Batch: SY-IT(B3) **Experiment Number:3**

Roll Number: 16010423076 **Name:Ritesh Jha**

Aim of the Experiment:Implementation of Fenwick Tree operations

Program/ Steps:

Write a program to solve range-based query over an array for performing sum and update operation using Fenwick tree.

```
#include <iostream>
#include <vector>

using namespace std;

class FenwickTree {
private:
   vector<int> BIT;
   int n;

public:
   FenwickTree(int size) {
      n = size;
      BIT.resize(n + 1, 0);
   }

   void update(int index, int delta) {
      while (index <= n) {
         BIT[index] += delta;
         index += index & -index;
      }
   }

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

   int rangeQuery(int left, int right) {
      return query(right) - query(left - 1);
   }
};
```

```
int main() {
   int n;
   cout << "Enter the size of the array: ";
   cin >> n;

   vector<int> arr(n + 1);
   FenwickTree fenwick(n);

   cout << "Enter the elements of the array: ";
   for (int i = 1; i <= n; i++) {
      cin >> arr[i];
      fenwick.update(i, arr[i]);
   }

   int choice;
   do {
      cout << "\nMenu:\n1. Update Value\n2. Query Sum\n3. Exit\nEnter choice: ";
      cin >> choice;

      if (choice == 1) {
         int index, value;
         cout << "Enter index and new value: ";
         cin >> index >> value;
         int delta = value - arr[index];
         arr[index] = value;
         fenwick.update(index, delta);
      } else if (choice == 2) {
         int left, right;
         cout << "Enter range (left right): ";
         cin >> left >> right;
         cout << "Sum from index " << left << " to " << right << " is " << fenwick.rangeQuery(left, right)
<< endl;
      }
   } while (choice != 3);

   return 0;
}
```

Output/Result:

```
Output
Enter the size of the array: 5
Enter the elements of the array: 3 2 1 6 5

Menu:
1. Update Value
2. Query Sum
3. Exit
Enter choice: 2
Enter range (left right): 1 3
Sum from index 1 to 3 is 6

Menu:
1. Update Value
2. Query Sum
3. Exit
Enter choice: 1
Enter index and new value: 3 9
```

```
Menu:
1. Update Value
2. Query Sum
3. Exit
Enter choice: 2
Enter range (left right): 1 3
Sum from index 1 to 3 is 14

Menu:
1. Update Value
2. Query Sum
3. Exit
Enter choice: 3


=== Code Execution Successful ===
```

Outcomes: CO2. Understand the fundamental concepts for managing the data using different data structures such as lists, queues, trees etc.

Conclusion (based on the Results and outcomes achieved):

From this experiment, I learned how to implement a Fenwick Tree to efficiently handle range sum queries and updates in an array. I understood how the tree structure helps in reducing the time complexity of these operations to O(log N), making it useful for competitive programming. By working on this program, I

improved my understanding of how binary indexed trees store and manipulate data using bitwise operations.

References:

1. https://www.hackerearth.com/practice/data-structures/advanced-data-ructures/segment-trees/tutorial/
2. https://cp-algorithms.com/data_structures/segment_tree.html