

Competitive Programming

Name: Thulasi Shylasri

HTNO: 2303A51876

Batch-14

Date: 04-02-2026

Week-6 ASSIGNMENT (Wednesday Lab)

Assignment 1:

Practical Exercises with Fenwick Trees -Binary Indexed Trees:Problem: Library Book Borrowing Records

Problem Statement :

A university library records the number of books borrowed each day. Due to late returns or corrections, daily records may change. You are required to efficiently support:

- 1.Prefix Query – Find the total number of books borrowed from Day 1 to Day x
 - 2.Update Operation – Update the number of books borrowed on a given day
- Implement a Binary Indexed Tree (Fenwick Tree) to process these operations in $O(\log n)$ time.

Input Format

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

For each test case:

- The first line contains an integer N, the number of days
 - The second line contains N space-separated integers, representing books borrowed each day
 - The third line contains an integer Q, the number of queries
 - The next Q lines contain queries of the form:
SUM x → Find total books borrowed till Day x
UPDATE i val → Increase books borrowed on Day i by val
- Output Format
For each SUM query, print the result on a new line.

Constraints

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$
- $-10^9 \leq \text{arr}[i] \leq 10^9$
- $1 \leq Q \leq 200000$
- $0 \leq i < N$

Sample Input

```
1
6
12 15 10 20 18 25
4
SUM 4
UPDATE 3 5
SUM 4
SUM 6
```

Sample Output

```
57
62
105
```

Python Code:

```
#T.shylasri(2303A51876)
#WEEK-6 ASSIGNMENT
#Wednesday Lab-(04-02-26)

class FenwickTree:
    def __init__(self, n):
        self.n = n
        self.bit = [0] * (n + 1)

    def update(self, i, val):
        while i <= self.n:
            self.bit[i] += val
            i += i & -i

    def query(self, i):
```

```

s = 0
while i > 0:
    s += self.bit[i]
    i -= i & -i
return s

t = int(input())
for _ in range(t):
    n = int(input())
    arr = list(map(int, input().split()))
    ft = FenwickTree(n)
    # Build Fenwick Tree
    for i in range(n):
        ft.update(i + 1, arr[i])
    q = int(input())
    for _ in range(q):
        query = input().split()
        if query[0] == "SUM":
            x = int(query[1])
            print(ft.query(x))
        elif query[0] == "UPDATE":
            i = int(query[1])
            val = int(query[2])
            ft.update(i, val)

```

Python Code Screenshot:

```
Untitled198.ipynb ☆
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all +

[4]
#1.shylor1(2385A31876)
week-6-ASSIGNMENT-1
Wednesday Lab-(04-02-20)
class FenwickTree:
    def __init__(self, n):
        self.n = n
        self.bit = [0] * (n + 1)
    def update(self, i, val):
        while i <= self.n:
            self.bit[i] += val
            i += i & -i
    def query(self, i):
        s = 0
        while i > 0:
            s += self.bit[i]
            i -= i & -i
        return s
t = int(input())
for _ in range(t):
    n = int(input())
    arr = list(map(int, input().split()))
    ft = FenwickTree(n)
    # Build Fenwick Tree
    for i in range(n):
        ft.update(i + 1, arr[i])
    q = int(input())
    for _ in range(q):
        query = input().split()
        if query[0] == "SUM":
            x = int(query[1])
            print(ft.query(x))
        elif query[0] == "UPDATE":
            i = int(query[1])
            val = int(query[2])
            ft.update(i, val)
```

```
Untitled198.ipynb ☆
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all +

[5]
#1.shylor1(2385A31876)
week-6-ASSIGNMENT-1
Wednesday Lab-(04-02-20)
class FenwickTree:
    def __init__(self, n):
        self.n = n
        self.bit = [0] * (n + 1)
    def update(self, i, val):
        while i <= self.n:
            self.bit[i] += val
            i += i & -i
    def query(self, i):
        s = 0
        while i > 0:
            s += self.bit[i]
            i -= i & -i
        return s
t = int(input())
for _ in range(t):
    n = int(input())
    arr = list(map(int, input().split()))
    ft = FenwickTree(n)
    # Build Fenwick Tree
    for i in range(n):
        ft.update(i + 1, arr[i])
    q = int(input())
    for _ in range(q):
        query = input().split()
        if query[0] == "SUM":
            x = int(query[1])
            print(ft.query(x))
        elif query[0] == "UPDATE":
            i = int(query[1])
            val = int(query[2])
            ft.update(i, val)
```

Python Output:

```
Untitled198.ipynb ☆
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all +

[5]
1
6
12 15 18 20 22 25
4
SUM 4
57
UPDATE 3 5
SUM 4
62
SUM 6
105
```

C Program:

```
#include <stdio.h>

#include <stdlib.h>

long long *BIT;

int N;

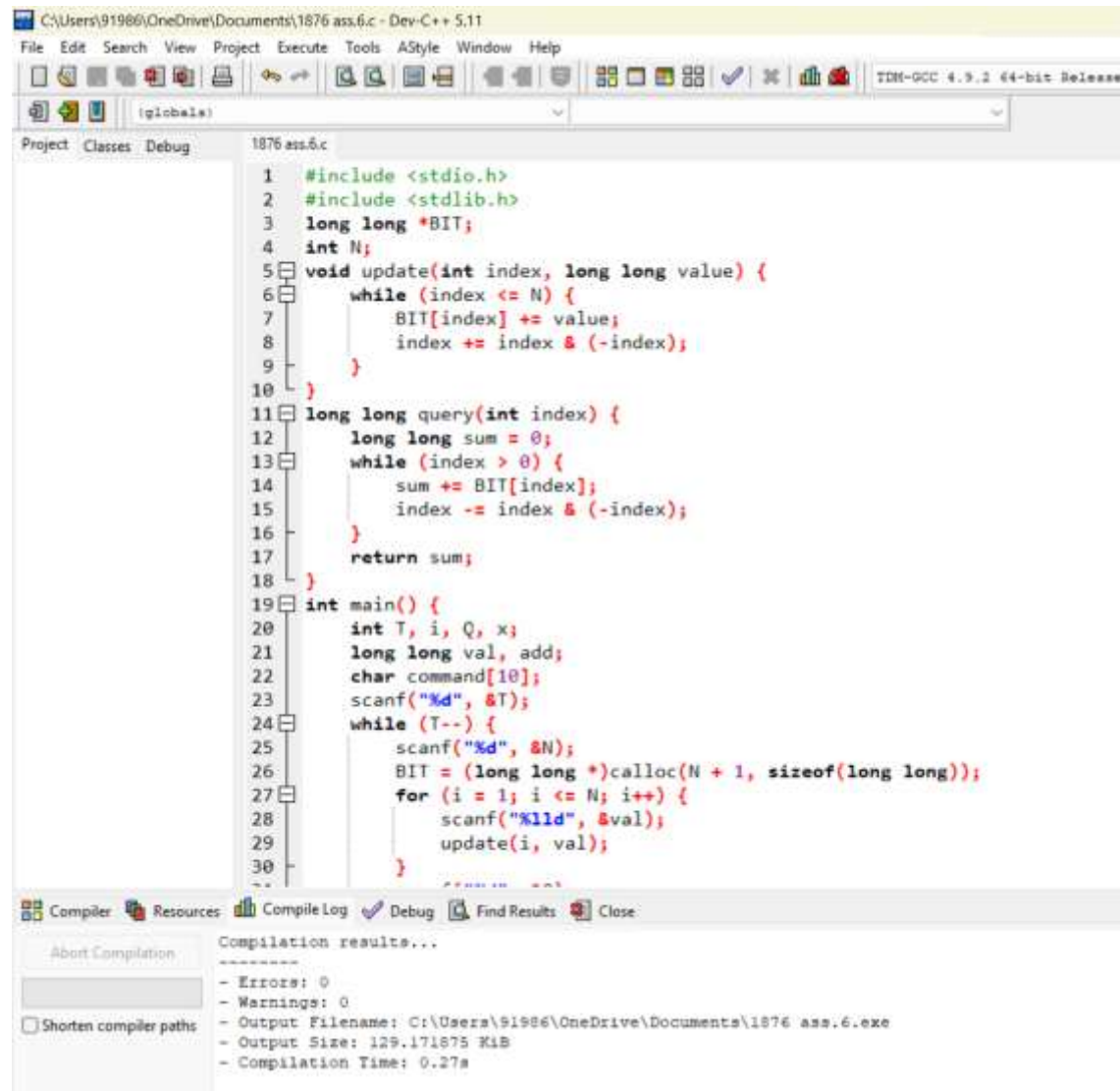
void update(int index, long long value) {
    while (index <= N) {
        BIT[index] += value;
        index += index & (-index);
    }
}

long long query(int index) {
    long long sum = 0;
    while (index > 0) {
        sum += BIT[index];
        index -= index & (-index);
    }
    return sum;
}

int main() {
    int T, i, Q, x;
    long long val, add;
    char command[10];
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &N);
        BIT = (long long *)calloc(N + 1, sizeof(long long));
        for (i = 1; i <= N; i++) {
            scanf("%lld", &val);
```

```
        update(i, val);
    }
    scanf("%d", &Q);
    while (Q--) {
        scanf("%s", command);
        if (command[0] == 'S') {    // SUM
            scanf("%d", &x);
            printf("%lld\n", query(x));
        }
        else if (command[0] == 'U') { // UPDATE
            scanf("%d %lld", &x, &add);
            update(x, add);
        }
    }
    free(BIT);
}
return 0;
}
```

C Code Screenshot:



The screenshot displays a C code editor window titled "C:\Users\91986\OneDrive\Documents\1876 ass.6.c - Dev-C++ 5.11". The code implements a BIT (Binary Indexed Tree) structure. It includes `<stdio.h>` and `<stdlib.h>`, declares a `long long *BIT;` array, and defines a `void update(int index, long long value)` function. The `update` function uses a while loop to update the BIT array. A `long long query(int index)` function is also defined, which calculates the sum of elements up to a given index. The `main` function reads input for the number of elements `N` and the number of queries `Q`, initializes the BIT array, and processes each query by updating the BIT and querying the sum.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 long long *BIT;
4 int N;
5 void update(int index, long long value) {
6     while (index <= N) {
7         BIT[index] += value;
8         index += index & (-index);
9     }
10 }
11 long long query(int index) {
12     long long sum = 0;
13     while (index > 0) {
14         sum += BIT[index];
15         index -= index & (-index);
16     }
17     return sum;
18 }
19 int main() {
20     int T, i, Q, x;
21     long long val, add;
22     char command[10];
23     scanf("%d", &T);
24     while (T--) {
25         scanf("%d", &N);
26         BIT = (long long *)calloc(N + 1, sizeof(long long));
27         for (i = 1; i <= N; i++) {
28             scanf("%lld", &val);
29             update(i, val);
30         }
31     }
32 }
```

The bottom panel shows the "Compiler" tab with the "Compile Log" selected. It displays the following compilation results:

```
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91986\OneDrive\Documents\1876 ass.6.exe
- Output Size: 129.171875 KiB
- Compilation Time: 0.27s
```

```
18 }
19 int main() {
20     int T, i, Q, x;
21     long long val, add;
22     char command[10];
23     scanf("%d", &T);
24     while (T--) {
25         scanf("%d", &N);
26         BIT = (long long *)calloc(N + 1, sizeof(long long));
27         for (i = 1; i <= N; i++) {
28             scanf("%lld", &val);
29             update(i, val);
30         }
31         scanf("%d", &Q);
32         while (Q--) {
33             scanf("%s", command);
34             if (command[0] == 'S') { // SUM
35                 scanf("%d", &x);
36                 printf("%lld\n", query(x));
37             }
38             else if (command[0] == 'U') { // UPDATE
39                 scanf("%d %lld", &x, &add);
40                 update(x, add);
41             }
42         }
43         free(BIT);
44     }
45     return 0;
46 }
47
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91986\OneDrive\Documents\1876 ass.6.exe
- Output Size: 129.171875 KiB
- Compilation Time: 0.27s

C Output:

```
1
6
12 15 10 20 18 25
4
SUM 4
57
UPDATE 3 5
SUM 4
62
SUM 6
105

-----
Process exited after 59.79 seconds with return value 0
Press any key to continue . . .
```


Assignment 2:

Python Code:

```
#T.shylasri(2303A51876)

#WEEK-6 ASSIGNMENT-2

#Wednesday Lab-(04-02-26)

class FenwickTree:

    def __init__(self, n):

        self.n = n

        self.bit = [0] * (n + 1)

    def update(self, i, val):

        while i <= self.n:

            self.bit[i] += val

            i += i & -i

    def query(self, i):

        s = 0

        while i > 0:

            s += self.bit[i]

            i -= i & -i

        return s

# USER INPUT

n = int(input("Enter number of days: "))

arr = list(map(int, input("Enter daily patient count: ").split()))

ft = FenwickTree(n)

# Build Fenwick Tree

for i in range(n):

    ft.update(i + 1, arr[i])

# First Query

day = int(input("Enter day to find total patients till: "))
```

```
print("Total patients till Day", day, "=", ft.query(day))
```

Update Operation

```
update_day = int(input("Enter day to update: "))
```

```
new_val = int(input("Enter new patient count: "))
```

```
diff = new_val - arr[update_day - 1]
```

```
arr[update_day - 1] = new_val
```

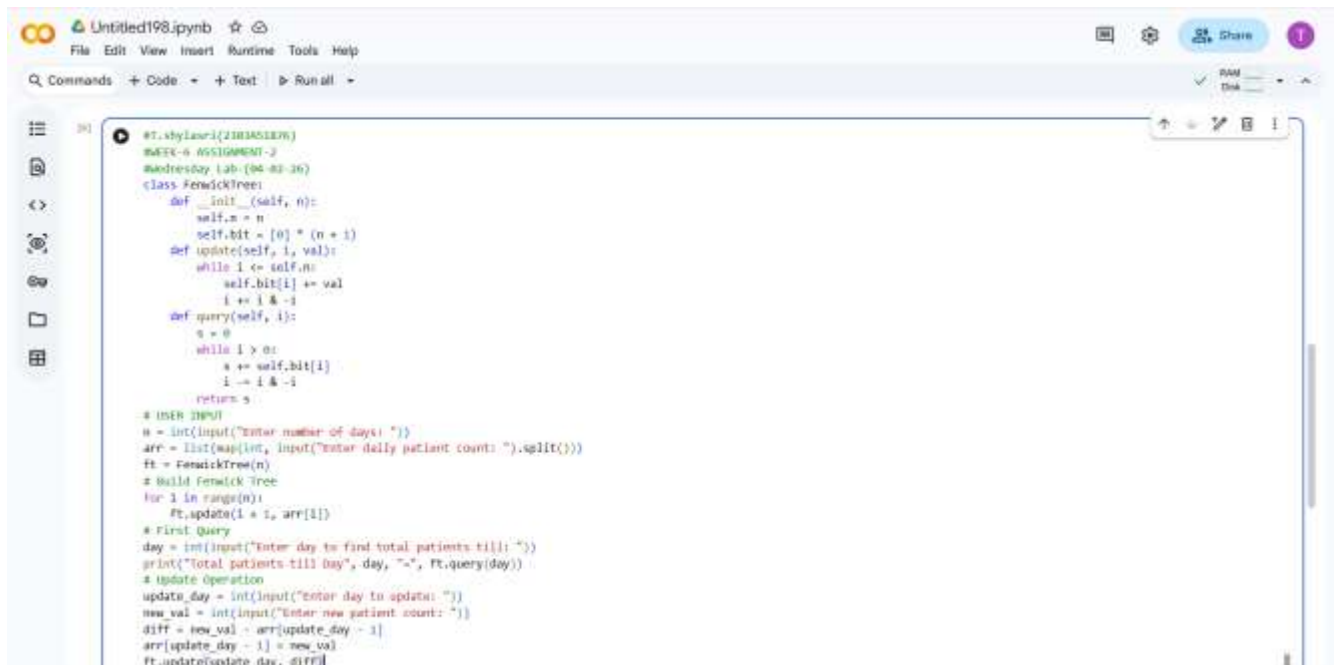
```
ft.update(update_day, diff)
```

Second Query

```
day = int(input("Enter day to find total patients till after update: "))
```

```
print("After update, total patients till Day", day, "=", ft.query(day))
```

Python Code Screenshot:



```
#T.shylauri(2181351176)
#DATE-6 ASSIGNMENT-2
#Address Lab (04-01-26)
class FenwickTree:
    def __init__(self, n):
        self.n = n
        self.bit = [0] * (n + 1)
    def update(self, i, val):
        while i <= self.n:
            self.bit[i] += val
            i += i & -i
    def query(self, i):
        s = 0
        while i > 0:
            s += self.bit[i]
            i -= i & -i
        return s
# USER INPUT
n = int(input("Enter number of days: "))
arr = list(map(int, input("Enter daily patient counts: ").split()))
ft = FenwickTree(n)
# Build Fenwick Tree
for i in range(n):
    ft.update(i + 1, arr[i])
# First Query
day = int(input("Enter day to find total patients till: "))
print("Total patients till Day", day, "=", ft.query(day))
# Update Operation
update_day = int(input("Enter day to update: "))
new_val = int(input("Enter new patient count: "))
diff = new_val - arr[update_day - 1]
arr[update_day - 1] = new_val
ft.update(update_day, diff)
```

```

Untitled198.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all

[6]
class FenwickTree:
    def __init__(self, n):
        self.n = n
        self.bit = [0] * (n + 1)
    def update(self, i, val):
        while i <= self.n:
            self.bit[i] += val
            i = i & -i
    def query(self, i):
        s = 0
        while i > 0:
            s += self.bit[i]
            i = i & -i
        return s

# Using Input
n = int(input("Enter number of days: "))
arr = list(map(int, input("Enter daily patient count: ").split()))
ft = FenwickTree(n)
# Build Fenwick Tree
for i in range(n):
    ft.update(i + 1, arr[i])
# First Query
day = int(input("Enter day to find total patients till: "))
print("Total patients till day", day, "=", ft.query(day))
# Update operation
update_day = int(input("Enter day to update: "))
new_val = int(input("Enter new patient count: "))
diff = new_val - arr[update_day - 1]
arr[update_day - 1] = new_val
ft.update(update_day, diff)
# Second Query
day = int(input("Enter day to find total patients till after update: "))
print("After update, total patients till Day", day, "=", ft.query(day))

```

Python Output:

```

Untitled198.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all

[6]
day = int(input("Enter day to find total patients till after update: "))
print("After update, total patients till Day", day, "=", ft.query(day))

... Enter number of days: 7
Enter daily patient count: 18 22 20 25 19 23 21
Enter day to find total patients till: 5
Total patients till Day 5 = 104
Enter day to update: 4
Enter new patient count: 27
Enter day to find total patients till after update: 5
After update, total patients till Day 5 = 106

```

C Program:

```

#include <stdio.h>

#include <stdlib.h>

int N;

int *BIT;

void update(int index, int value) {
    while (index <= N) {

```

```

        BIT[index] += value;

        index += index & (-index);
    }
}

int query(int index) {
    int sum = 0;
    while (index > 0) {
        sum += BIT[index];
        index -= index & (-index);
    }
    return sum;
}

int main() {
    int i, day, update_day, new_val, diff;
    printf("Enter number of days: ");
    scanf("%d", &N);
    int *patients = (int *)malloc(N * sizeof(int));
    printf("Enter daily patient count:\n");
    for (i = 0; i < N; i++) {
        scanf("%d", &patients[i]);
    }
    BIT = (int *)calloc(N + 1, sizeof(int));
    for (i = 1; i <= N; i++) {
        update(i, patients[i - 1]);
    }
    printf("Enter day to find total patients till: ");
    scanf("%d", &day);
    printf("Total patients till Day %d = %d\n", day, query(day));
    printf("Enter day to update: ");

```

```

scanf("%d", &update_day);

printf("Enter new patient count: ");

scanf("%d", &new_val);

diff = new_val - patients[update_day - 1];

patients[update_day - 1] = new_val;

update(update_day, diff);

printf("Enter day to find total patients till after update: ");

scanf("%d", &day);

printf("After update, total patients till Day %d = %d\n", day, query(day));

free(BIT);

free(patients);

return 0;
}

```

C Code Screenshot:

```

C:\Users\91986\OneDrive\Documents\1876.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
1876.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3  int N;
4  int *BIT;
5  void update(int index, int value) {
6      while (index <= N) {
7          BIT[index] += value;
8          index += index & (-index);
9      }
10 }
11 int query(int index) {
12     int sum = 0;
13     while (index > 0) {
14         sum += BIT[index];
15         index -= index & (-index);
16     }
17     return sum;
18 }
19 int main() {
20     int i, day, update_day, new_val, diff;
21     printf("Enter number of days: ");
22     scanf("%d", &N);
23     int *patients = (int *)malloc(N * sizeof(int));
24     printf("Enter daily patient count:\n");
25     for (i = 0; i < N; i++) {
26         scanf("%d", &patients[i]);
27     }
28     BIT = (int *)calloc(N + 1, sizeof(int));
29     for (i = 1; i <= N; i++) {
30         update(i, patients[i - 1]);

```

The screenshot shows a C++ IDE with the file 1876.cpp open. The code implements a program to manage patient counts over time. It uses arrays for patient counts and a BIT (Binary Indexed Tree) for efficient range queries and updates. The program prompts the user for the number of days, daily patient counts, a day to query, a day to update, and a new patient count. It then outputs the total patient count up to the queried day and after the update.

```
19 int main() {
20     int i, day, update_day, new_val, diff;
21     printf("Enter number of days: ");
22     scanf("%d", &N);
23     int *patients = (int *)malloc(N * sizeof(int));
24     printf("Enter daily patient count:\n");
25     for (i = 0; i < N; i++) {
26         scanf("%d", &patients[i]);
27     }
28     BIT = (int *)calloc(N + 1, sizeof(int));
29     for (i = 1; i <= N; i++) {
30         update(i, patients[i - 1]);
31     }
32     printf("Enter day to find total patients till: ");
33     scanf("%d", &day);
34     printf("Total patients till Day %d = %d\n", day, query(day));
35     printf("Enter day to update: ");
36     scanf("%d", &update_day);
37     printf("Enter new patient count: ");
38     scanf("%d", &new_val);
39     diff = new_val - patients[update_day - 1];
40     patients[update_day - 1] = new_val;
41     update(update_day, diff);
42     printf("Enter day to find total patients till after update: ");
43     scanf("%d", &day);
44     printf("After update, total patients till Day %d = %d\n", day, query(day));
45     free(BIT);
46     free(patients);
47     return 0;
48 }
```

The compilation results show 0 errors and 0 warnings. The output filename is C:\Users\91986\OneDrive\Documents\1876.exe, the output size is 129.8623046675 KiB, and the compilation time is 0.39s.

C Output:

The terminal window shows the output of the program. The user enters 7 for the number of days, followed by the daily patient counts: 18, 22, 20, 25, 19, 23, 21. Then, the user enters 5 for the day to find the total patient count, which is 104. Next, the user enters 4 for the day to update, followed by a new patient count of 27. Finally, the user enters 5 for the day to find the total patient count after the update, which is 106. The program then exits with a return value of 0.

```
Enter number of days: 7
Enter daily patient count:
18 22 20 25 19 23 21
Enter day to find total patients till: 5
Total patients till Day 5 = 104
Enter day to update: 4
Enter new patient count: 27
Enter day to find total patients till after update: 5
After update, total patients till Day 5 = 106

-----
Process exited after 62.36 seconds with return value 0
Press any key to continue . . .
```