Maximum path sum in matrix

Given a NxN matrix of positive integers. There are only three possible moves from a cell **Matrix[r][c]**.

1. Matrix [r+1] [c]
2. Matrix [r+1] [c-1]
3. Matrix [r+1] [c+1]

Starting from any column in row 0 return the largest sum of any of the paths up to row N-1.
Example 1:

```
Input: N = 2
Matrix = {{348, 391},
          {618, 193}}
Output: 1009
Explaination: The best path is 391 -> 618.
It gives the sum = 1009.
```

Example 2:

```
Input: N = 2
Matrix = {{2, 2},
          {2, 2}}
Output: 4
Explaination: No matter which path is
chosen, the output is 4.
```

Your Task:
You do not need to read input or print anything. Your task is to complete the function **maximumPath()** which takes the size N and the Matrix as input parameters and returns the highest maximum path sum.

Expected Time Complexity: O(N**N)**
Expected Auxiliary Space: O(NN)

```python
class Solution:
    def maximumPath(self, N, Matrix):
        # code here
        dp = [[0]*N for _ in range(N)]
        for j in range(N):
            dp[0][j] = Matrix[0][j]
        for i in range(1,N):
```

```
            for j in range(N):
                if j==0:
                    dp[i][j] = Matrix[i][j]+max(dp[i-1][j],dp[i-1][j+1])
                elif j==N-1:
                    dp[i][j] = Matrix[i][j]+max(dp[i-1][j-1],dp[i-1][j])
                else:
                    dp[i][j] = Matrix[i][j]+max(dp[i-1][j-1],dp[i-1]
[j],dp[i-1][j+1])
        return max(dp[-1])
```