Dynamic Programming (DAA, M.Tech + Ph.D.)

By:

Sunil Kumar Singh, PhD
Assistant Professor,
Department of Computer Science and Information Technology

School of Computational Sciences, Information and Communication Technology,
Mahatma Gandhi Central University, Motihari
Bihar, India-845401

Outline

- Dynamic Programming
- Floyd Warshall Algorithm
- Conclusion
- References

Dynamic Programming

- Dynamic programming is an algorithm design paradigm: like divide-and-conquer algorithm.
- Usually it is for solving optimization problems. For exam. Shortest path

- **Elements of dynamic programming**
  - ✓ Big problems break up into little problems
  - ✓ The optimal solution of a problem can be expressed in terms of optimal solutions of smaller sub-problems.
  - ✓ The sub-problem overlap a lot.

Cont..

- **Optimal substructure:**

  optimal solutions to sub-problems are sub-solutions to the optimal solution of the original problem.

- **Overlapping subproblems:**

  the subproblems show up again and again

Cont..

- **Using these properties, we can design a dynamic programming algorithm:**
  - ✓ Keep a table of solutions to the smaller problems
  - ✓ Use the solutions in the table to solve bigger problems
  - ✓ At the end we can use information we collected along the way to find the optimal solution.

Floyd Warshall Algorithm

- The Floyd Warshall Algorithm is a Graph Analysis Algorithm for finding shortest paths in weighted, directed graph.
- It aims to compute the shortest path form each vertex to every other nodes.
- It uses a Dynamic Programming methodology to solve the all Pair Shortest Path Problem.

Cont..

```
FLOYD-WARSHALL(W, n)
{
    D^(0) = W;

    for k := 1 to n
        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);
}
```

FLOYD-WARSHALL(W, n)

{

$D^{(0)} = W;$

for $k := 1$ to n

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 $D^{(n)}$;

}

Complexity

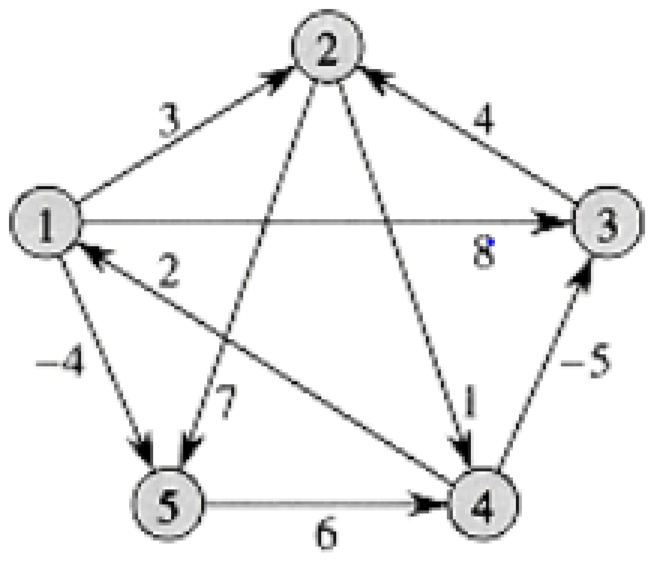
- The algorithm is required to apply n-times, each time choosing different as the source.

- So the total computations time is

$$n \; X \; O(n^2) = O(n^3)$$

- The time complexity is same for best, average and worst case.

Example



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

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

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

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

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

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

Path construction

$$\pi_{ij}^{(k)} = \begin{cases} NIL & \text{if } i = j \text{ or } w_{ij} = \infty \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty \end{cases}$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Shortest path

$$\Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$\Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$\Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$\Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$\Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$\Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Applications

- Fast computation of path finder networks
- Optimal routing
- All pair shortest path
- Detecting negative-weight cycles in graph
- Floyd-warshall algorithm is applied in electrical power system to get the shortest electrical path.

References

1. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
2. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to algorithms second edition." *The Knuth-Morris-Pratt Algorithm, year* (2001).
3. Seaver, Nick. "Knowing algorithms." (2014): 1441587647177.

Thank You