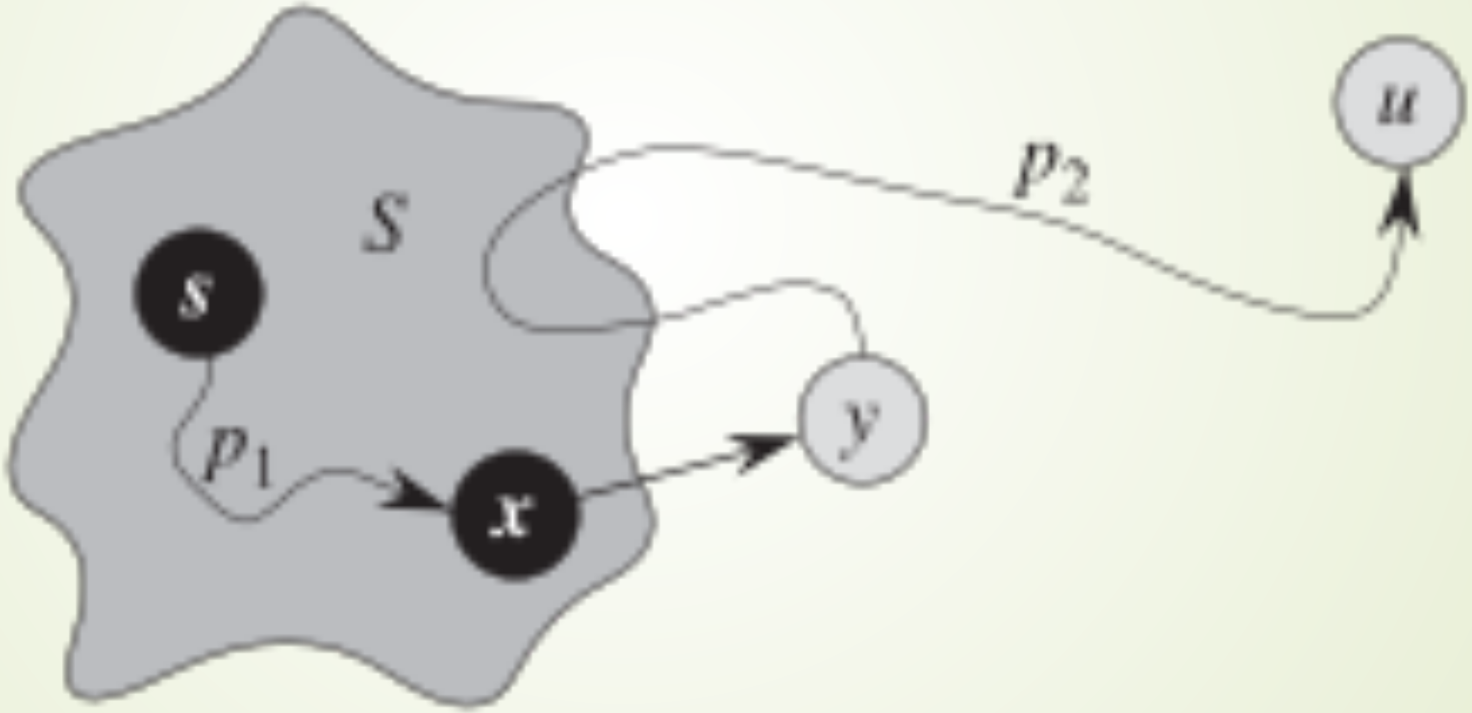
COT 5407: Introduction to Algorithms

Giri NARASIMHAN

www.cs.fiu.edu/~giri/teach/5407S19.html

Correctness of Dijkstra's Alg



Analysis of Dijkstra's Algorithm

- O(n) calls to INSERT, EXTRACT-MIN
- O(m) calls to DECREASE-KEY

Approach	Insert	Dec-Key	Extract-Min	Total
PQ in Arrays	O(1)	O(1)	O(n)	O(n^2)
Heaps	O(log n)	O(log n)	O(log n)	O((m+n)log n)
Fibonacci Heaps	O(1)*	O(1)*	O(log n)*	O(m + n log n)*

* Amortized Time Complexity

SSSP Algorithms

- Dijkstra's algorithm (only non-negative edges allowed)
  - Best: O(m + n log n)
- Bellman-Ford algorithm (allows non-negative edges, but less efficient)
  - Repeated RELAX steps until we have answers
  - O(mn) time complexity

All Pairs Shortest Path Algorithm

Need to find shortest paths (or length) between every pair of vertices

- Invoke Dijkstra's SSSP algorithm n times.
- Or an alternative approach …

Structure of a Shortest Path

- Optimal substructure property
  - Every subpath of a shortest path is “optimal”, i.e., it is a shortest path between the relevant vertices.
- Can we use DP?
  - Invent appropriate subproblems to cast it as a DP

Idea!

- In iteration k, find $SP_k(u,v)$, shortest paths between u and v that use at most k edges
- Iteration 1: find all shortest paths that use at most 1 edge
  - Every edge (u,v) is a SP between u and v
  - Every non-edge (u,v) means no SP exists between u and v using at most one edge

Iteration k

- We already have $SP_{k-1}(u,v)$ from iteration k-1
- If path of length k between u & v exists, then path of length k-1 exists between u and a neighbor of v
- $SP_k(u,v)$ can be computed as follows:
  - $SP_k(u,v) = \min (SP_{k-1}(u,v), \min_w \{SP_{k-1}(u,w) + SP_1(w,v)\})$
  - This is our recurrence relation

Toward APSP

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j\,, \\ \infty & \text{if } i \neq j\,. \end{cases}$$

Extend SP to use one extra edge.

EXTEND-SHORTEST-PATHS(L, W)

```
n = L.rows
let L' = l'_ij  be a new n   n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

$SP_k(i,j) = l_{ij}(i,j)$

APSP Algorithm

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
for m = 2 to n    1
        let L^(m) be a new n    n matrix
        L^(m) = EXTEND-SHORTEST-PATHS(L^(m  1), W)
return L^(n  1)
```

O(n^4) time complexity

APSP Algorithm – Matrix Mult

EXTEND-SHORTEST-PATHS(L, W)

```
n = L.rows
let L' = l'_ij be a new n   n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

SQUARE-MATRIX-MULTIPLY(A, B)

```
n = A.rows
let C be a new n   n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik   b_kj
return C
```

Intermediate Matrices …

$$L^{(1)} = \left.\begin{matrix} 0 & 3 & 8 & \infty & 4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & 5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{matrix}\right) \qquad L^{(2)} = \left.\begin{matrix} 0 & 3 & 8 & 2 & 4 \\ 3 & 0 & 4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 1 & 5 & 0 & 2 \\ 8 & \infty & 1 & 6 & 0 \end{matrix}\right)$$

$$L^{(3)} = \left.\begin{matrix} 0 & 3 & 3 & 2 & 4 \\ 3 & 0 & 4 & 1 & 1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & 1 & 5 & 0 & 2 \\ 8 & 5 & 1 & 6 & 0 \end{matrix}\right) \qquad L^{(4)} = \left.\begin{matrix} 0 & 1 & 3 & 2 & 4 \\ 3 & 0 & 4 & 1 & 1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & 1 & 5 & 0 & 2 \\ 8 & 5 & 1 & 6 & 0 \end{matrix}\right)$$

Improved Idea

```
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    n matrix
    L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), L^(m))
    m = 2m
return L^(m)
```

$O(n^3 \log n)$ time complexity

Repeating Squaring Idea

$$
\begin{aligned}
L^{(1)} &= W\,, \\
L^{(2)} &= W^2 &&= W \cdot W\,, \\
L^{(4)} &= W^4 &&= W^2 \cdot W^2 \\
L^{(8)} &= W^8 &&= W^4 \cdot W^4\,, \\
&\ \ \vdots \\
L^{(2^{\lceil \lg(n-1) \rceil})} &= W^{2^{\lceil \lg(n-1) \rceil}} &&= W^{2^{\lceil \lg(n-1) \rceil - 1}} \cdot W^{2^{\lceil \lg(n-1) \rceil - 1}}\,.
\end{aligned}
$$

Floyd-Warshall's Algorithm

- $SP_k(u,v)$, shortest paths between u and v that use at most k edges
- Old definition

- $SP_k(u,v)$, shortest paths between u and v that uses intermediate vertices from {1,2,…,k}
- New definition

Recurrence Relation

- Old Relation
- $SP_k(u,v) = \min(\, SP_{k-1}(u,v),\ \min_w \{SP_{k-1}(u,w) + SP_1(w,v)\})$
- New Relation
- $SP_k(u,v) = \min(\, SP_{k-1}(u,v),\ SP_{k-1}(u,k) + SP_{k-1}(k,v)\})$

Floyd-Warshall: Improved APSP

FLOYD-WARSHALL(W)

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

$O(n^3)$ time complexity

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Figure 25.4 The sequence of matrices $D^{(k)}$ and $\Pi^{(k)}$ computed by the Floyd-Warshall algorithm for the graph in Figure 25.1.

Figure 14.38

Worst-case running times of various graph algorithms

Type of Graph Problem	Running Time	Comments
Unweighted	$O(\|E\|)$	Breadth-first search
Weighted, no negative edges	$O(\|E\| \log \|V\|)$	Dijkstra's algorithm
Weighted, negative edges	$O(\|E\| \cdot \|V\|)$	Bellman–Ford algorithm
Weighted, acyclic	$O(\|E\|)$	Uses topological sort