```cpp
//
//  Taro and Jiro will play the following game against each other.
//  Initially, they are given a sequence a = (a[1], a[2], … , a[n])
//  Until a becomes empty, the two players perform the following operation alternately,
//  starting from Taro:
//    - Remove the element at the beginning or the end of a.
//      The player earns x points, where x is the removed element.
//
//  Let X and Y be Taro's and Jiro's total score at the end of the game, respectively.
//  Taro tries to maximize X-Y, while Jiro tries to minimize X-Y.
//  Assuming that the two players play optimally, find the resulting value of X-Y.
//
//     Time Complexity: O(N^2)
//

#include <bits/stdc++.h>
#define ll long long

using namespace std;

int main() {
    int n;
    cin >> n;

    vector<ll> awards(n);
    vector<vector<ll>> dp(n, vector<ll>(n, 0));
    for (int i = 0; i < n; i++) {
        cin >> awards[i];

        // BASE CASE:
        dp[i][i] = awards[i];
    }

    // interval starting and ending at position i = score is awards[i]
    // solving for subintervals

    ///  _ _ _ _ _ _ _ _
    ///    ⟷
    ///  ⟷
    ///    ⟷

    for (int size = 1; size < n; size++) { // size of interval
        for (int start = 0; start < n - size; start++) { // starting position of interval
            if (size % 2 == 0) { // Taro (max)
                //  START - L
                //  START+SIZE = R
                dp[start][start+size] =
                    max(dp[start][start+size-1] + awards[start+size],
                        dp[start+1][start+size] + awards[start]);
            } else { // Jiro (min)
                dp[start][start+size] =
                    min(dp[start][start+size-1] - awards[start+size],
                        dp[start+1][start+size] - awards[start]);
            }
        }
    }

    // dp[0][n-1]
    cout << (n % 2 == 0 ? -1LL : 1LL) * dp[0][n-1] << endl;
    return 0;
}
```