**Max Array Sum**

Given an array of integers, find the subset of non-adjacent elements with the maximum sum. Calculate the sum of that subset.

For example, given an array=[-2,1,3,-4,5]  we have the following possible subsets:

Subset Sum

[-2, 3, 5] 6

[-2, 3] 1

[-2, -4] -6

[-2, 5] 3

[1, -4] -3

[1, 5] 6

[3, 5] 8

Our maximum subset sum is 8 .

**Function Description**

Complete the maxSubsetSum function in the editor below. It should return an integer

representing the maximum subset sum for the given array.

maxSubsetSum has the following parameter(s):

* *arr*: an array of integers

**Input Format**

The first line contains an integer, n .   
The second line contains  n space-separated integers arr[i].

**Constraints**

* 1<=n<=10^5
* -10^4<=arr[i]<=10^4

**Output Format**

Return the maximum sum described in the statement.

**Sample Input 0**

5

3 7 4 6 5

**Sample Output 0**

13

**Explanation 0**

Our possible subsets are [3,4,5],[3,4],[3,6],[3,5],[7,6],[7,5] and [4,5]. The largest subset sum

is 13  from subset  [7,6]

**Sample Input 1**

5

2 1 5 8 4

**Sample Output 1**

11

**Explanation 1**

Our subsets are [2,5,4],[2,5],[2,8],[2,4],[1,8],[1,4] and [5,4]. The maximum subset sum is 11  from the first subset listed.

**Sample Input 2**

5

3 5 -7 8 10

**Sample Output 2**

15

**Explanation 2**

Our subsets are [3,-7,10],[3,-7][3,8],[3,10],[5,8],[5,10],  and [-7,10] . The maximum subset sum is  15 from the sixth subset listed.

using System.CodeDom.Compiler;

using System.Collections.Generic;

using System.Collections;

using System.ComponentModel;

using System.Diagnostics.CodeAnalysis;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Runtime.Serialization;

using System.Text.RegularExpressions;

using System.Text;

using System;

class Solution {

static int maxSubsetSum(int[] a) {

int[] dp = new int[a.Length];

dp[0]=a[0];

if(dp[0]<0){

dp[0]=0;

}

dp[1] = a[0]>a[1]?a[0]:a[1];

if(dp[1]<0){

dp[1]=0;

}

for(int i=2;i<a.Length;i++){

dp[i] = Math.Max(a[i]+dp[i-2], dp[i-1]);

}

return dp[a.Length-1];

}

static void Main(string[] args) {

TextWriter textWriter = new StreamWriter(@System.Environment.GetEnvironmentVariable("OUTPUT\_PATH"), true);

int n = Convert.ToInt32(Console.ReadLine());

int[] arr = Array.ConvertAll(Console.ReadLine().Split(' '), arrTemp => Convert.ToInt32(arrTemp))

;

int res = maxSubsetSum(arr);

textWriter.WriteLine(res);

textWriter.Flush();

textWriter.Close();

}

}

**Congratulations**

You solved this challenge. Would you like to challenge your friends?

[Next Challenge](https://www.hackerrank.com/challenges/abbr?h_l=interview&playlist_slugs%5B%5D=interview-preparation-kit&playlist_slugs%5B%5D=dynamic-programming&h_r=next-challenge&h_v=zen)

* **Test case 0**
* **Test case 1**
* **Test case 2**
* **Test case 3**
* **Test case 4**
* **Test case 5**
* **Test case 6**
* **Test case 7**
* **Test case 8**
* **Test case 9**
* **Test case 10**
* **Test case 11**
* **Test case 12**
* **Test case 13**
* **Test case 14**
* **Test case 15**
* **Test case 16**
* **Test case 17**
* **Test case 18**
* **Test case 19**
* **Test case 20**
* **Test case 21**
* **Test case 22**
* **Test case 23**
* **Test case 24**
* **Test case 25**
* **Test case 26**
* **Test case 27**
* **Test case 28**
* **Test case 29**
* **Test case 30**
* **Test case 31**
* **Test case 32**

Compiler Message

**Success**

Hidden Test Case

Unlock this testcase for 5 hackos.

I follow combination

public class Combinations

{

private List<int> output = new List<int>();

private int[] inputstring;

private int sum=0;

public Combinations(int[] str )

{

inputstring = str;

Console.WriteLine("The input string is : " + inputstring);

}

public static void Main()

{

int[] arr = new int[] { 3, 5, -7, 8, 10 };

Combinations combobj = new Combinations(arr);

Console.WriteLine("All possible combinations are : ");

combobj.combine();

}

public void combine() { combine(0); }

private void combine(int start)

{

for (int i = start; i < inputstring.Length; ++i)

{

output.Add(inputstring[i]);

sum += sum + inputstring[i];

Console.WriteLine(sum);

if (i < inputstring.Length)

{

combine(i + 2);

}

output.Capacity=output.Capacity-1;

}

}

}

Code passed but test doesn’t pass time out.