Dynamic Programming (5)

By: Aminul Islam

Minimum Cost Path Problem

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.

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

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));
}
```

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));
}
```

Time complexity, $T(n) =$

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));
}
```

Time complexity, $T(n) = \mathrm{T(n\text{-}1)+T(n\text{-}1)+1}$

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));
}
```

Time complexity, $T(n) =$ T(n-1)+T(n-1)+1
$T(n) \in O(2^n)$

Algorithm: DP solution to Minimum Cost Path Problem

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.)

Example of Minimum Cost Path Problem(DP sol.)

cost

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

Example of Minimum Cost Path Problem(DP sol.)

cost

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

sol

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[0, 0] = cost[0, 0];
```

sol

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[0, 0] = cost[0, 0];
```

sol

1			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int j=1; j < m; j++) {
   sol[0, j] = sol[0, j-1] + cost[0, j];
}
```

sol

1			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int j=1; j < m; j++) {
   sol[0, j] = sol[0, j-1] + cost[0, j];
}
```

sol

1	8		

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int j=1; j < m; j++) {
   sol[0, j] = sol[0, j-1] + cost[0, j];
}
```

sol

1	8	17	

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int j=1; j < m; j++) {
   sol[0, j] = sol[0, j-1] + cost[0, j];
}
```

sol

1	8	17	19

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int i=1; i < n; i++) {
   sol[i, 0] = sol[i-1, 0] + cost[i, 0];
}
```

sol

1	8	17	19

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int i=1; i < n; i++) {
   sol[i, 0] = sol[i-1, 0] + cost[i, 0];
}
```

sol

1	8	17	19
9			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int i=1; i < n; i++) {
   sol[i, 0] = sol[i-1, 0] + cost[i, 0];
}
```

sol

1	8	17	19
9			
10			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
for(int i=1; i < n; i++) {
   sol[i, 0] = sol[i-1, 0] + cost[i, 0];
}
```

sol

1	8	17	19
9			
10			
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9			
10			
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14		
10			
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	
10			
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10			
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16		
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16	23	
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12			

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21		

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
sol[i, j] = cost[i, j]+min(sol[i-1, j], sol[i, j-1])
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is:

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity =

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$\rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$\rightarrow 8 \rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $=O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$\rightarrow 2 \rightarrow 8 \rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

  $\rightarrow 3 \rightarrow 2 \rightarrow 8 \rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:
  $\rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 8 \rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $=O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$\rightarrow$ 7$\rightarrow$ 6$\rightarrow$ 3$\rightarrow$ 2$\rightarrow$ 8$\rightarrow$ 2

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $=O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$1 \rightarrow 7 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 8 \rightarrow 2$

Example of Minimum Cost Path Problem(DP sol.)

cost

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

```
return sol[n, m];
```

sol

1	8	17	19
9	14	17	19
10	16	23	27
12	21	29	29

- The cost of Minimum Cost Path is: 29
- Time Complexity $= O(mn)$ or $O(n^2)$
- The Minimum Cost Path (optimal solution) is:

$1 \rightarrow 7 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 8 \rightarrow 2$

- Can you write the algorithm to find out the optimal solution?

Exercise: Minimum Cost Path Problem

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

Exercise: Revised Minimum Cost Path Problem

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.

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.

cost

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

Revised Algorithm: Recursive solution to Minimum Cost Path Problem

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));
}
```

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: 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, $T(n) = \text{T(n-1)+T(n-1)+T(n-1)+1}$

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, $T(n) =$ T(n-1)+T(n-1)+T(n-1)+1
$T(n) = \frac{3^n - 1}{2}$ (Assuming $T(0) = 0$)

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, $T(n) =$ T(n-1)+T(n-1)+T(n-1)+1
$T(n) = \frac{3^n - 1}{2}$ (Assuming $T(0) = 0$)
$T(n) \in O(3^n)$

Revised Algorithm: DP solution to Minimum Cost Path Problem

Revised 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-1], sol[i-1, j],
sol[i, j-1])
       }
    }
    return sol[n, m];
}
```

Revised 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-1], sol[i-1, j],
sol[i, j-1])
       }
    }
    return sol[n, m];
}
```

• Time Complexity =

Revised 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-1], sol[i-1, j],
sol[i, j-1])
       }
    }
    return sol[n, m];
}
```

- Time Complexity $=O(mn)$ or $O(n^2)$