Optimal strategy for a game In C++

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
    int arr[] = {20, 30, 2, 10};
    int n = sizeof(arr) / sizeof(arr[0]);

    int dp[n][n];  // Create a 2D array of size n x n

    for (int g = 0; g < n; g++) {
        for (int i = 0, j = g; j < n; i++, j++) {
            if (g == 0) {
                dp[i][j] = arr[i];
            } else if (g == 1) {
                dp[i][j] = max(arr[i], arr[j]);
            } else {
                int val1 = arr[i] + min((i + 2 <= j ? dp[i + 2][j] : 0), (i + 1 <= j - 1 ? dp[i + 1][j - 1] : 0));
                int val2 = arr[j] + min((i + 1 <= j - 1 ? dp[i + 1][j - 1] : 0), (i <= j - 2 ? dp[i][j - 2] : 0));

                dp[i][j] = max(val1, val2);
            }
        }
    }

    cout << dp[0][n - 1] << endl;  // Print the maximum value that can be collected

    return 0;
}
```

Dry Run with the Input arr[] = {20, 30, 2, 10}

We need to compute dp[0][n-1], which gives the result for the entire array.

Initialization:

- **dp table (initial values):**

```
dp[][] = {
    {0, 0, 0, 0},
    {0, 0, 0, 0},
    {0, 0, 0, 0},
    {0, 0, 0, 0}
}
```

Step-by-Step Iteration:

1. **g = 0** (Subarrays of size 1):
   - dp[0][0] = arr[0] = 20
   - dp[1][1] = arr[1] = 30
   - dp[2][2] = arr[2] = 2
   - dp[3][3] = arr[3] = 10
2. **g = 1** (Subarrays of size 2):
   - dp[0][1] = max(arr[0], arr[1]) = max(20, 30) = 30
   - dp[1][2] = max(arr[1], arr[2]) = max(30, 2) = 30
   - dp[2][3] = max(arr[2], arr[3]) = max(2, 10) = 10
3. **g = 2** (Subarrays of size 3):
   - For dp[0][2]: We compute two options:
     - val1 = arr[0] + min(dp[2][2], dp[1][1]) = 20 + min(2, 30) = 22
     - val2 = arr[2] + min(dp[1][1], dp[0][0]) = 2 + min(30, 20) = 22
     - dp[0][2] = max(22, 22) = 22
   - For dp[1][3]: We compute two options:
     - val1 = arr[1] + min(dp[3][3], dp[2][2]) = 30 + min(10, 2) = 32
     - val2 = arr[3] + min(dp[2][2], dp[1][1]) = 10 + min(2, 30) = 12
     - dp[1][3] = max(32, 12) = 32
4. **g = 3** (Subarrays of size 4):
   - For dp[0][3]: We compute two

	options:
	▪ val1 = arr[0] + min(dp[2][3], dp[1][2]) = 20 + min(10, 30) = 30
	▪ val2 = arr[3] + min(dp[1][2], dp[0][1]) = 10 + min(30, 30) = 40
	▪ dp[0][3] = max(30, 40) = 40
	
	Final DP Table:
	
	After all iterations, the final dp table looks like this:
	
	dp[][] = {
	{20, 30, 22, 40},
	{0, 30, 30, 32},
	{0, 0, 2, 10},
	{0, 0, 0, 10}
	}
	
	Result:
	
	The final value dp[0][3] = 40 is the maximum sum that can be collected by the first player in this game.

Output:- 40	