

Problem statement: Consider a row of n coins of values $v_1 \dots v_n$, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

Note: The opponent is as clever as the user.

Let us understand the problem with few examples:

1. 5, 3, 7, 10 : The user collects maximum value as $15(10 + 5)$

2. 8, 15, 3, 7 : The user collects maximum value as $22(7 + 15)$

Does choosing the best at each move give an optimal solution?

No. In the second example, this is how the game can finish:

1.

.....User chooses 8.

.....Opponent chooses 15.

.....User chooses 7.

.....Opponent chooses 3.

Total value collected by user is $15(8 + 7)$

2.

.....User chooses 7.

.....Opponent chooses 8.

.....User chooses 15.

.....Opponent chooses 3.

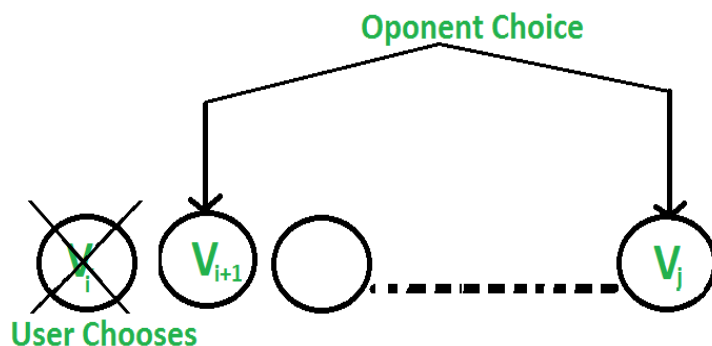
Total value collected by user is $22(7 + 15)$

So if the user follows the second game state, maximum value can be collected although the first move is not the best.

There are two choices:

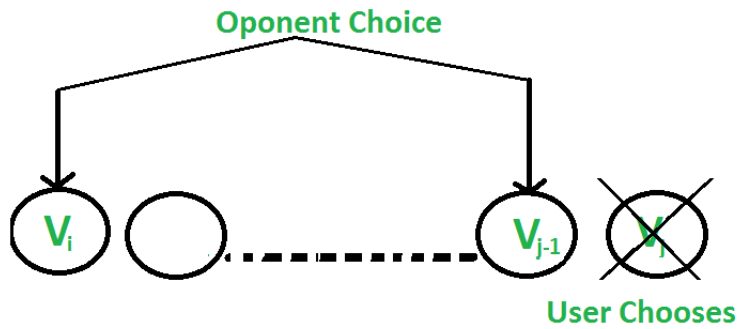
1. The user chooses the i th coin with value V_i : The opponent either chooses $(i+1)$ th coin or j th coin. The opponent intends to choose the coin which leaves the user with minimum value.

i.e. The user can collect the value $V_i + \min(F(i+2, j), F(i+1, j-1))$



2. The user chooses the j th coin with value V_j : The opponent either chooses i th coin or $(j-1)$ th coin. The opponent intends to choose the coin which leaves the user with minimum value.

i.e. The user can collect the value $V_j + \min(F(i+1, j-1), F(i, j-2))$



Following is recursive solution that is based on above two choices. We take the maximum of two choices.

$F(i, j)$ represents the maximum value the user can collect from i 'th coin to j 'th coin.

$$F(i, j) = \text{Max}(V_i + \min(F(i+2, j), F(i+1, j-1)), V_j + \min(F(i+1, j-1), F(i, j-2)))$$

Base Cases

$$F(i, j) = V_i \quad \text{If } j == i$$

$$F(i, j) = \max(V_i, V_j) \quad \text{If } j == i+1$$

Why Dynamic Programming?

The above relation exhibits overlapping sub-problems. In the above relation, $F(i+1, j-1)$ is calculated twice.

// C program to find out maximum value from a given sequence of coins

```
#include <stdio.h>
#include <limits.h>
```

// Utility functions to get maximum and minimum of two integers

```
int max(int a, int b) { return a > b ? a : b; }
int min(int a, int b) { return a < b ? a : b; }
```

// Returns optimal value possible that a player can collect from
// an array of coins of size n. Note that n must be even

```
int optimalStrategyOfGame(int* arr, int n)
{
    // Create a table to store solutions of subproblems
    int table[n][n], gap, i, j, x, y, z;

    // Fill table using above recursive formula. Note that the table
    // is filled in diagonal fashion (similar to http://goo.gl/PQqoS),
    // from diagonal elements to table[0][n-1] which is the result.
    for (gap = 0; gap < n; ++gap)
    {
        for (i = 0, j = gap; j < n; ++i, ++j)
        {
            // Here x is value of F(i+2, j), y is F(i+1, j-1) and
            // z is F(i, j-2) in above recursive formula
            x = ((i+2) <= j) ? table[i+2][j] : 0;
            y = ((i+1) <= (j-1)) ? table[i+1][j-1] : 0;
            z = (i <= (j-2)) ? table[i][j-2] : 0;

            table[i][j] = max(arr[i] + min(x, y), arr[j] + min(y, z));
        }
    }

    return table[0][n-1];
}
```

// Driver program to test above function

```
int main()
{
```

```
int arr1[] = {8, 15, 3, 7};
int n = sizeof(arr1)/sizeof(arr1[0]);
printf("%d\n", optimalStrategyOfGame(arr1, n));

int arr2[] = {2, 2, 2, 2};
n = sizeof(arr2)/sizeof(arr2[0]);
printf("%d\n", optimalStrategyOfGame(arr2, n));

int arr3[] = {20, 30, 2, 2, 2, 10};
n = sizeof(arr3)/sizeof(arr3[0]);
printf("%d\n", optimalStrategyOfGame(arr3, n));

return 0;
}
```

Output:

```
22
4
42
```

Exercise

Your thoughts on the strategy when the user wishes to only win instead of winning with the maximum value. Like above problem, number of coins is even.

Can Greedy approach work quite well and give an optimal solution? Will your answer change if number of coins is odd?