CSC 317: Data Structures and Algorithm Analysis

Dilip Sarkar

Department of Computer Science
University of Miami

Outline I

- All-pairs Shortest Paths
  - Shortest Paths and Matrix Multiplication
  - Floyd-Warshall Algorithm

Structure of a Shortest Path (SP)

Three steps for developing a dynamic programming algorithm

- Characterize the structure of an optimal solution
- Recursively define the value of an optimal solution
- Compute the value of an optimal solution in a bottom-up fashion

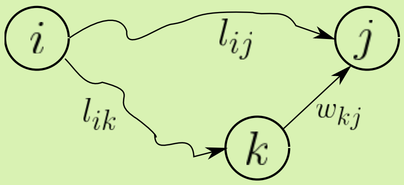


Figure: Extend path by one more edge

- SP distances with at most m edges are known
- Let l_{ij} and l_{ik} be the SPs from i to j and k with at most m edges
- And, let l_{ik} be the path length to k from node i using m edges
- We want to increase the number of edges to $(m+1)$, if that reduces the path length
- Let w_{kj} be the distance from node k to node j
- $l_{ij} = \min\{l_{ij}, l_{ik} + w_{kj}\}$, where w_{kj} weight of the edge from k to j

Computing the SP weights bottom up

- Given weight matrix $w = (w_{ij})$ of directed graph
- Compute a series of shortest-distance matrices $L^{(1)}, L^{(2)}, \cdots, L^{(n-1)}$
- $L^{(m)}$ contains all-pairs shortest-paths of m edges for $1 < k < n$.
- Note that $W = L^{(1)}$, shortest paths of length one — the graph

  EXTEND-SHORTEST-PATHS$(L^{(k)}, w)$

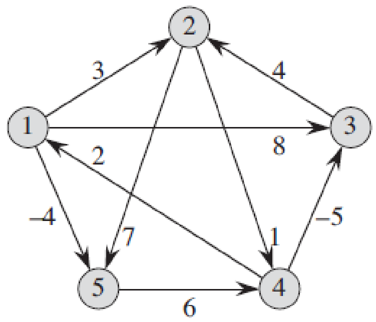
  for $i = 1$ to n
    for $j = 1$ to n
      $L_{ij}^{(k+1)} = \infty$
      for $m = 1$ to n
        $L_{ij}^{(k+1)} = \min\left(L^{(k)}, L_{im}^{(k)} + w_{mj}\right)$
  return $L^{(k+1)}$

Slow All-Pairs Shortest Paths

- Given weight matrix $w = (w_{ij})$ of directed graph
- Compute a series of shortest-distance matrices $L^{(1)}, L^{(2)}, \cdots, L^{(n-1)}$
- $L^{(k)}$ contains all-pairs shortest-paths of k edges for $1 \le k < n$.
- Note that $W = L^{(1)}$, shortest paths of length one — the graph

SLOW-ALL-PAIRS-SHORTEST-PATHS(w)
$L^{(1)} = w$
for $m = 2$ to $n - 1$
    $L^{(m)} =$
return $L^{(n-1)}$

Printing All-pair Shortest Paths Algorithm



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Figure: All pair SP computation

Extending path length and Matrix multiplication

EXTEND-SHORTEST-PATHS($L^{(k)}, w$)
for $i = 1$ to n
  for $j = 1$ to n
    $L_{ij}^{(k+1)} = \infty$
    for $m = 1$ to n
      $L_{ij}^{(k+1)} = \min\left(L^{(k)}, L_{im}^{(k)} + w_{mj}\right)$
return $L^{(k+1)}$

MATRIX-MULTIPLICATION(A, B)
for $i = 1$ to n
  for $j = 1$ to n
    $C[ij] = 0$
    for $m = 1$ to n
      $C_{ij} = C_{ij} + A_{im} * B_{mj}$
return C

Note that both algorithms are identical if we replace $(C_{ij} = C_{ij} + A_{im} * B_{mj})$ with $(L_{ij}^{(k+1)} = \min\left(L^{(k)}, L_{im}^{(k)} + w_{mj}\right))$

Using matrix multiplication notations we have,

$L^{(1)} = W,$

$L^{(2)} = L^{(1)} \cdot W = W^2$

$L^{(3)} = L^{(2)} \cdot W = W^3$

$\ldots$

$L^{(n-1)} = L^{(n-2)} \cdot W = W^{n-1}$

Faster All-pair Shortest Paths Algorithm

FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
$n = W.rows$
$L^{(1)} = W$
$m = 1$
While $m < n - 1$
  let $L^{(2m)}$ be a new $n \times n$ matrix
  $L^{(2m)} =$ EXTEND-SHORTEST-PATHS($L^{(m)}, L^{(m)}$)
  $m = 2m$
return $L^{(m)}$

Structure of a SP

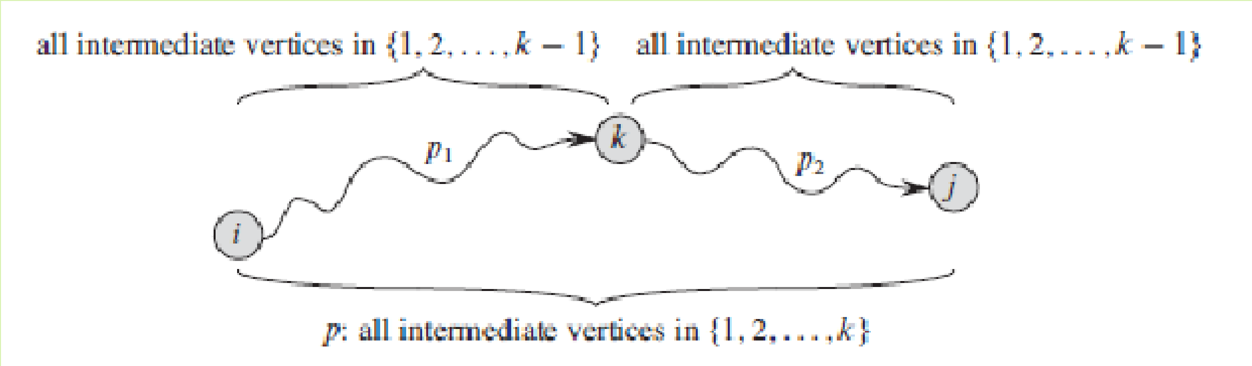


Figure: Strusture of a SP

Path p_1 has all intermediate vertices in $\{1, 2, \cdots, (k-1)\}$
Path p_2 has all intermediate vertices in $\{1, 2, \cdots, (k-1)\}$
New path p containing vertex k
A recursive solution

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1 \end{cases}$$

Floyd-Warshall Algorithm

FLOYD-WARSHALL(W)
$n = W.rows$
$D^{(0)} = W$
for $k = 1$ to n
  let $D^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
  for $i = 1$ to n
    for $j = 1$ to n
      $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $D^{(n)}$

Transitive-Closure

TRANSITIVE-CLOSURE(G)

$n = |G.V|$
let $T^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
`for` $i = 1$ `to` n
    `for` $j = 1$ `to` n
        `if` $i == j$ `or` $(i, j) \in G.E$
            $t_{ij}^{(0)} = 1$
        `else` $t_{ij}^{(0)} = 0$
`for` $k = 1$ `to` n
    let $T^{(k)} = (t_{ij}^{(k)})$ be a new $n \times n$ matrix
    `for` $i = 1$ `to` n
        `for` $j = 1$ `to` n
            $t_{ij}^{(k)} = t_{ij}(k-1) \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$
`return` $T^{(n)}$