Computative programming

Assignment 3-5

Hallticket no:2303A510A6

Batch no:02

Assignment -1 Subset Sum Using Meet-in-the-Middle

Problem Statement:

You are given an array of N integers and a target sum S.

Using the Meet-in-the-Middle technique, determine whether there exists a subset whose sum is exactly equal to S.

Input Format:

- **The first line contains an integer T, the number of test cases.**

For each test case:

- **The first line contains two integers N and S.**
- **The second line contains N integers.**

Output Format:

For each test case, print YES if such a subset exists, otherwise print NO.

Constraints:

- **1 ≤ T ≤ 20**
- **1 ≤ N ≤ 40**
- **-10^9 ≤ A[i] ≤ 10^9**

Sample input

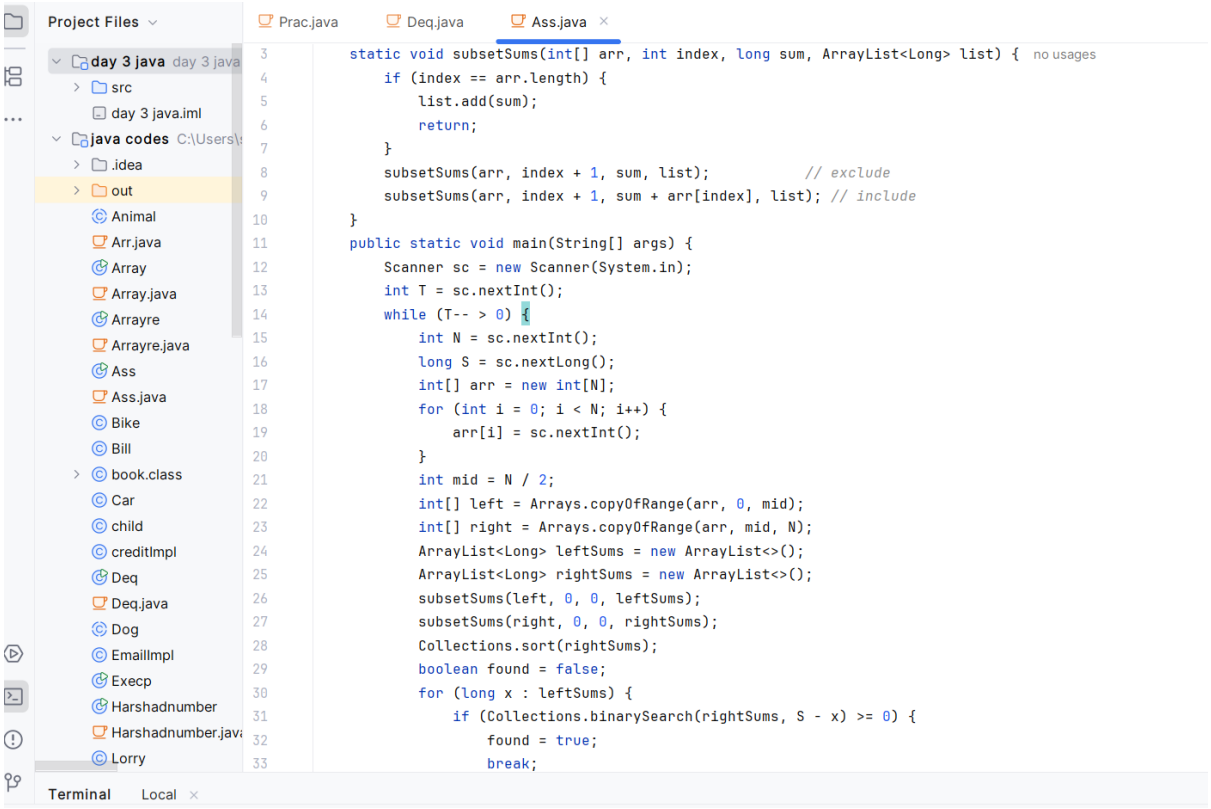
```
2
4 9
3 1 5 7
5 10
2 4 6 8 1
```

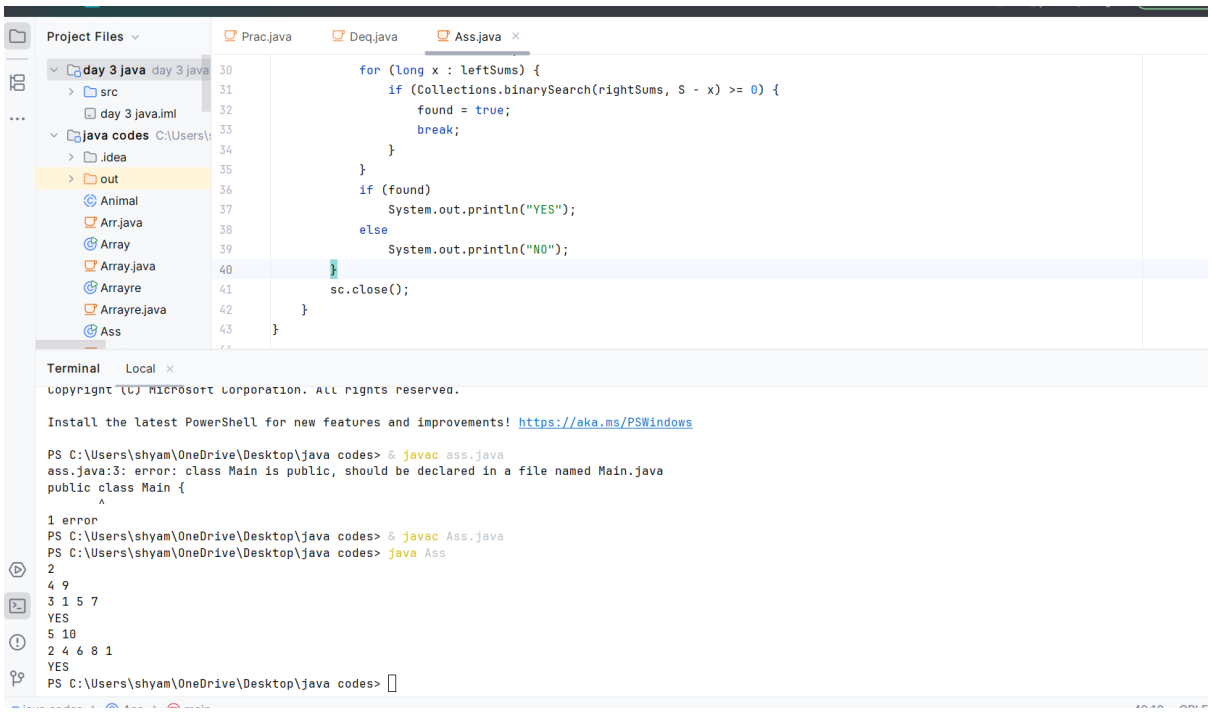
Sample output

```
YES
NO
```

CODE (java):

```
import java.util.*;
public class Ass {
   static void subsetSums(int[] arr, int index, long sum, ArrayList<Long> list) {
      if (index == arr.length) {
         list.add(sum);
         return;
      }
      subsetSums(arr, index + 1, sum, list);          // exclude
      subsetSums(arr, index + 1, sum + arr[index], list); // include
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int T = sc.nextInt();
      while (T-- > 0) {
         int N = sc.nextInt();
         long S = sc.nextLong();
         int[] arr = new int[N];
         for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
         }
         int mid = N / 2;
         int[] left = Arrays.copyOfRange(arr, 0, mid);
         int[] right = Arrays.copyOfRange(arr, mid, N);
         ArrayList<Long> leftSums = new ArrayList<>();
         ArrayList<Long> rightSums = new ArrayList<>();
         subsetSums(left, 0, 0, leftSums);
         subsetSums(right, 0, 0, rightSums);
         Collections.sort(rightSums);
         boolean found = false;
         for (long x : leftSums) {
            if (Collections.binarySearch(rightSums, S - x) >= 0) {
               found = true;
               break;
            }
         }
         if (found)
            System.out.println("YES");
         else
            System.out.println("NO");
      }
      sc.close();
   }
}
```

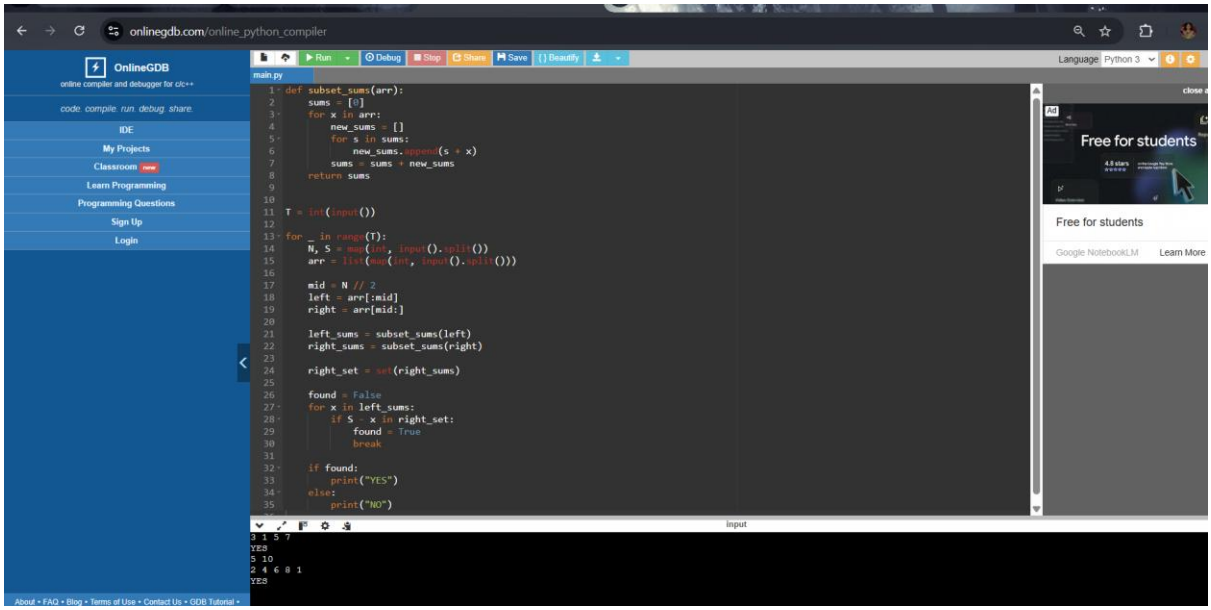




Code (python):

```
def subset_sums(arr):
    sums = [0]
for x in arr:
```

```
        new_sums = []
        for s in sums:
            new_sums.append(s + x)
        sums = sums + new_sums
    return sums
T = int(input())
for _ in range(T):
    N, S = map(int, input().split())
    arr = list(map(int, input().split()))
    mid = N // 2
    left = arr[:mid]
    right = arr[mid:]
    left_sums = subset_sums(left)
    right_sums = subset_sums(right)
    right_set = set(right_sums)
    found = False
    for x in left_sums:
        if S - x in right_set:
            found = True
            break
    if found:
        print("YES")
    else:
        print("NO")
```


onlinegdb.com/online_python_compiler
OnlineGDB
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial •
Run
Debug
Stop
Share
Save
{ } Beautify
Language Python 3
main.py
def subset_sums(arr):
sums = [0]
for x in arr:
new_sums = []
for s in sums:
new_sums.append(s + x)
sums = sums + new_sums
return sums
T = int(input())
for _ in range(T):
N, S = map(int, input().split())
arr = list(map(int, input().split()))
mid = N // 2
left = arr[:mid]
right = arr[mid:]
left_sums = subset_sums(left)
right_sums = subset_sums(right)
right_set = set(right_sums)
found = False
for x in left_sums:
if S - x in right_set:
found = True
break
if found:
print("YES")
else:
print("NO")
Input
3 1 5 7
YES
5 10
2 4 6 8 1
YES
Ad
Free for students
4.8 stars
Free for students
Google NotebookLM
Learn More