

CodeCheck Report: trainingZZSXXG-6J3

[Check out Codility training tasks](#)

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
NumberSolitaire Java 8	30 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 $I + K$.

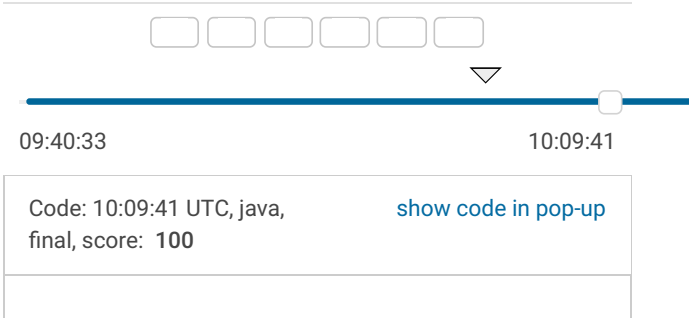
After the game finishes (when the pebble is standing on square number $N - 1$), we calculate the result. The result of the game is the sum of the numbers written on all marked squares.

For example, given the following array:

Solution

Programming language used:	Java 8	
Total time used:	30 minutes	
Effective time used:	30 minutes	
Notes:	not defined yet	

Task timeline



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:

```
class Solution { public int solution(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].

Copyright 2009–2023 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
1 // you can also use imports, for example:
2 import java.util.*;
3
4 // you can write to stdout for debugging purposes
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] A) {
9         int N = A.length;
10
11         int[] dp = new int[N];
12         dp[0] = A[0];
13
14         for (int i = 1; i < N; i++) {
15             int maxPrevious = Arra
16             dp[i] = maxPrevious +
17         }
18         return dp[N - 1];
19     }
20 }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

collapse all		Example tests
▼	example	✓ OK
example test		
1. 0.008 s		OK
collapse all		Correctness tests
▼	extreme	✓ OK
two or three fields		
1. 0.012 s		OK
2. 0.008 s		OK
3. 0.008 s		OK
4. 0.008 s		OK
▼	simple	✓ OK
simple test		
1. 0.008 s		OK
2. 0.008 s		OK
3. 0.008 s		OK
▼	medium_all_negative	✓ OK
all values negative, length = ~1,000		
1. 0.020 s		OK
▼	medium_monotonic	✓ OK
monotonic sequence, length = ~1,000		
1. 0.016 s		OK
▼		

medium_random ✓ OK
random sequence of values, length =
~1,000

1. 0.016 s OK

collapse all Performance tests

▼ big_all_negative ✓ OK
all values negative, length = ~100,000

1. 0.876 s OK

▼ big_random ✓ OK
random sequence of values, length =
~100,000

1. 0.892 s OK

▼ extreme_answers ✓ OK
maximal and minimal answers

1. 0.920 s OK

2. 0.928 s OK