Transitive Closure and all paths Shortest Paths

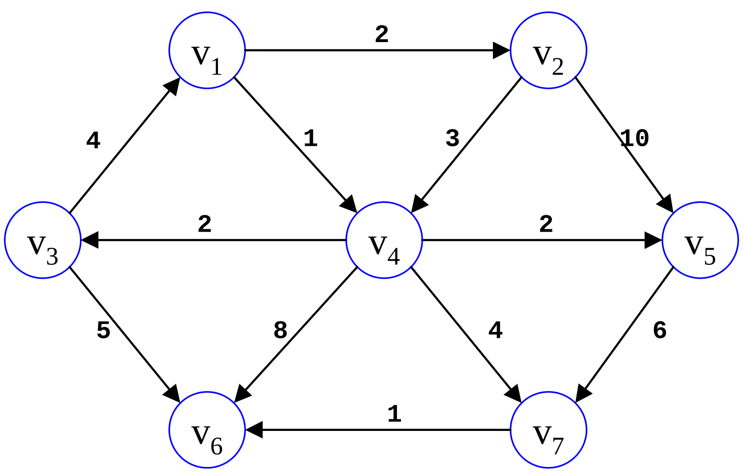
CSE 373

Data Structures

All Pairs Shortest Path

- Given an edge weighted directed graph G = (V,E) find for all u,v in V the length of the shortest path from u to v. Use matrix representation.

$$
\begin{array}{c|c}
C & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} &
\left(\begin{array}{ccccccc}
0 & 2 & : & 1 & : & : & : \\
: & 0 & : & 3 & 10 & : & : \\
4 & : & 0 & : & : & 5 & : \\
: & : & 2 & 0 & 2 & 8 & 4 \\
: & : & : & : & 0 & : & 6 \\
: & : & : & : & : & 0 & : \\
: & : & : & : & : & 1 & 0
\end{array}\right)
\end{array}
$$



: = infinity

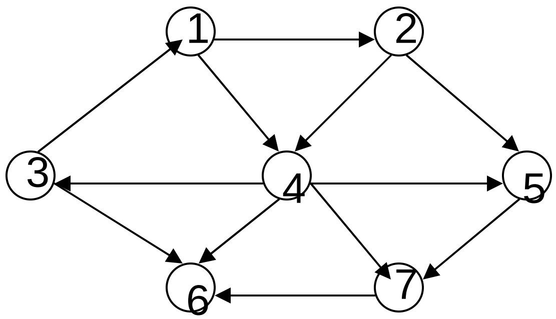
A (simpler) Related Problem: Transitive Closure

- Given a digraph G(V,E) the transitive closure is a digraph G'(V',E') such that
  - V' = V (same set of vertices)
  - If (v_i, v_{i+1},…,v_k) is a path in G, then (v_i, v_k) is an edge of E'

Unweighted Digraph Boolean Matrix Representation

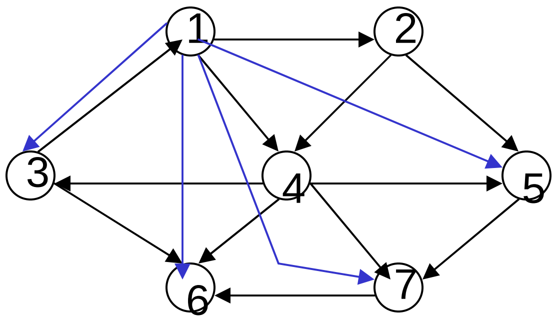
- C is called the connectivity matrix

$$
\begin{array}{c|ccccccc}
C & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
4 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$



Transitive Closure

$$
\begin{array}{c|ccccccc}
C & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
3 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$



On the graph, we show only the edges added with 1 as origin. The matrix represents the full transitive closure.

Finding Paths of Length 2

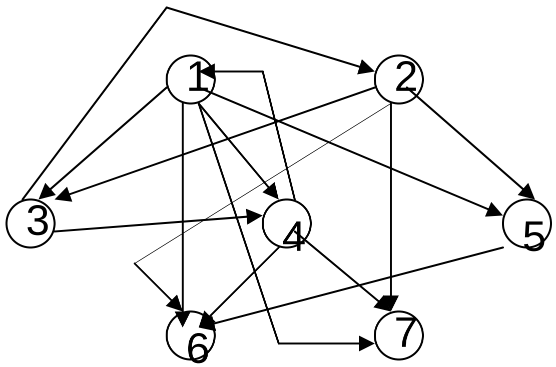
```
Length2 {          //Initialization of C2[i,j]
for k = 1 to n     // to all 0's not shown
 for i = 1 to n do
  for j = 1 to n do
     C2[i,j] := C2[i,j] ∪ (C[i,k] ∩ C[k,j]);
}
where ∩ is Boolean And (&&) and ∪ is Boolean OR (||)
This means if there is an edge from i to k
AND an edge from k to j, then there is a path
of length 2 between i and j.
Column k (C[i,k]) represents the predecessors of k
Row k (C[k,j]) represents the successors of k
```

Paths of Length 2

C

$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Time O(n^3)



C2

$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Transitive Closure

- Union of paths of length 0, length 1, length 2, …, length n-1.
  - Time complexity n * $O(n^3) = O(n^4)$
- There exists a better ($O(n^3)$) algorithm: Warshall's algorithm

Warshall Algorithm (1962)

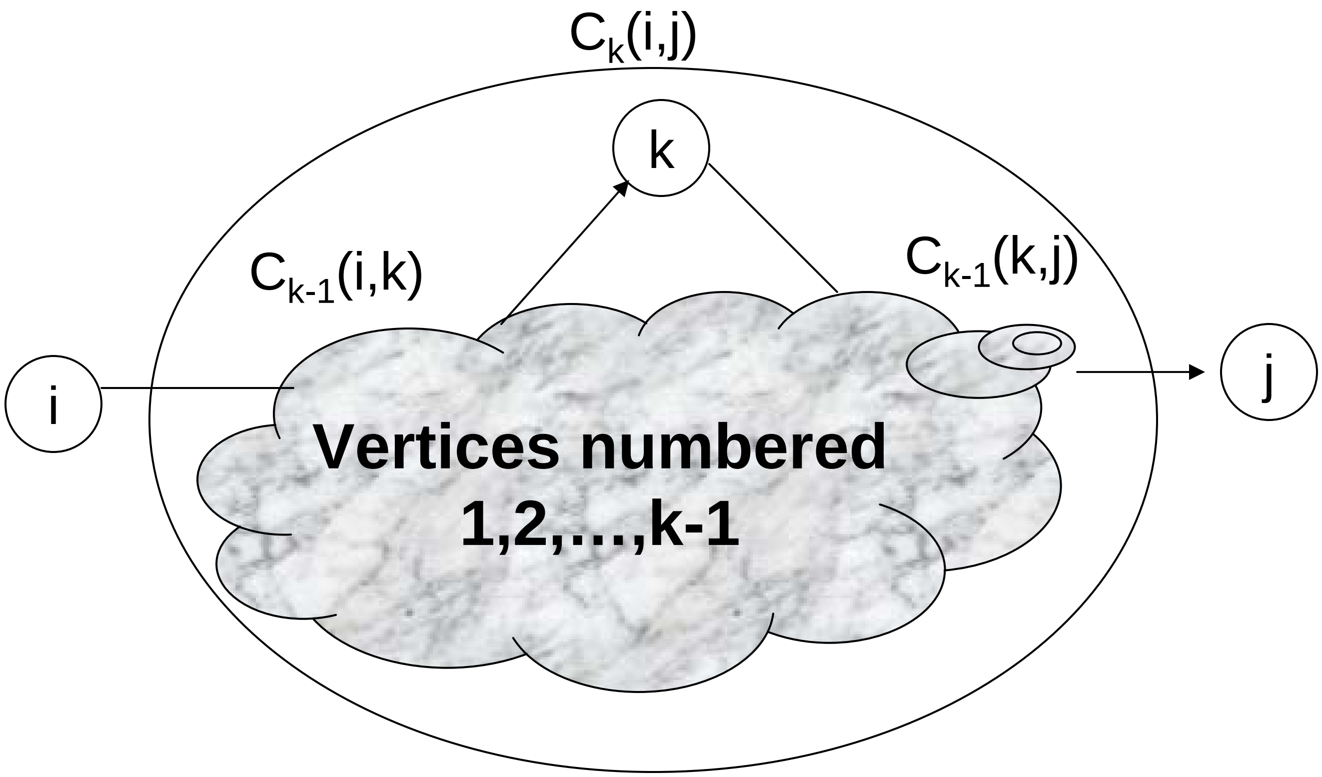
```
TransitiveClosure {
for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
      C [i,j] := C[i,j] ∪ (C[i,k] ∩ C[k,j]);
}

where C[i,j] is the original connectivity matrix
```

Proof of Correctness

- After the k-th time through the loop, C[i,j] =1 if there is a path from i to j that only passes through vertices numbered 1,2,…,k (except for the initial edges)
- Base case: k = 1. C [i,j] = 1 for the initial connectivity matrix (path of length 0) and C [i,j] = 1 if there is a path (i,1,j)

Cloud Argument



Inductive Step

- Assume true for k-1.
  - › All paths from i to j that only go through vertices 1,2, …, k do not go through vertex k at all.
    - $C_k[i,j] = C_{k-1}[i,j]$ ($C_k[i,j]$ is result after k passes)
  - › A path from i to j that goes through (vertices 1,2, …, k must go through vertex k.
    - $C_k[i,j] = C_{k-1}[i,k] + C_{k-1}[k,j]$

Back to Weighted graphs: Matrix Representation

- C[i,j] = the cost of the edge (i,j)
  - C[i,i] = 0 because no cost to stay where you are
  - C[i,j] = infinity (:) if no edge from i to j.

$$
\begin{array}{c|c}
C & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} &
\begin{pmatrix}
0 & 2 & : & 1 & : & : & : \\
: & 0 & : & 3 & 10 & : & : \\
4 & : & 0 & : & : & 5 & : \\
: & : & 2 & 0 & 2 & 8 & 4 \\
: & : & : & : & 0 & : & 6 \\
: & : & : & : & : & 0 & : \\
: & : & : & : & : & 1 & 0
\end{pmatrix}
\end{array}
$$

Floyd – Warshall Algorithm

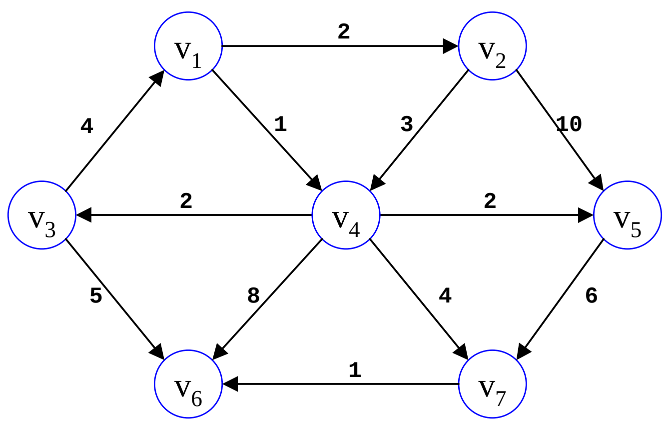
```
All_Pairs_Shortest_Path {
for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
      C[i,j] := min(C[i,j], C[i,k] + C[k,j]);
}

Note x + : = : by definition
```

On termination C[i,j] is the length of the shortest path from i to j.

The Computation

$$\begin{array}{cc} C & \begin{array}{ccccccc}1&2&3&4&5&6&7\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\\6\\7\end{array} & \left(\begin{array}{ccccccc} 0&2&:&1&:&:&: \\ :&0&:&3&10&:&: \\ 4&:&0&:&:&5&: \\ :&:&2&0&2&8&4 \\ :&:&:&:&0&:&6 \\ :&:&:&:&:&0&: \\ :&:&:&:&:&1&0 \end{array}\right)\end{array} \Rightarrow \begin{array}{cc} C & \begin{array}{ccccccc}1&2&3&4&5&6&7\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\\6\\7\end{array} & \left(\begin{array}{ccccccc} 0&2&3&1&3&6&5 \\ 9&0&5&3&5&8&7 \\ 4&6&0&5&4&5&6 \\ 6&8&2&0&2&5&4 \\ :&:&:&:&0&7&6 \\ :&:&:&:&:&0&: \\ :&:&:&:&:&1&0 \end{array}\right)\end{array}$$



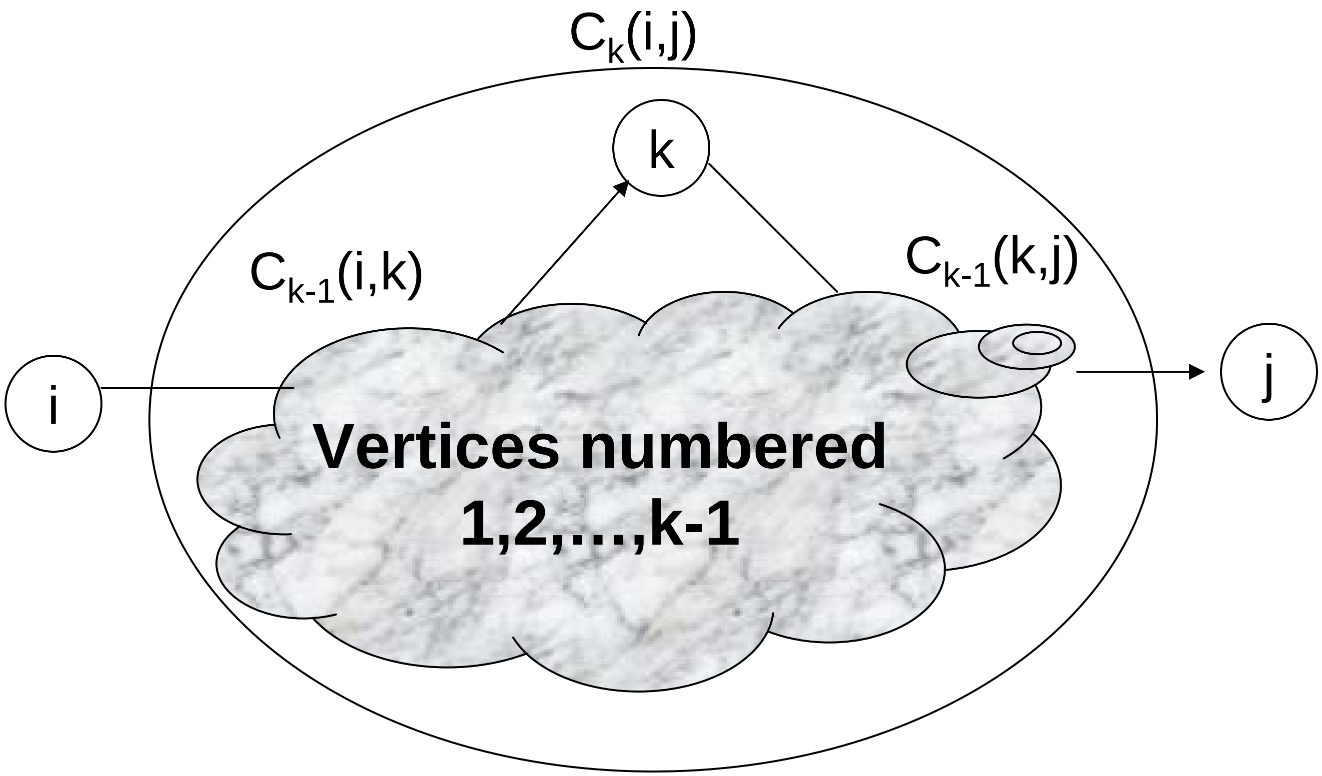
Proof of Correctness

- After the k-th time through the loop C[i,j] is the length of the shortest path that only passes through vertices numbered 1,2,…,k.
  - Let C_k[i,j] be C[i,j] after k time through the loop.
- Base case: k = 0. C_0[i,j] is the cost of an edge that does not pass through any vertices.

Inductive Step

- Assume true for k-1.
  - › A shortest path from i to j that only goes through vertices 1,2, …, k does not go through vertex k at all.
    - $C_k[i,j] = C_{k-1}[i,j]$
  - › All shortest paths from i to j that only go through vertices 1,2, …, k must go through vertex k.
    - $C_k[i,j] = C_{k-1}[i,k] + C_{k-1}[k,j]$

Cloud Argument



Time Complexity of All Pairs Shortest Path

- n is the number of vertices
- Three nested loops. $O(n^3)$
  - Shortest paths can be found too
- Repeated Dijkstra's algorithm
  - $O(n(n + m)\log n)$ (= $O(n^3 \log n)$ for dense graphs).
  - Run Dijkstra starting at each vertex.
  - Dijkstra also gives the shortest paths not just their lengths.