**Transitive closure:  
Unconnected directed graph**

**WARSHALL ALGORITHM**

```

for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] && A[i][t])
        A[s][t] = TRUE;
  
```

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	1	1	0

*i*=0 Column 0 all 0, so no  $A[s][i]$   
*i*=1 Row 1 all 0 so no  $A[i][t]$   
*i*=2 Column 2 all 0, so no  $A[s][i]$   
*i*=3  $A[2][3]$  &&  $A[3][4]$ , so  $A[2][4]$   
*i*=4 Row 4 all 0, so no  $A[i][t]$   
*i*=5  $A[6][5]$  &&  $A[5][4]$ , so  $A[6][4]$   
*i*=6 Column 6 all 0, so no  $A[s][i]$ <sup>1-10</sup>

**Transitive closure:  
Unconnected directed graph**

	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	0	0	1	1	0

1-11

**Transitive Closure with multi-segment paths**

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0

COMP 20003 Algorithms and Data Structures 1-12

## Warshall algorithm: Analysis

```

WARSHALL ALGORITHM
for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if( A[s][i] && A[i][t])
                A[s][t] = TRUE;
    
```

$\Theta(?)$  for graph of  $V$  vertices and  $E$  edges

How does this compare with running  
**Dijkstra's** algorithm  $V$  times?

COMP 20003 Algorithms and Data Structures

1-13

## Floyd-Warshall algorithm

- Warshall, Stephen (January 1962). "A theorem on Boolean matrices". *Journal of the ACM* **9** (1): 11–12.
- Floyd, Robert W. (June 1962). "Algorithm 97: Shortest Path". *Communications of the ACM* **5** (6): 345.

COMP 20003 Algorithms and Data Structures

1-14

## Use Warshall framework to get shortest path lengths

**Warshall algorithm**, boolean matrix, no self-loops:

```

for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if( A[s][i] && A[i][t])
                A[s][t] = TRUE;
    
```

COMP 20003 Algorithms and Data Structures

1-15

## Use Warshall framework to get shortest path lengths

**Warshall algorithm** (boolean matrix, no self-loops):

```

for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if( A[s][i] && A[i][t]) A[s][t] = TRUE;
Floyd-Warshall algorithm (weights,  $A[i][i] = 0$ , no path =  $\infty$ )
for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if(          ) A[s][t] =          ;
    
```

1-16

## Use Warshall framework to get shortest path lengths

Warshall algorithm (boolean matrix, no self-loops):

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if(A[s][i] && A[i][t]) A[s][t] = TRUE;
```

Floyd-Warshall algorithm (weights,  $A[i][i]=0$ , no path= $\infty$ )

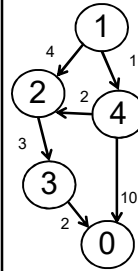
```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] + A[i][t] < A[s][t] )
        A[s][t] = ( A[s][i] + A[i][t] );
```

1-17

## All pairs shortest paths: Unconnected directed graph

Floyd-Warshall algorithm

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] + A[i][t] < A[s][t] )
        A[s][t] = ( A[s][i] + A[i][t] );
```



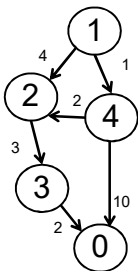
	0	1	2	3	4
0	0	$\infty$	$\infty$	$\infty$	$\infty$
1	$\infty$	0	4	$\infty$	1
2	$\infty$	$\infty$	0	3	$\infty$
3	2	$\infty$	$\infty$	0	$\infty$
4	10	$\infty$	2	$\infty$	0

1-18

## All pairs shortest paths: Unconnected directed graph

Floyd-Warshall algorithm

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] + A[i][t] < A[s][t] )
        A[s][t] = ( A[s][i] + A[i][t] );
```



	0	1	2	3	4
0	0	$\infty$	$\infty$	$\infty$	$\infty$
1	11	0	3	7	1
2	5	$\infty$	0	3	$\infty$
3	2	$\infty$	$\infty$	0	$\infty$
4	10	$\infty$	2	5	0

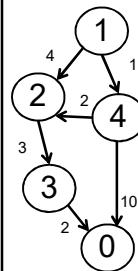
Shortest paths len  $\leq 2$

1-19

## All pairs shortest paths: Unconnected directed graph

Floyd-Warshall algorithm

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] + A[i][t] < A[s][t] )
        A[s][t] = ( A[s][i] + A[i][t] );
```



	0	1	2	3	4
0	0	$\infty$	$\infty$	$\infty$	$\infty$
1	9	0	3	6	1
2	5	$\infty$	0	3	$\infty$
3	2	$\infty$	$\infty$	0	$\infty$
4	7	$\infty$	2	5	0

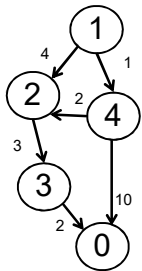
Shortest paths len  $\leq 3$

1-20

## All pairs shortest paths: Unconnected directed graph

Floyd-Warshall algorithm

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      if( A[s][i] + A[i][t] < A[s][t] )
        A[s][t] = ( A[s][i] + A[i][t] );
```



Shortest paths len  $\leq 4$

	0	1	2	3	4
0	0	$\infty$	$\infty$	$\infty$	$\infty$
1	8	0	3	6	1
2	5	$\infty$	0	3	$\infty$
3	2	$\infty$	$\infty$	0	$\infty$
4	7	$\infty$	2	5	0

1-21

## Floyd-Warshall Algorithm: Analysis

- Floyd-Warshall algorithm

```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
    for( t=0; t < V; t++)
      A[s][t] = min( A[s][i] + A[i][t], A[s][t] );
```

$\Theta(?)$

COMP 20003 Algorithms and Data Structures

1-22

## Floyd-Warshall algorithm: Maximum length of path

Note:

No shortest path has *length* (number of segments, *not* distance) greater than  $V-1$

Why not?

COMP 20003 Algorithms and Data Structures

1-23

## Floyd-Warshall algorithm: What is the path?

Floyd-Warshall gives

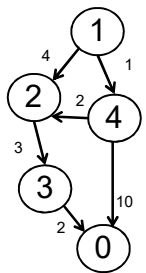
- Distance of shortest path, for all  $a \rightarrow x$
- But does not established the actual paths!

Path information can be obtained through a small addition to the code:

- Keep another 2-dimensional path array
- For each update to distance array, update path array to save:
  - node that made the path shorter

1-24

### All pairs shortest paths: Unconnected directed graph



next along shortest path

	0	1	2	3	4
0	0	\	\	\	\
1	4	0	4	4	1
2	3	\	0	3	\
3	0	\	\	0	\
4	2	\	2	\	0

	0	1	2	3	4
0	0	$\infty$	$\infty$	$\infty$	$\infty$
1	8	0	3	6	1
2	5	$\infty$	0	3	$\infty$
3	2	$\infty$	$\infty$	0	$\infty$
4	10	$\infty$	2	$\infty$	0

shortest path lengths



1-25

### Floyd-Warshall algorithm: What is the path?



Path information can be obtained through a small addition to the code.

For details and Java code, see:

Sedgewick, R., Algorithms in Java, 3<sup>rd</sup> edition,  
Part 5: Graph Algorithms, Addison-Wesley, 308.

1-26

### Floyd-Warshall algorithm: A big assumption



- Assumed graph representation is matrix
- For sparse graphs, adjacency list representation, use Johnson's algorithm
  - Run Dijkstra's single source algorithm for each vertex
  - Use Fibonacci heap for priority queue

D.S. Johnson, "Efficient algorithms for shortest paths in sparse networks", *Journal of the ACM* **24**(1), 1-13, 1977

COMP 20003 Algorithms and Data Structures

1-27