Discussion #34

Warshall’s and Floyd’s Algorithms
Topics

• Faster reachability algorithms
• Warshall’s algorithm
• Weighted graphs
• Floyd’s algorithm
Warshall’s Algorithm

• A more efficient way to compute reachability.

• Algorithm:

1. Initialize adjacency matrix A:
   (For an edge from \( i \) to \( j \), \( a_{ij} \) is set to 1; otherwise, 0.)

2. for (\( k=1 \) to \( n \))
   for (\( i=1 \) to \( n \))
     for (\( j=1 \) to \( n \))
       if (\( a_{ij} = 0 \)) \( a_{ij} \leftarrow a_{ik} \land a_{kj} \)

Clearly more efficient than our straightforward reachability algorithm
\( O(n^3) < O(n^4) \);
but does it work?
Warshall’s Algorithm: Intuitive Idea

We’ll see that after $k$ outer loops, $a_{ij} = 1$ iff there is a path from $i$ to $j$ directly or through $\{1, 2, \ldots, k\}$.

$k$ is called a pivot

- $k=1$: $a_{41} \land a_{13}$ adds (4,3)

- $k=2$: $a_{32} \land a_{26}$ adds (3,6), $a_{62} \land a_{26}$ adds (6,6)

- $k=3$: $a_{13} \land a_{32}$ adds (1,2), $a_{43} \land a_{32}$ adds (4,2), $a_{43} \land a_{36}$ adds (4,6), $a_{13} \land a_{36}$ adds (1,6)

- $k=4$ and $k=5$ add nothing (either no in or no out)

- $k=6$: adds (2,5), (3,5), (1,5), (4,5), (2,2)

\[
\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 & 1 & 1 \\
3 & 0 & 1 & 1 & 0 & 1 & 1 \\
4 & 1 & 1 & 1 & 0 & 1 & 1 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 1 & 0 & 0 & 1 & 1 \\
\end{array}
\]
Warshall’s Algorithm Works

• Are you convinced?
  – If not, how can you become convinced?
  – Prove it — even a sketch helps

• What should you prove?
  – Terminates: clear (can map each loop to a descending sequence on a well-founded poset)
  – Inductive proof based on the loop invariant, which is the key observation about Warshall’s algorithm.

• $W(k)$: After $k$ outer loops, $a_{ij} = 1$ iff there is a path from $i$ to $j$ directly or through $\{1, 2, \ldots, k\}$.

• Proof:
  – Basis: $k = 0$. After 0 outer loops, $a_{ij} = 1$ iff there is a path from $i$ to $j$ directly (by definition of an adjacency matrix).
  – Induction: Show $W(k) \implies W(k+1)$. 
Form of Proof for Warshall’s Algorithm

W(k): After k outer loops, \( a_{ij} = 1 \) iff there is a path from i to j directly or through \{1, 2, \ldots, k\}.

For the induction part, we must prove: \( W(k) \Rightarrow W(k+1) \).

Let P be “\( a_{ij} = 1 \)”, Q be “there is a path from i to j directly or through \{1, 2, \ldots, k\}”, and R be “there is a path from i to j directly or through \{1, 2, \ldots, k+1\}”. Then what we have to prove is that after k outer loops:

\[
(P \iff Q) \Rightarrow (P \iff R)
\]

\[
\equiv (P \iff Q) \Rightarrow ((P \Rightarrow R) \land (R \Rightarrow P))
\]

\[
\equiv ((P \iff Q) \Rightarrow (P \Rightarrow R))
\]

\[
\land ((P \iff Q) \Rightarrow (R \Rightarrow P))
\]

\[
\equiv ((P \iff Q) \land P \Rightarrow R)
\]

\[
\land ((P \iff Q) \land R \Rightarrow P)
\]

Thus, we do two proofs. For the first, we can choose to start by assuming P. For the second, we can choose to start by assuming R. For both proofs, we can also assume the induction hypotheses (P \iff Q). If we wish, we can also, prove by contradiction, and for the first we choose to do so and thus also assume \( \neg R \).
Induction Proof: \((P \iff Q) \land P \Rightarrow R)\)

- Assume \(a_{ij} = 1\) and, by way of contradiction, assume that there does not exist a path from \(i\) to \(j\) directly or indirectly through \(\{1, 2, \ldots, k, k+1\}\).
- There are two cases to consider.
  - Case 1: \(a_{ij} = 1\) in the initialization. But then there is a path from \(i\) to \(j\) directly (as established in the basis) — a contradiction.
  - Case 2: \(a_{ij} = 0\) in the initialization. But then in some iteration, say the \(q^{th}\), \(1 \leq q \leq k+1\), \(a_{ij}\) becomes 1. Hence, in the \(q^{th}\) iteration \(a_{iq} = 1\) and \(a_{qj} = 1\), and thus there is a path from \(i\) to \(j\) through \(\{1, 2, \ldots, k, k+1\}\) — a contradiction.
Induction Proof: \((P \iff Q) \land R \Rightarrow P\)

• Assume there is a path from \(i\) to \(j\) directly or through \(\{1, 2, \ldots, k, k+1\}\). (We need to show that \(a_{ij} = 1\).)

• There are two cases to consider.
  
  – Case 1: \(k+1\) is not on the path from \(i\) to \(j\). Then the path from \(i\) to \(j\) is direct or through \(\{1, 2, \ldots, k\}\). Thus by the induction hypothesis, \(a_{ij} = 1\).

  – Case 2: \(k+1\) is on the path from \(i\) to \(j\). But, now there is a path from \(i\) to \(k+1\) directly or through \(\{1, 2, \ldots, k\}\) and thus by the induction hypothesis, \(a_{i(k+1)} = 1\). Similarly, there is a path from \(k+1\) to \(j\) directly or through \(\{1, 2, \ldots, k\}\) and thus by the induction hypothesis \(a_{(k+1)j} = 1\). Thus, in the \(k+1^{\text{st}}\) loop \(a_{ij} = 1\) since \(a_{ij} = a_{i(k+1)} \land a_{(k+1)j}\).
Shortest Distance in a Weighted Graph

- Minimum Dist[1,4] = 6
- Minimum Dist[3,2] = \(\infty\)
Floyd’s Algorithm

Initialize
for \( k = 1 \) to \( n \)
for \( i = 1 \) to \( n \)
for \( j = 1 \) to \( n \)
if \( \text{Dist}[i,j] > \text{Dist}[i,k] + \text{Dist}[k,j] \)
then \( \text{Dist}[i,j] \leftarrow \text{Dist}[i,k] + \text{Dist}[k,j] \)

\[ O(n^3) \]

Initial \( \text{Dist}[i,j] = \)

\[
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 0 & \infty & 8 & \infty \\
2 & \infty & 0 & 2 & \infty \\
3 & \infty & \infty & 0 & 3 & 1 \\
4 & \infty & \infty & 4 & 0 & \infty \\
5 & \infty & \infty & \infty & \infty & 0 \\
\end{array}
\]

The initial matrix has the edge weights or infinity.

Variation of Warshall’s Algorithm:
Pivot on node \( k \)

Discussion #34 10/17
Floyd’s Algorithm (continued…)

Initialize
for k=1 to n
  for i=1 to n
    for j=1 to n
      if Dist[i,j] > Dist[i,k] + Dist[k,j]
        then Dist[i,j] ← Dist[i,k] + Dist[k,j]

Pivot k:
  k=1: nothing
  k=2: 1→3:3, 1→5:7
Floyd’s Algorithm (continued…)

Initialize
for k=1 to n
   for i=1 to n
      for j=1 to n
         if Dist[i,j] > Dist[i,k] + Dist[k,j]
            then Dist[i,j] ← Dist[i,k] + Dist[k,j]

Pivot k:

k=1: nothing

k=2: 1→3:3, 1→5:7
   1→5:4
   2→4:5
   2→5:3
   4→5:5
   4→4:0 (not changed)

k=3: 1→4:1→3+3→4:6

Discussion #34 12/17
Floyd’s Algorithm (continued…)

Initialize
for k=1 to n
    for i=1 to n
        for j=1 to n
            if Dist[i,j] > Dist[i,k] + Dist[k,j]
                then Dist[i,j] ← Dist[i,k] + Dist[k,j]

Pivot k:

k=1: nothing

k=2: 1→3:3, 1→5:7

k=3: 1→4:1→3+3→4:6
    1→5:4
    2→4:5
    2→5:3
    4→5:5
    4→4:0 (not changed)

k=4: no changes

k=5: nothing
Proof that Floyd’s Algorithm Works

• Termination: Clear (can map each loop to a descending sequence on a well-founded poset)

• Loop Invariant
  – What does the algorithm do?
  – After \( k \) outer loops, each \( \text{Dist}[i,j] \) is as short or shorter than any other path from \( i \) to \( j \) whose intermediate nodes are in \( \{1, \ldots, k\} \).

• Proof (why does it work?):
  – Basis: \( k=0 \), no intermediate nodes; initialized to the shortest path with no intermediate nodes.
  – Induction: We need to prove \( S(k) \Rightarrow S(k+1) \), where \( S(k) \) is the loop invariant.
Induction Part of the Proof

• Assume after $k$ iterations, each $\text{Dist}[i,j]$ is as short or shorter than any other path from $i$ to $j$ whose intermediate nodes are in $\{1, \ldots, k\}$.

• There are two cases for the $k+1$\textsuperscript{st} iteration.
  
  – Case 1: $\text{Dist}[i,j] > \text{Dist}[i,k+1] + \text{Dist}[k+1,j]$. The new distance is $\text{Dist}[i,k+1] + \text{Dist}[k+1,j]$, which is shorter than the previous distance, and thus by the induction hypothesis is as short or shorter than any other path from $i$ to $j$ whose intermediate nodes are in $\{1, \ldots, k+1\}$.
  
  – Case 2: $\text{Dist}[i,j] \leq \text{Dist}[i,k+1] + \text{Dist}[k+1,j]$. The current path is as short or shorter and left unchanged, and thus by the induction hypothesis is as short or shorter than any other path from $i$ to $j$ whose intermediate nodes are in $\{1, \ldots, k+1\}$.
Notes on Warshall’s and Floyd’s Algorithms

• Warshall’s and Floyd’s algorithms do not produce paths.
  – Can keep track of paths
  – Update as the algorithms execute

• Warshall’s and Floyd’s algorithms only work for directed graphs.
  – For undirected graphs, replace each edge with two directed edges
  – Then, run algorithms as given.
Notes on Warshall’s and Floyd’s Algorithms

• With DFS, cycle detection can be done in $O(m)$, $O(n^2)$ in the worst case. This beats Warshall’s algorithm, $O(n^3)$—i.e. run algorithm and check to see if a 1 is on the diagonal.

• With DFS, reachability can be done in $O(m)$, $O(n^2)$ in the worst, which beats Warshall’s $O(n^3)$, but Warshall’s tests reachability for all pairs.