

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:

- $1 \leq T \leq 20$
- $1 \leq N \leq 40$
- $-10^9 \leq A[i] \leq 10^9$

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:



The screenshot shows a Jupyter Notebook interface with a file named 'Untitled192.ipynb'. The code is written in Python and implements a function to find the closest subset sum. The code is as follows:

```

# @_why_lasri(23031401876)
# WEEK-2 ASSIGNMENT
# Wednesday Lab-(28-5-20)
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))

```



```
left_sum.append(s)
right_sum = []
for mask in range(1 << len(right)):
    s = 0
    for i in range(len(right)):
        if mask & (1 << i):
            s += right[i]
    right_sum.append(s)
right_sum.sort()
ans = float('inf')
for ls in left_sum:
    target = S - ls
    idx = bisect.bisect_left(right_sum, target)
    if idx < len(right_sum):
        ans = min(ans, ans(ls + right_sum[idx] - S))
    if idx > 0:
        ans = min(ans, ans(ls + right_sum[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 Output:



```
1
4 10
1 4 7 12
1
```

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:

```
OnlineGDB
online compiler and debugger for c/c++

Welcome, 2303A51876

Create New Project
My Projects
Classroom
Learn Programming
Programming Questions
Upgrade
Logout

auth0
All the connections, none of the hassle. Integrated OAuth and social connections on our free Plan.
ADD VIA IMPORT/EXPORT

main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <limits.h>
4 long long left_sums[1 << 20];
5 long long right_sums[1 << 20];
6 int cmp(const void *a, const void *b) {
7     long long x = *(long long *)a;
8     long long y = *(long long *)b;
9     return (x > y) - (x < y);
10 }
11 int main() {
12     int T;
13     scanf("%d", &T);
14     while (T--) {
15         int N;
16         long long S;
17         scanf("%d %lld", &N, &S);
18         long long arr[40];
19         for (int i = 0; i < N; i++) {
20             scanf("%lld", &arr[i]);
21         }
22         int mid = N / 2;
23         int n1 = mid;
24         int n2 = N - mid;
25
26         int sz1 = 1 << n1;
27         int sz2 = 1 << n2;
28         // Left subset sums
29         for (int mask = 0; mask < sz1; mask++) {
30             long long sum = 0;
31             for (int i = 0; i < n1; i++) {
32                 if (mask & (1 << i)) {
33                     sum += arr[i];
34                 }
35             }
36             left_sums[mask] = sum;
37         }
38         // Right subset sums
39         for (int mask = 0; mask < sz2; mask++) {
40             long long sum = 0;
41             for (int i = 0; i < n2; i++) {
42                 if (mask & (1 << i)) {
43                     sum += arr[mid + i];
44                 }
45             }
46             right_sums[mask] = sum;
47         }
48         sort(right_sums, sz2, sizeof(long long), cmp);
49         long long ans = LLONG_MAX;
50         // Binary search
51         for (int i = 0; i < sz1; i++) {
52             long long need = S - left_sums[i];
53             int l = 0, r = sz2 - 1;
54             while (l <= r) {
55                 int m = (l + r) / 2;
56                 long long total = left_sums[i] + right_sums[m];
57                 long long diff = llabs(total - S);
58                 if (diff < ans) {
59                     ans = diff;
60                 }
61                 if (right_sums[m] < need) {
62                     l = m + 1;
63                 }
64                 else {
65                     r = m - 1;
66                 }
67             }
68         }
69         printf("%lld\n", ans);
70     }
71     return 0;
72 }
```

```
main.c
32     for (int i = 0; i < n1; i++) {
33         if (mask & (1 << i)) {
34             sum += arr[i];
35         }
36     }
37     left_sums[mask] = sum;
38 }
39 // Right subset sums
40 for (int mask = 0; mask < sz2; mask++) {
41     long long sum = 0;
42     for (int i = 0; i < n2; i++) {
43         if (mask & (1 << i)) {
44             sum += arr[mid + i];
45         }
46     }
47     right_sums[mask] = sum;
48 }
49 sort(right_sums, sz2, sizeof(long long), cmp);
50 long long ans = LLONG_MAX;
51 // Binary search
52 for (int i = 0; i < sz1; i++) {
53     long long need = S - left_sums[i];
54     int l = 0, r = sz2 - 1;
55     while (l <= r) {
56         int m = (l + r) / 2;
57         long long total = left_sums[i] + right_sums[m];
58         long long diff = llabs(total - S);
59         if (diff < ans) {
60             ans = diff;
61         }
62         if (right_sums[m] < need) {
63             l = m + 1;
64         }
65         else {
66             r = m - 1;
67         }
68     }
69 }
70 printf("%lld\n", ans);
71 }
72 return 0;
73 }
```

```
main.c
37 // Right subset sums
38 for (int mask = 0; mask < sz2; mask++) {
39     long long sum = 0;
40     for (int i = 0; i < n2; i++) {
41         if (mask & (1 << i)) {
42             sum += arr[mid + i];
43         }
44     }
45     right_sums[mask] = sum;
46 }
47 sort(right_sums, sz2, sizeof(long long), cmp);
48 long long ans = LLONG_MAX;
49 // Binary search
50 for (int i = 0; i < sz1; i++) {
51     long long need = S - left_sums[i];
52     int l = 0, r = sz2 - 1;
53     while (l <= r) {
54         int m = (l + r) / 2;
55         long long total = left_sums[i] + right_sums[m];
56         long long diff = llabs(total - S);
57         if (diff < ans) {
58             ans = diff;
59         }
60         if (right_sums[m] < need) {
61             l = m + 1;
62         }
63         else {
64             r = m - 1;
65         }
66     }
67 }
68 printf("%lld\n", ans);
69 }
70 return 0;
71 }
```

C Output:



A screenshot of a terminal window titled "input". The window has a dark background with light blue text. The output shows a list of numbers: 3, 4, 10, 1, 4, 7, 12, and 5. Below the list, a message states "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a cursor.

```
3
4
10
1 4 7 12
5

...Program finished with exit code 0
Press ENTER to exit console.
```