Competitive Programming

Name: Thulasi Shylasri

HTNO: 2303A51876

Batch-14

Date: 28-01-2026

Week-4 ASSIGNMENT(Wednesday Lab)

Assignment 1: Closest Subset Sum

Problem Statement:

You are given an array of N integers and a target value S.
Using Meet-in-the-Middle, find the subset whose sum is closest to S.
Output the minimum absolute difference.

Input Format:

• The first line contains an integer T.

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 the minimum absolute difference.

Constraints:

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

Sample input

1
4 10
1 4 7 12

Sample output

1

Python Code:

```
#T.shylasri(2303A51876)
#WEEK-4 ASSIGNMENT
#Wednesday Lab-(28-1-26)
import bisect
def closest_subset_sum(arr, S):
    n = len(arr)
    mid = n // 2
    left = arr[:mid]
    right = arr[mid:]
    # Generate subset sums
    left_sums = []
    for mask in range(1 << len(left)):
        s = 0
        for i in range(len(left)):
            if mask & (1 << i):
                s += left[i]
        left_sums.append(s)
    right_sums = []
    for mask in range(1 << len(right)):
        s = 0
        for i in range(len(right)):
            if mask & (1 << i):
                s += right[i]
        right_sums.append(s)
    right_sums.sort()
    ans = float('inf')
    for ls in left_sums:
        target = S - ls
```

```
        idx = bisect.bisect_left(right_sums, target)
        if idx < len(right_sums):
            ans = min(ans, abs(ls + right_sums[idx] - S))
        if idx > 0:
            ans = min(ans, abs(ls + right_sums[idx - 1] - S))
    return ans
T = int(input())
for _ in range(T):
    N, S = map(int, input().split())
    arr = list(map(int, input().split()))
    print(closest_subset_sum(arr, S))
```

Python Code Screenshot:

Python Output:

C Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
long long left_sums[1 << 20];
long long right_sums[1 << 20];
int cmp(const void *a, const void *b) {
    long long x = *(long long *)a;
    long long y = *(long long *)b;
    return (x > y) - (x < y);
}
int main() {
    int T;
```

```
  scanf("%d", &T);
while (T--) {
    int N;
    long long S;
    scanf("%d %lld", &N, &S);
    long long arr[40];
    for (int i = 0; i < N; i++) {
       scanf("%lld", &arr[i]);
    }
    int mid = N / 2;
    int n1 = mid;
    int n2 = N - mid;

    int sz1 = 1 << n1;
    int sz2 = 1 << n2;
    // Left subset sums
    for (int mask = 0; mask < sz1; mask++) {
       long long sum = 0;
       for (int i = 0; i < n1; i++) {
          if (mask & (1 << i))
             sum += arr[i];
       }
       left_sums[mask] = sum;
    }
    // Right subset sums
    for (int mask = 0; mask < sz2; mask++) {
       long long sum = 0;
       for (int i = 0; i < n2; i++) {
          if (mask & (1 << i))
```

```
                sum += arr[mid + i];
            }
            right_sums[mask] = sum;
        }
        qsort(right_sums, sz2, sizeof(long long), cmp);
        long long ans = LLONG_MAX;
        // Binary search
        for (int i = 0; i < sz1; i++) {
            long long need = S - left_sums[i];
            int l = 0, r = sz2 - 1;
            while (l <= r) {
                int m = (l + r) / 2;
                long long total = left_sums[i] + right_sums[m];
                long long diff = llabs(total - S);
                if (diff < ans)
                    ans = diff;
                if (right_sums[m] < need)
                    l = m + 1;
                else
                    r = m - 1;
            }
        }
        printf("%lld\n", ans);
    }
    return 0;
}
```

C Code Screenshot:

C Output:


input
4
4 10
1 4 7 11
1
...Program finished with exit code 0
Press ENTER to exit console.