|  |  |
| --- | --- |
| **Optimal strategy for a game In C++** | |
| #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;  } | **Step-by-Step Dry Run with Table**  **Initialization**  Given input:  int arr[] = {20, 30, 2, 10};  Size of arr:  n = 4;  A **2D DP table (dp[i][j])** is used, where dp[i][j] represents the **maximum score the first player can collect from arr[i] to arr[j]**.  **Step 1: Fill Diagonal (g = 0)**  When i == j, only one element is available, so:   | **i** | **j** | **dp[i][j]** | | --- | --- | --- | | 0 | 0 | 20 | | 1 | 1 | 30 | | 2 | 2 | 2 | | 3 | 3 | 10 |   **Step 2: Fill g = 1 (Two Elements)**  When g = 1, two elements are available, so the first player picks the maximum:   | **i** | **j** | **Computation** | **dp[i][j]** | | --- | --- | --- | --- | | 0 | 1 | max(20, 30) | 30 | | 1 | 2 | max(30, 2) | 30 | | 2 | 3 | max(2, 10) | 10 |   **Step 3: Fill g = 2 (Three Elements)**  Now, we consider **three elements** and the optimal choices:   | **i** | **j** | **Computation** | **dp[i][j]** | | --- | --- | --- | --- | | 0 | 2 | max(20 + min(2, 30), 2 + min(30, 20)) → max(20+2, 2+20) = **22** | 22 | | 1 | 3 | max(30 + min(10, 2), 10 + min(2, 30)) → max(30+2, 10+2) = **32** | 32 |   Step 4: Fill g = 3 (Entire Array)   | **i\j** | **0** | **1** | **2** | **3** | | --- | --- | --- | --- | --- | | **0** | 20 | 30 | 22 | **40** | | **1** |  | 30 | 30 | 32 | | **2** |  |  | 2 | 10 | | **3** |  |  |  | 10 |   **Final Output:**  40 |
| Output:- 40 | |