Codility_

CodeCheck Report: trainingP2NYP2-C2Q

Test Name:

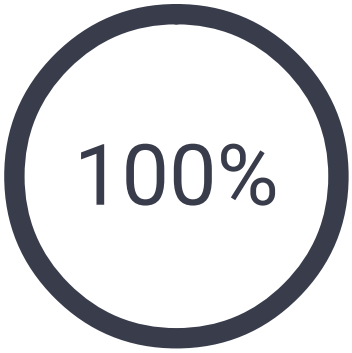
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
NumberSolitaire C++	1 min	100%

Total score



Tasks Details

Medium

1. **NumberSolitaire**

In a given array, find the subset of maximal sum in which the distance between consecutive elements is at most 6.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A game for one player is played on a board consisting of N consecutive squares, numbered from 0 to N − 1. There is a number written on each square. A non-empty array A of N integers contains the numbers written on the squares. Moreover, some squares can be marked during the game.

At the beginning of the game, there is a pebble on square number 0 and this is the only square on the board which is marked. The goal of the game is to move the pebble to square number N − 1.

During each turn we throw a six-sided die, with numbers from 1 to 6 on its faces, and consider the number K, which shows on the upper face after the die comes to rest. Then we move the pebble standing on square number I to square number I + K, providing that square number I + K exists. If square number I + K does not exist, we throw the die again until we obtain a valid move. Finally, we mark square number I + K.

After the game finishes (when the pebble is standing on square number N − 1), we calculate the result. The result of the game is the sum of the numbers written on all marked squares.

For example, given the following array:

Solution

Programming language used:	C++
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

14:40:02 14:40:39

Code: 14:40:39 UTC, cpp, final, score: **100** show code in pop-up

```
A[0] = 1
A[1] = −2
A[2] = 0
A[3] = 9
A[4] = −1
A[5] = −2
```

one possible game could be as follows:

- the pebble is on square number 0, which is marked;
- we throw 3; the pebble moves from square number 0 to square number 3; we mark square number 3;
- we throw 5; the pebble does not move, since there is no square number 8 on the board;
- we throw 2; the pebble moves to square number 5; we mark this square and the game ends.

The marked squares are 0, 3 and 5, so the result of the game is 1 + 9 + (−2) = 8. This is the maximal possible result that can be achieved on this board.

Write a function:

```
int solution(vector<int> &A);
```

that, given a non-empty array A of N integers, returns the maximal result that can be achieved on the board represented by array A.

For example, given the array

```
A[0] = 1
A[1] = −2
A[2] = 0
A[3] = 9
A[4] = −1
A[5] = −2
```

the function should return 8, as explained above.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [2..100,000];
- each element of array A is an integer within the range [−10,000..10,000].


```cpp
// you can use includes, for example:
// #include <algorithm>
#include <climits>
// you can write to stdout for debugging purpo
// cout << "this is a debug message" << endl;

int solution(vector<int> &A) {
    int N = A.size();

    vector<int> dp(N, INT_MIN);

    dp[0] = A[0];

    for (int i = 1; i < N; i++) {
        for (int dice = 1; dice <= 6; dice++)
            if (i - dice >= 0) {
                dp[i] = max(dp[i], dp[i-dice]+
            }
        }

    }

    // for (auto e : dp) {
    //     cout << e << " ";
    // }
    // cout << endl;

    return dp[N-1];
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N)**

expand all	Example tests	
▶	example example test	✔ OK
expand all	Correctness tests	
▶	extreme two or three fields	✔ OK
▶	simple simple test	✔ OK
▶	medium_all_negative all values negative, length = ~1,000	✔ OK
▶	medium_monotonic monotonic sequence, length = ~1,000	✔ OK
▶	medium_random random sequence of values, length = ~1,000	✔ OK
expand all	Performance tests	
▶	big_all_negative all values negative, length = ~100,000	✔ OK
▶	big_random random sequence of values, length = ~100,000	✔ OK
▶	extreme_answers maximal and minimal answers	✔ OK