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?

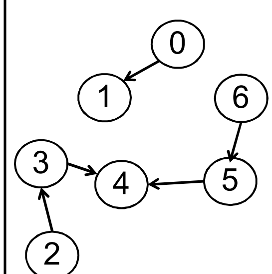
Shortest Paths: Single Source *vs.* All Pairs

Using Dijkstra's multiple times:

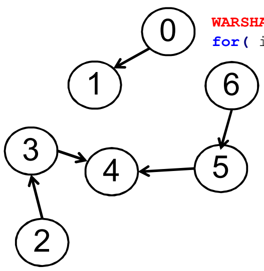
- Dijkstra's algorithm: O((V+E) log V)
- Once for every vertex: O((V^2+VE) log V)
  - O(V^3 log V) for dense graphs.

Can we do better?

Transitive closure: Unconnected directed graph

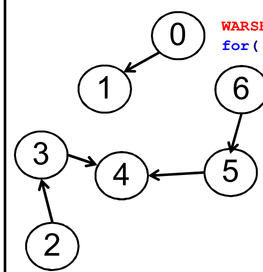


Transitive closure: Unconnected directed graph



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;
```


Transitive closure: Unconnected directed graph



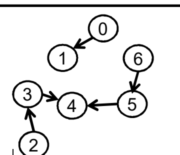
WARSHALL ALGORITHM
```
for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if( A[s][i] && A[i][t])
                A[s][t] = TRUE;
```

	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

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

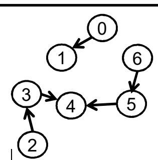


Transitive closure: Unconnected directed graph

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]
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]



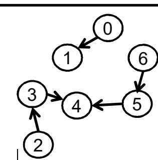
Transitive closure: Unconnected directed graph

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]



Transitive closure: Unconnected directed graph

WARSHALL ALGORITHM

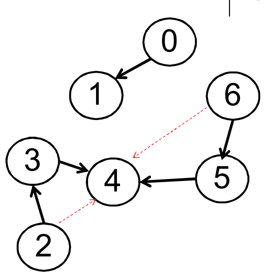
```
for( i=0; i < V; i++)
    for( s=0; s < V; s++)
        for( t=0; t < V; t++)
            if( A[s][i] && A[i][t])
                A[s][t] = TRUE;
```

	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]

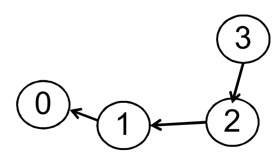
Transitive closure: Unconnected directed graph

	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



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



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;
```

$\Theta(V^3)$ for graph of V vertices and E edges

How does this compare with running Dijkstra's algorithm V times? $\Theta(V * (V+E)\log V)$ could be $\Theta(V^3 \log V)$ for dense graphs

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.

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;
```


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=∞)

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

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=∞)

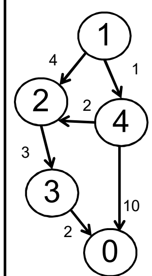
```
for( i=0; i < V; i++)
  for( s=0; s < V; s++)
     for( t=0; t < V; t++)
        if( A[s][i] + A[i][t] < A[s][t] )
              A[s][t] = ( A[s][i] + A[i][t] );
```

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] );
```



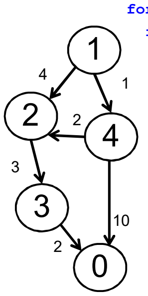
	0	1	2	3	4
0	0	∞	∞	∞	∞
1	∞	0	4	∞	1
2	∞	∞	0	3	∞
3	2	∞	∞	0	∞
4	10	∞	2	∞	0

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] );
```



Shortest paths len ≤ 2

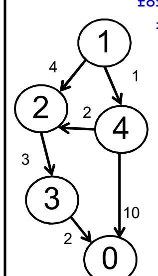
	0	1	2	3	4
0	0	∞	∞	∞	∞
1	11	0	3	7	1
2	5	∞	0	3	∞
3	2	∞	∞	0	∞
4	10	∞	2	5	0

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] );
```



Shortest paths len ≤ 3

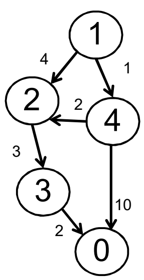
	0	1	2	3	4
0	0	∞	∞	∞	∞
1	9	0	3	6	1
2	5	∞	0	3	∞
3	2	∞	∞	0	∞
4	7	∞	2	5	0

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] );
```



	0	1	2	3	4
0	0	∞	∞	∞	∞
1	8	0	3	6	1
2	5	∞	0	3	∞
3	2	∞	∞	0	∞
4	7	∞	2	5	0

Shortest paths len ≤ 4

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] );
```

$\Theta(V^3)$

Floyd-Warshall agorithm: Maximum length of path

Note:

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

Why not? Because you do not visit edges more than once!

Floyd-Warshall agorithm: What is the path?

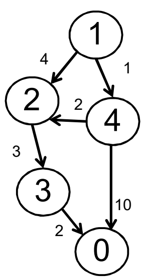
Floyd-Warshall gives

- *Distance* of shortest path, for all a→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

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

shortest path lengths

	0	1	2	3	4
0	0	∞	∞	∞	∞
1	8	0	3	6	1
2	5	∞	0	3	∞
3	2	∞	∞	0	∞
4	10	∞	2	∞	0

Floyd-Warshall agorithm: 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, 3rd edition, Part 5: Graph Algorithms, Addison-Wesley, 308.

Floyd-Warshall agorithm: 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