Matrix Based Methods

Two Graph algorithms

1. Bellman Ford Algorithm: it solves single source shortest path problem that allows negative edge costs (no negative cycle, of course)
2. Floyd Warshall Algorithm: It solves all pair shortest path problem that allows negative edge costs (no negative cycle, of course)

Bellman Ford algorithm

```
BELLMAN-FORD(G, w, s)
  INIT-SINGLE-SOURCE(G, s)
  for i = 1 to |G.V| - 1
      for each edge (u, v) ∈ G.E
          RELAX(u, v, w)
  for each edge (u, v) ∈ G.E
      if v.d > u.d + w(u, v)
          return FALSE
  return TRUE
```

Figure : Pseudocode for Bellman Ford Algorithm

Example: Bellman Ford algorithm

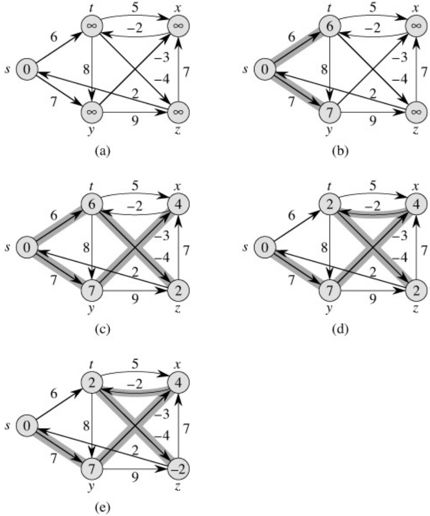


Figure : Running Bellman Ford Algorithm

Example

We will go thorough an example in the note from previous semesters to show how Bellman Ford algorithm works.

Question

Fill in the blanks:

Given a graph $G = (V, E)$. The worst case running time for running *Bellman Ford Algorithm* over G is (in Big-O notation with parameters $|V|$ and $|E|$): $O(\qquad\qquad)$

Floyd Warshall Algorithm

1. $G=(V,E)$: a *directed* graph with edge weights $w_{i,j}$ for each each $(i,j) \in E$
2. Use the weight *matrix* W as input
3. Algorithm is derived from the following recursive relation:

$$d_{i,j}^{k} = \begin{cases} w_{i,j} & k = 0 \\ \min\left(d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{j,j}^{k-1}\right) & k \geq 1 \end{cases}$$

4. Implement via *table based* iteration

Question Which data structure you will choose if you implement Floyd Warshall ?

Floyd Warshall Algorithm: Pseudocode

FLOYD-WARSHALL(W, n)

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

Figure : Floyd Warshall Algorithm

Question What are the common features in Floyd Warshall and other matrix based graph algorithms ?

Floyd Warshall Relation

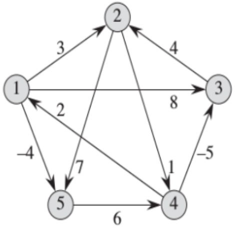
1. Recall the recursive relation:

$$d_{i,j}^{k} = \text{the length of the shortest path from vertex } i \text{ to } j$$
$$\text{via intermediate vertices from vertices } 1, \ldots, k$$

2. $$d_{i,j}^{k} = \begin{cases} w_{i,j} & k = 0 \\ \min\left(d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{k,j}^{k-1}\right) & k \geq 1 \end{cases}$$

3. Use a table based iteration to avoid re-computing known sub-cases.

Illustrations: Floyd Warshall Algorithm I

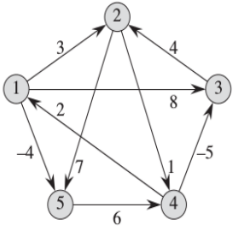


$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL}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

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

Figure : Apply Floyd Warshall: Stage 1

Illustrations: Floyd Warshall Algorithm II



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

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

Figure : Apply Floyd Warshall: Stage 2

An algorithm design technique: dynamic programming

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

Question

Question Fill in the blanks:

The algorithm design technique used in the Floyd Warshall Algorithm is ____________________