Dynamic Programming (5)

By: Aminul Islam

Minimum Cost Path Problem

Minimum Cost Path Problem Statement: Given a two dimensional cost matrix having a cost at each cell. The cost is to travel through that cell. Find the minimum cost it will take to reach bottom-right corner cell (m, n) from top left corner cell (0, 0). The only allowed directions to move from a cell are right or down.

cost	1	7	9	2
	8	6	3	2
	1	6	7	8
	2	9	8	2

Algorithm: Recursive solution to Minimum Cost Path Problem

```
int minCost(int cost[][], int m, int n)
{
    if (n < 0 || m < 0)
        return INT_MAX_VALUE;
    else if (m == 0 && n == 0)
        return cost[m, n];
    else
        return cost[m, n] + min( minCost(cost, m-1, n),
minCost(cost, m, n-1));
}</pre>
Time complexity, T(n) =
```

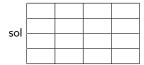
Algorithm: DP solution to Minimum Cost Path

```
Problem
int minCost( int cost[][], int n, int m )
   int sol[n, m];
   int i, j;
   sol[0, 0] = cost[0, 0];
   for(int j=1; j < m; j++) {
      sol[0, j] = sol[0, j-1] + cost[0, j];
   for(int i=1; i < n; i++) {
      sol[i, 0] = sol[i-1, 0] + cost[i, 0];
   for (i=1: i<n: i++)
      for (j=1; j < m; j++)
        sol[i, j] = cost[i, j] + min(sol[i-1, j], sol[i, j-1])
   return sol[n, m];
```

Example of Minimum Cost Path Problem (DP sol.)

cost 8 6 3 2 1 6 7 8 2 9 8 2

```
sol[0, 0] = cost[0, 0];
for(int j=1; j < m; j++) {
    sol[0, j] = sol[0, j-1] + cost[0, j];
}
for(int i=1; i < n; i++) {
    sol[i, 0] = sol[i-1, 0] + cost[i, 0];
}
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
return sol[n, m];</pre>
```



Time Complexity —

• The cost of Minimum Cost Path is:

Exercise: Minimum Cost Path Problem

cost | 1 | 3 | 5 | 8 | | 4 | 2 | 1 | 7 | | 4 | 3 | 2 | 3 |

Exercise: Revised Minimum Cost Path Problem

Minimum Cost Path Problem Statement: Given a two dimensional cost matrix having a cost at each cell. The cost is to travel through that cell. Find the minimum cost it will take to reach bottom-right corner cell (m, n) from top left corner cell (0, 0). The only allowed directions to move from a cell are right or down or diagonally lower cell.

Revised Algorithm: Recursive solution to Minimum Cost Path Problem

```
int minCost(int cost[][], int m, int n)
   if (n < 0 | | m < 0)
      return INT_MAX_VALUE:
   else if (m == 0 \&\& n == 0)
      return cost[m, n];
   else
      return cost[m, n] + min( minCost(cost, m-1, n-1),
minCost(cost, m-1, n), minCost(cost, m, n-1));
Time complexity, T(n) =
```

Revised Algorithm: DP solution to Minimum Cost

```
int minCost( int cost[][], int n. int m
   int sol[n, m];
   int i, j;
   sol[0, 0] = cost[0, 0];
   for(int j=1; j < m; j++) {
      sol[0, j] = sol[0, j-1] + cost[0, j];
   for(int i=1; i < n; i++) {
      sol[i, 0] = sol[i-1, 0] + cost[i, 0];
   for (i=1; i<n; i++)
      for (j=1; j<m; j++)
         sol[i, j] = cost[i, j] + min(sol[i-1, j-1], sol[i-1, j],
sol[i, j-1])
   return sol[n, m];
   Time Complexity =
                                9/9
```