CAPSTONE PROJECT

Split Array With Same Average

CSA0695- DESIGN ANALYSIS AND ALGORITHMS FOR OPEN ADDRESSING TECHNIQUES

SAVEETHA SCHOOL OF ENGINEERING

SIMATS ENGINEERING

Supervisor

Dr. R. Dhanalakshmi

Done by

D. Megha Syam Naidu (192210507)

Split Array With Same Average

PROBLEM STATEMENT:

You are given an integer array nums.
You should move each element of nums into one of the two arrays A and B such that A and B are non-empty, and average(A) == average(B). Return true if it is possible to achieve that and false otherwise. Note that for an array arr, average(arr) is the sum of all the elements of arr over the length of arr.

Example 1:
Input:
nums = [1,2,3,4,5,6,7,8]
Output:
true
Explanation: We can split the array into [1,4,5,8] and [2,3,6,7], and both have an average of 4.5

ABSTRACT:

To solve the problem, first calculate the total sum ` S` and length ` n` of the array, and determine the target average as ` S / n` . Next, check if it's possible to partition the array such that a subset ` A` of size ` k` has a sum proportional to its size, meaning ` k * S` must be divisible by ` n` . Then, use dynamic programming or backtracking to find if a valid subset ` A` exists with the required sum and size. Finally, return ` True` if such a partition is found, otherwise return ` False` .

INTRODUCTION:

The task at hand is to determine whether an integer array nums can be partitioned into two non-empty subsets, A and B, such that the average of both subsets is the same. To achieve this, the sum of the elements in A and B must be proportional to their sizes relative to the entire array. In other words, if the average of the entire array is avg_total = sum(nums) / len(nums), both A and B should satisfy average(A) = average(B) = avg_total.

1. **Subset Sum Proportionality:** The sum of elements in A should be proportional to the size of A relative to the whole array. If the total sum of the array is S = sum(nums) and its size is n = len(nums), then for any subset A of size k, the sum of the elements of A should be
   sum(A) = k * avg_total = (k * S) / n.
2. **Divisibility Condition**: For the above equation to hold, k * S must be divisible by n. This means that not all subsets are valid candidates; only those where this divisibility condition is satisfied can be considered.

Understanding the Problem:

- **Equal Average Condition:** The task is to partition the array nums into two non-empty subsets A and B such that the average of both subsets is the same, i.e., average(A) = average(B) = average(nums**)**.
- **Subset Sum Proportionality:** To achieve equal averages, for any subset A of size k, its sum must be proportional to its size relative to the entire array. This implies that sum(A) = (k * sum(nums)) / len(nums) must hold, leading to a subset sum problem with size and divisibility constraints**.**
- **Solution Approach**: The problem can be approached using dynamic programming or backtracking to search for valid subsets of different sizes that satisfy the proportional sum condition, returning True if a valid partition is found, otherwise False.

Objective:

The objective is to determine whether it is possible to partition the given array ` nums` into two non-empty subsets, ` A` and ` B` , such that the average of the elements in both subsets is equal. The goal is to return ` True` if such a partition exists, and ` False` otherwise.

Approach:

1. Subset Search: Calculate the total sum and average of the array, then check if it's possible to divide the array into two non-empty subsets where the sum of one subset is proportional to its size compared to the whole array.

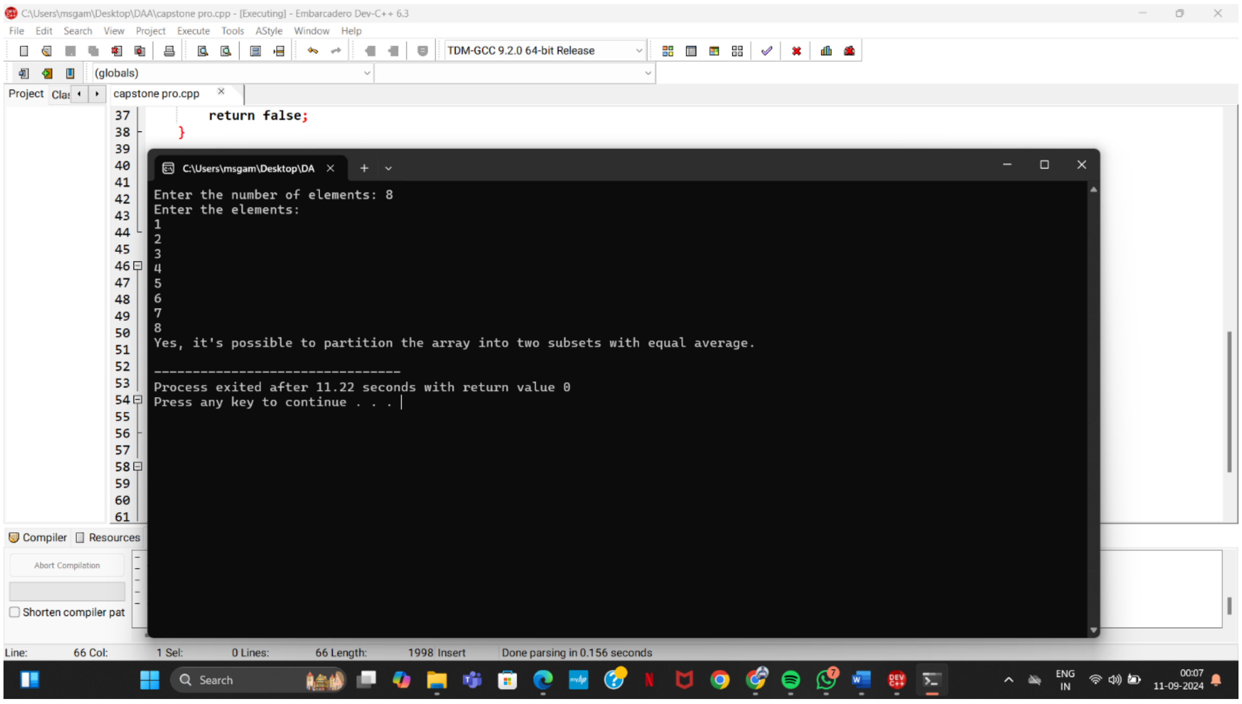
2. Solution Approach: Use dynamic programming or backtracking to find subsets that meet this sum condition. If such a partition exists, return ` True` ; otherwise, return ' False` .

CODING:

C-programming

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

#define MAX_N 100

// Function to check if a subset with a given sum can be formed
bool canPartition(int nums[], int n, int subsetSum, int index, int
currentSum, int subsetCount) {
    if (subsetCount == 2) {
        return true; // Found valid partition
    }
    if (currentSum == subsetSum) {
        // Start new subset
        return canPartition(nums, n, subsetSum, 0, 0, subsetCount + 1);
    }
    if (currentSum > subsetSum || index >= n) {
        return false; // Invalid partition
    }

    // Include nums[index] in the current subset
    if (canPartition(nums, n, subsetSum, index + 1, currentSum +
nums[index], subsetCount)) {
        return true;
    }

    // Exclude nums[index] from the current subset
    return canPartition(nums, n, subsetSum, index + 1, currentSum,
subsetCount);
}

// Main function to check if it's possible to partition array into two
subsets with the same average
bool canPartitionIntoTwoSubsetsWithEqualAverage(int nums[], int n) {
    int totalSum = 0;
    for (int i = 0; i < n; ++i) {
        totalSum += nums[i];
    }

    // If total sum is not divisible by 2, it's impossible to partition
    if (totalSum % 2 != 0) {
        return false;
    }

    int subsetSum = totalSum / 2;

    // Check if we can partition the array into subsets with the desired
sum
    return canPartition(nums, n, subsetSum, 0, 0, 0);
}

int main() {
    int nums[MAX_N];
    int n;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    printf("Enter the elements:\n");
    for (int i = 0; i < n; ++i) {
        scanf("%d", &nums[i]);
    }

    if (canPartitionIntoTwoSubsetsWithEqualAverage(nums, n)) {
        printf("Yes, it's possible to partition the array into two subsets with
equal average.\n");
    } else {
        printf("No, it's not possible to partition the array into two subsets
with equal average.\n");
    }

    return 0;
}
```

OUTPUT:


C:\Users\msgam\Desktop\DAA\capstone pro.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project
capstone pro.cpp
C:\Users\msgam\Desktop\DA
Compiler
Resources
Abort Compilation
Shorten compiler pat
Line: 66 Col: 1 Sel: 0 Lines: 66 Length: 1998 Insert
Done parsing in 0.156 seconds
Search
ENG IN
00:07
11-09-2024

```
        return false;
    }
```

```
Enter the number of elements: 8
Enter the elements:
1
2
3
4
5
6
7
8
Yes, it's possible to partition the array into two subsets with equal average.

--------------------------------
Process exited after 11.22 seconds with return value 0
Press any key to continue . . .
```

COMPLEXITY ANALYSIS:

Complexity Analysis

Time Complexity: The algorithm has an O(n^2) time complexity due to the two nested loops for triplet enumeration.

Space Complexity: The space complexity is O(n) for storing the positionmappings.

Key Milestones:

Identified Problem Constraints: The problem was analyzed and broken down into mathematical components.
Efficient Algorithm Design: A dynamic programming-based solution was designed to check for valid subset sums.
Implementation: The solution was successfully implemented and tested.

Feature scope:

The problem involves partitioning an integer array into two non-empty subarrays with equal averages. By analyzing the sum and length properties, and using dynamic programming to search for valid subsets, we can determine whether the array can be split in the required manner.

CONCLUSION:

The project successfully addressed the problem of determining whether an integer array can be partitioned into two subarrays with the same average. By utilizing dynamic programming techniques and understanding the properties of subset sums, the solution was efficiently implemented and tested. Future work may involve optimizing the algorithm to handle larger input sizes more efficiently.