**Split array in two arrays with the same average**

1. **Problem description**

In a given integer array A, we must move every element of A to either list B or list C. B and C are initially empty. We must return true if and only if after such a move, it is possible that the average value of B is equal to the average of C, and B and C are both non-empty.

* 1. **Example**

Sample input: [1,2,3,4,5,6,7,8]

Sample output: True

The array can be split in [1,4,5,8] and [2,3,6,7], both with an average of 4.5.

1. **Problem analysis**

We will look at the problem from a general point of view. We have an array of length n. The sum of all its’s elements will be s. If we split it in two arrays (A, B) of length respective , and the sum of all their elements and . We can write the condition of the problem as follows:

We will keep the values related to A and will write the as the number of integers that remain in array after we move elements, and as the sum of the al these elements. After this change the equation will look like this:

(1)

(2)

Starting from the condition of the problem we observe from (2) that the average of both A and B are equal only if they have the same average as the array we start from.

In equation (1), represents all the possible lengths of A from 1 to n-1. If that is the case, we can check if there is possible a partition of and n - so that both have the same average. The interval in which situates in is [1, n/2]. Using the equation, we can decide if it is worth to investigate more or just return false because the array can’t be split in two arrays with the same average.

If it turns out it is possible, we will need to check if at last one of the subarrays has the same average as the array we start from (). For each that passed the first check we will verify if the average of the subarray is equal to for each sum we can obtain combining elements from the array.

1. **Implementation**
   1. **Environment and programming language**

The environment used for implementation is Visual Studio 2019. The code is

written in C++.

* 1. **Classes and methods**

For the shake of reusability, we will create a class Verifier with a public method

verify (). This way we can easy call the method from different parts of the same project, or we can import it in other projects. The class will have three local variable a vector and two integer values that will retain the array we will verify, length of the array and the sum of all the integers in the vector. This way we can concentrate on the solution from verify method.

Text

Description automatically generated

Let’s look at the code from above. In the first for loop, we check if the right part of the equation (1) is an integer for each i / from 1 to n/2. We know the array contains only integers, that means that is an integer too. We verify only until n/2, because if there is a i(an array of length i) that satisfy the condition ,there is also an array of length n-i with the same qualities. All the lengths that satisfied the first check will be saved in a vector. Next, we will dynamically generate all the sums for each possible length of a partition (1 to n/2). To store all the sums that are generated we will use a vector of sets.

So that the first set from the vector will contain all the possible sums of an empty array (0). The second set will contain all the possible sums of an array with one element and so on. We will add the sums in a set as follows: for each integer from the array, we will add from the last set to the second set, all the sums of the integer with every element from the set above. Example:

Array: [1,2,3], the length of the array is 4, the vector of the sums will have 3 sets. X takes by turn every value from the array.

vector<set<integer>> sums = {{0}, {}, {}}

x=1 => sums = {{0},{1}, {}}.

x=2=> sums = {{0}, {1,2},{3}}

x=3=> sums = {{0}, {1,2,3}, {3,4,5}}

The last part of the algorithm is to check for each index we saved if there is a sum that divided to it will have the same value as the average of the array. If we find one, we will return true else false.

1. **Testing**

To ensure the reliability of verify () we need to test the correctness

of the outputs we receive after calling the method. In this scope it is necessary to create a test library we can use to create an array for a true or false case and compare the expected values to the output. The logic used in creating an array we can split is that we will create two separate arrays and change them to fit some the requirements. Both lengths of the arrays are chosen random. While the length of the second array multiplied with the sum of the first subarray divided by the length of the first array isn’t an integer, we will decrement the length of the second array. In case the test isn’t working it is possible that the length of the second array to be negative since there is no stopping condition. Next, we will have to fix the sum of the elements from the second subarray so that it is equal to the second array multiplied with the sum of the first subarray divided by the length of the first array. We will decrement /add the difference between the two values to elements in the second array without exceeding the limit values. The user can uncomment the call of the function test () from main. The function will compare values received from the verify () and the expected values from the array that was created.

1. **Conclusions**

In the end, after analyzing the solution of the problem, we identify two essentials

steps of the algorithm: the first check that rules out the arrays we can’t split, and second

in which we dynamically generate all the possible sums of an array, after which we’ll test if there is at last one that produce the same average as the initial array. The complexity is O(n) in best case scenario, or O(n^3) in worst case scenario.

1. **Bibliography**

* https://www.youtube.com/watch?v=FBQbm26tSzA
* https://leetcode.com/problems/split-array-with-same-average/ (for testing)