Unit -III Part 2

3.All pairs shortest path problem

- All-pairs shortest-paths problem is to find a shortest path from u to v for every pair of vertices u and v.
- Although this problem can be solved by running a single-source algorithm once from each vertex, it can usually be solved faster using the dynamic programming technique.

Solving All pairs shortest path problem by dynamic programming

Step 1: - Optimal substructure of a shortest path

Shortest-paths algorithms typically rely on the property that a shortest path between two vertices contains other shortest paths within it.

Step 2:- A recursive solution

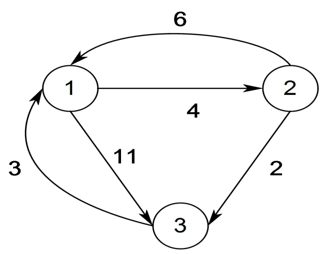
$$D^k[i,j] = \begin{cases} C[i,j] & \text{for } k=0 \\ \min\left\{ D^{k-1}[i,j],\ D^{k-1}[i,k]+D^{k-1}[k,j] \right\} & \text{for } k > 0 \end{cases}$$

where C[i,j] is the cost matrix of the given graph.

Step 3:- Computing the distance matrices D^k where k= 1, 2, ….. , n.

Step 4:- Finally D^n matrix gives the shortest distance form every vertex I to every other vertex j.

Example :- Find the shortest path between all pair of nodes in the following graph.



Solution: - The cost matrix of the given graph is as follows

$$C[i, j] = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

We know $D^0[i,j] = C[i,j] = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$

Now we have to calculate $D^1[i,j]$

$D^1[1,1] = \min\{ D^0[1,1], D^0[1,1]+ D^0[1,1]\} = \min\{0, 0+0\} = 0$
$D^1[1,2] = \min\{ D^0[1,2], D^0[1,1]+ D^0[1,2]\} = \min\{4, 0+4\} = 4$
$D^1[1,3] = \min\{ D^0[1,3], D^0[1,1]+ D^0[1,3]\} = \min\{11, 0+11\} = 11$
$D^1[2,1] = \min\{ D^0[2,1], D^0[2,1]+ D^0[1,1]\} = \min\{6, 6+0\} = 6$
$D^1[2,2] = \min\{ D^0[2,2], D^0[2,1]+ D^0[1,2]\} = \min\{0, 6+4\} = 0$
$D^1[2,3] = \min\{ D^0[2,3], D^0[2,1]+ D^0[1,3]\} = \min\{2, 6+11\} = 2$
$D^1[3,1] = \min\{ D^0[3,1], D^0[3,1]+ D^0[1,1]\} = \min\{3, 3+0\} = 3$
$D^1[3,2] = \min\{ D^0[3,2], D^0[3,1]+ D^0[1,2]\} = \min\{\infty, 3+4\} = 7$
$D^1[3,3] = \min\{ D^0[3,3], D^0[3,1]+ D^0[1,3]\} = \min\{0, 3+11\} = 0$

Thus

$$D^1[i,j] = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Similarly using the same procedure we get

$$D^2[i,j] = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \quad \text{and} \quad D^3[i,j] = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

As no of nodes in the given graph are 3, So $D^3[i,j]$ gives the shortest distance from every vertex i to every other vertex j.

Algorithm:

Algorithm AllPaths (cost, A, n)

```
{
    for i = 1 to n do
        for j = 1 to n do
            A[i, j] = cost[i, j];
    for k= 1 to n do
      for i = 1 to n do
            for j = 1 to n do
                A[i, j] = min { A [i,j], A [i,k]+ A [k,j] };
}
```

$$D^k[i,j] = \begin{cases} C[i,j] & \text{for } k=0 \\ \min\left\{ D^{k-1}[i,j],\ D^{k-1}[i,k]+D^{k-1}[k,j] \right\} & \text{for } k > 0 \end{cases}$$

Time Complexity: -

1. The time needed by All Paths algorithm is especially easy to determine because the loop is independent of the data in the matrix D.
2. The D [i, j] is obtained after the statement is iterated n^3 times.
3. So the time complexity of algorithm is Θ (n^3).

4.The String Editing Problem

- Given two strings, X and Y and edit operations . find minimum number operations required to convert string X into Y.
- As the problem consist of many sub problems which are solved repeatedly so we have over lapping sub problems.
- Hence problem can be solved using dynamic programming in bottom-up manner.
- Edit operations allowed are
  1. Insertion:Insert a new character.
  2. Deletion: Delete a character.
  3. Replace:Replace one character by another.

Example:

X = "aabab"

Y = "abbaa"

X can be converted to Y by changing 2nd character in to b and last character in to a

Approach:

Start comparing one character at a time in both strings. Here we are comparing string from right to left .

- If last characters in both the strings are same then just ignore the character and solve the rest of the string recursively.
- Else if last characters in both the strings are not same then we will try all the possible operations (insert, replace, delete) and get the solution for rest of the string recursively for each possibility and pick the minimum out of them.

Let's say given strings are X and Y with lengths m and n respectively.

case 1: last characters are same , ignore the last character.
recursively solve for m-1, n-1

case 2: last characters are not same then try all the possible operations
recursively.

a. Insert a character into X (same as last character in string Y so that last character in both the strings are same): now X length will be m+1, Y length : n, ignore the last character and recursively solve for m, n-1.

b. Remove the last character from string X. now s1 length will be m-1, Y length : n, recursively solve for m-1, n.
c. Replace last character into X (same as last character in string Y so that last character in both the strings are same): X length will be m, Y length : n, ignore the last character and recursively solve for m-1, n-1.

Cost function defined as

$$cost(i,j) = \begin{cases} 0 & i = j = 0 \\ cost(i-1,0) + D(x_i) & j = 0,\ i > 0 \\ cost(0,j-1) + I(y_j) & i = 0,\ j > 0 \\ cost'(i,j) & i > 0,\ j > 0 \end{cases}$$

$$\text{where } cost'(i,j) = \min \{ \begin{array}{l} cost(i-1,j) + D(x_i), \\ cost(i-1,j-1) + C(x_i,y_j), \\ cost(i,j-1) + I(y_j)\ \} \end{array}$$

Where $D(x_i)$ indicate deletion, $I(y_j)$ indicate insertion and $C(x_i,y_j)$ indicate change operation.
Example:
X=aabab,Y=babb

$$\begin{aligned} cost(1,1) &= \min\ \{cost(0,1) + D(x_1), cost(0,0) + C(x_1,y_1), cost(1,0) + I(y_1)\} \\ &= \min\ \{2,2,2\} = 2 \end{aligned}$$

Next is computed $cost(1,2)$.

$$\begin{aligned} cost(1,2) &= \min\ \{cost(0,2) + D(x_1), cost(0,1) + C(x_1,y_2), cost(1,1) + I(y_2)\} \\ &= \min\ \{3,1,3\} = 1 \end{aligned}$$

i ↓ \ j →	0	1	2	3	4
0	0	1	2	3	4
1	1	2	1	2	3
2	2	3	2	3	4
3	3	2	3	2	3
4	4	3	2	3	4
5	5	4	3	2	3

Time Complexity:
So in worst case we need to perform the operation on every character of the string, since we have operations on table of size m*n,
Time Complexity will be **O(mn).**
Let's see if there are overlapping sub-problems.

5.The travelling sales person problem

- The problem is to find a sequence of visiting n cities (and return to the starting city) with the objective of minimizing the total cost of travel.
- i.e Given a graph G=(V,E) representing n cities find minimum cost round trip path.
- The input data is a cost matrix C, where the (i,j) entry is the cost of going from city i to city j.
- Minimum cost of visiting a city in set S from I computed as

  where V represents cities to visit and is represented as set S

$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V - \{1, k\})\}$$

In a generalized form

$$g(i, S) = \min_{j \in S}\{c_{ij} + g(j, S - \{j\})\}$$

The cost of returning back to home city from city I is

$g(i,\phi)=C_{i,1}$

Time complexity is

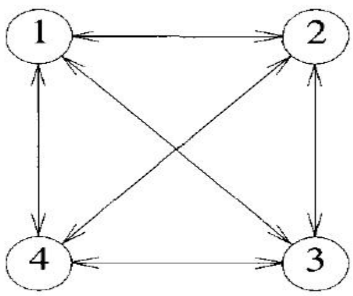
$O(n^2 2^n)$

Space complexity is

$n2^n$

Example:

Find the minimum cost round trip cost for the following travelling sales person problem instance.



$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Thus $g(2,\phi) = c_{21} = 5, g(3,\phi) = c_{31} = 6$, and $g(4,\phi) = c_{41} = 8$.

using cost function

$$\begin{aligned} g(2,\{3\}) &= c_{23} + g(3,\phi) = 15 & g(2,\{4\}) &= 18 \\ g(3,\{2\}) &= 18 & g(3,\{4\}) &= 20 \\ g(4,\{2\}) &= 13 & g(4,\{3\}) &= 15 \end{aligned}$$

Next, we compute $g(i, S)$ with $|S| = 2$, $i \neq 1$, $1 \notin S$ and $i \notin S$.

$$\begin{aligned} g(2,\{3,4\}) &= \min\ \{c_{23} + g(3,\{4\}), c_{24} + g(4,\{3\})\} = 25 \\ g(3,\{2,4\}) &= \min\ \{c_{32} + g(2,\{4\}), c_{34} + g(4,\{2\})\} = 25 \\ g(4,\{2,3\}) &= \min\ \{c_{42} + g(2,\{3\}), c_{43} + g(3,\{2\})\} = 23 \end{aligned}$$

Finally

$$\begin{aligned} g(1,\{2,3,4\}) &= \min\{c_{12} + g(2,\{3,4\}), c_{13} + g(3,\{2,4\}), c_{14} + g(4,\{2,3\})\} \\ &= \min\{35, 40, 43\} \\ &= 35 \end{aligned}$$