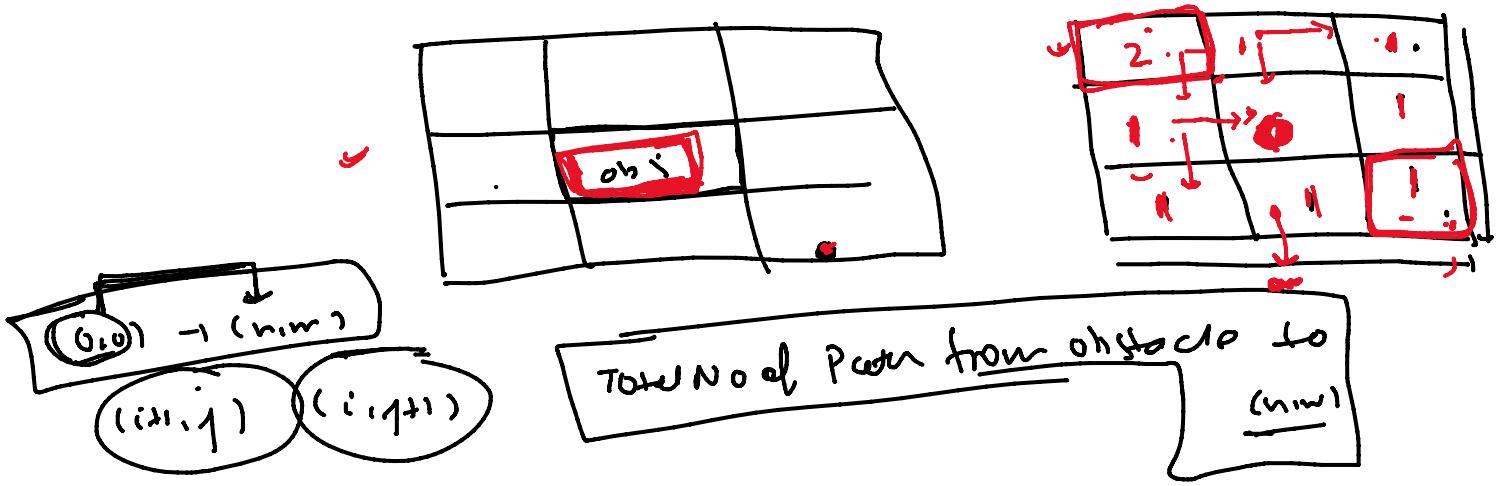
Class 87

17 May 2023 21:06

Class 87

Dynamic Programming

Unique Path 2



Vector < vector <int>> dp(n, vector<int> (m, 0));

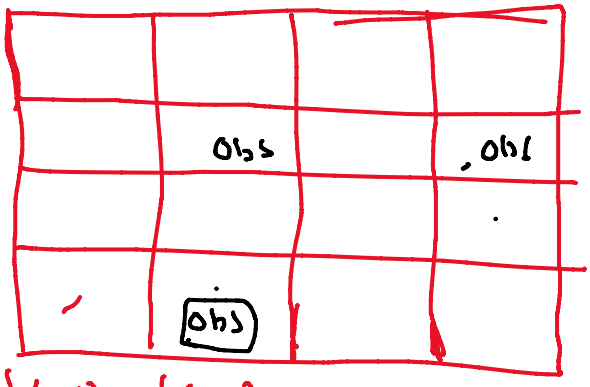
dp[n-1][m-1] = 1;

(0-0) → (n,m)

(n,m) → (n,m) = 1

```
for ( i = n-1; i >= 0; i-- )
{   for (j = m-1; j >= 0; j--)
    {   if (i==n-1 and j==m-1) continue
        if (grid[i][j] == 1)
        { dp[i][j] = 0; continue;
        }
        if ( i+1 < n)
        { dp[i][j] += dp[i+1][j];
        }
        if (j+1 < m)
```

```
if(j+1 < m)
    dp[i][j] += dp[i][j+1];
}
}
return dp[0][0];
```



(0,0) (0,1)

4	2	2	0
2	0		0
2	2	2	1
0	0	1	1

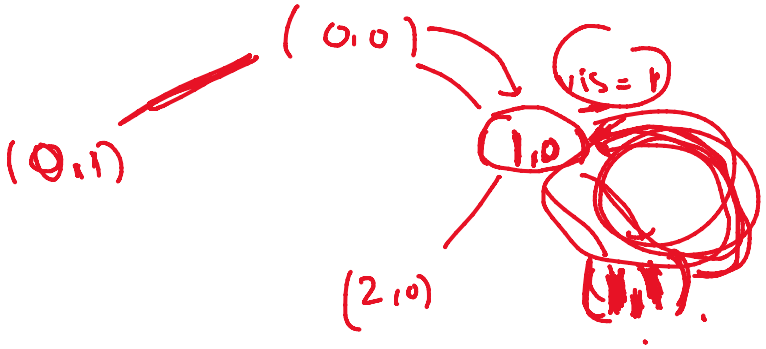
vis Array

DP

Backtracking

Backtracking DP

No cycle ✗



dp[0][1]

dp[1][0]

dp[1][2]

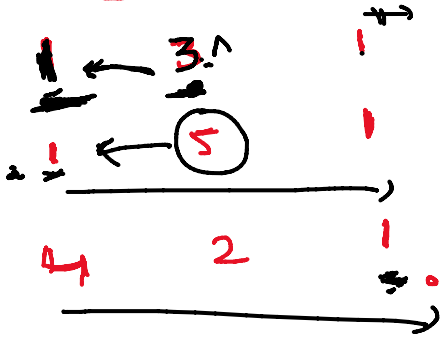
dp[2][1]

Min Path Sum

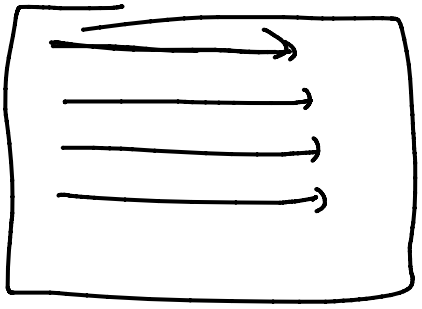
(0,0) → (0,0)

(0,0) (min)

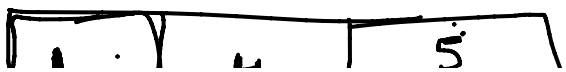
Recursion



vector < vector <int> > dp, vector<int> dp(INT_MAX));



```
For (i=0; i<n; i++)
{ For (j=0; j<n; j++)
{  if (i==0 and j==0)
      dp[i][j] = grid[i][j];

   else if ( i==0)
   {  dp[i][j] =  dp[i][j-1] + grid[i][j]
   }

   else if (j==0)
     dp[i][j] = dp[i-1][j] + grid[i][j];

   else
     dp[i][j] = min ( dp[i-1][j], dp[i][j-1] ) + grid[i][j];
}
}
return dp[n-1][n-1];
```

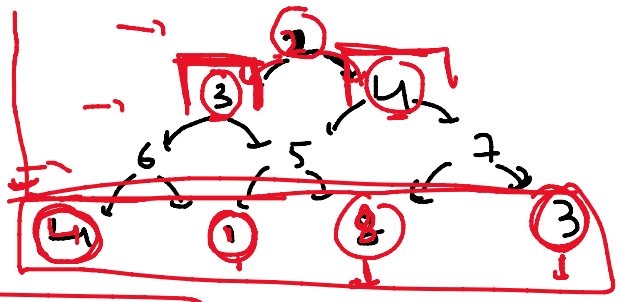


1	3	1
1	5	1
4	2	1

→

1	4	5
2	7	6
6	8	7

dp[i][j] = Min path sum to travel from (i,j) to (0,0)



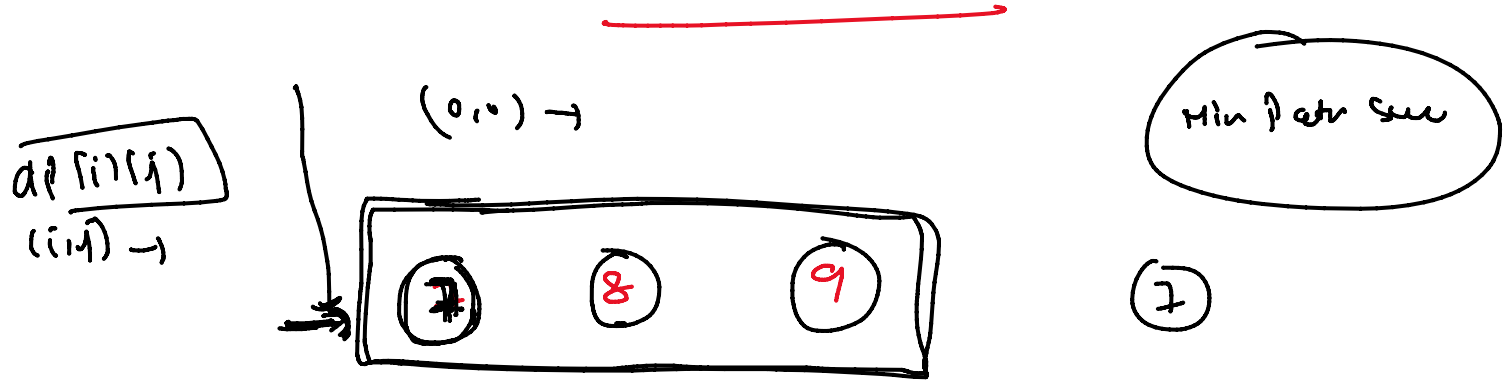
Min Path sum

0th row → nth row

```
Vector<vector<int>> dp(n+1);
for (i = 0; i<n; i++)
{ dp[i].resize (i+1);
}

For (i= 0; i< n; i++)
{ dp[n-1][i] = triangle[n-1][i];
}

For (i= n-2; i>= 0; i-- )
{ For (j=0; j<=i; j++)
```

```
    for (j=0; j<=i; j++)
    {
        dp[i][j] = min(dp[i+1][j], dp[i+1][j+1]) + triangle[i][j];
    }
}

return dp[0][0];
```

Min Path Fall Prog Sum



vector < vector <int> > dp (n, vector <int> (n, INT_MAX));

```
for (i=0; i<n; i++)
    dp[n-1][i] = grid[n-1][i];

for (i=n-2; i>=0; i--)
{
    for (j=0; j<n; j++)
    {
        dp[i][j] = grid[i][j] + dp[i+1][j];

        if (j>0)
            dp[i][j] = min(dp[i][j], dp[i+1][j-1] + grid[i][j]);
    }
}
```

left diagonally

```
dp[i][j] = min( dp[i][j], dp[i+1][j-1] + grid[i][j] );
if ( j<n-1 )
dp[i][j] = min ( dp[i][j], dp[i+1][j+1] + grid[i][j] )
}
```

MCM

```
for(i=0;i<n;i++)
        ans=min(ans,dp[0][i]);
    return ans;
```