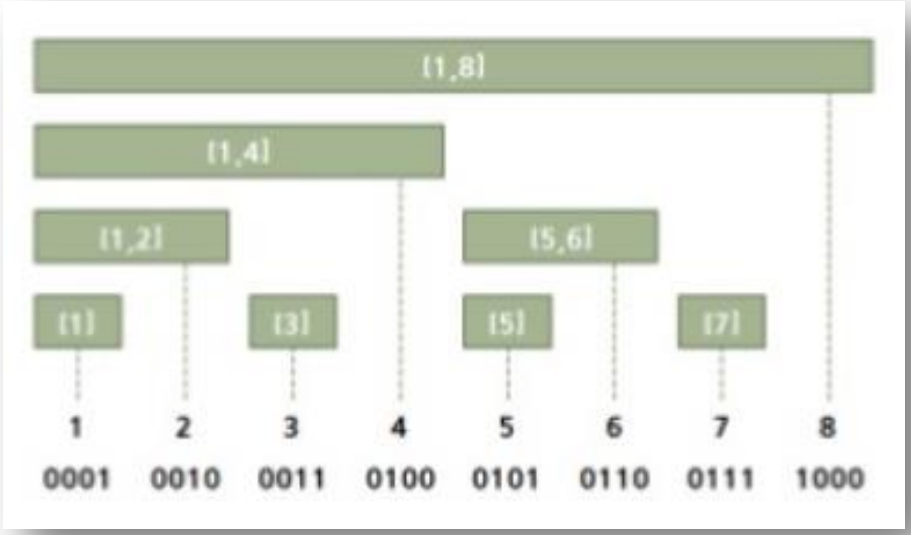
STAR LEAGUE

# 05 Fenwick Tree (Binary Indexed Tree – BIT)



# Content

- BIT Definition
- How Does It Work
- Array Implementation (Point Update and Range Query)
- Cumulative Frequency with Mapping
- Counting Inversions
- Range Update and Point Query
- Range Update and Range Query
- 2D BIT
- Problems

# BIT Definition

Fenwick Tree is a user defined data structure that fundamentally provides a way to represent **an array of numbers** in an array, allowing

- **prefix sums** and
- **point update**

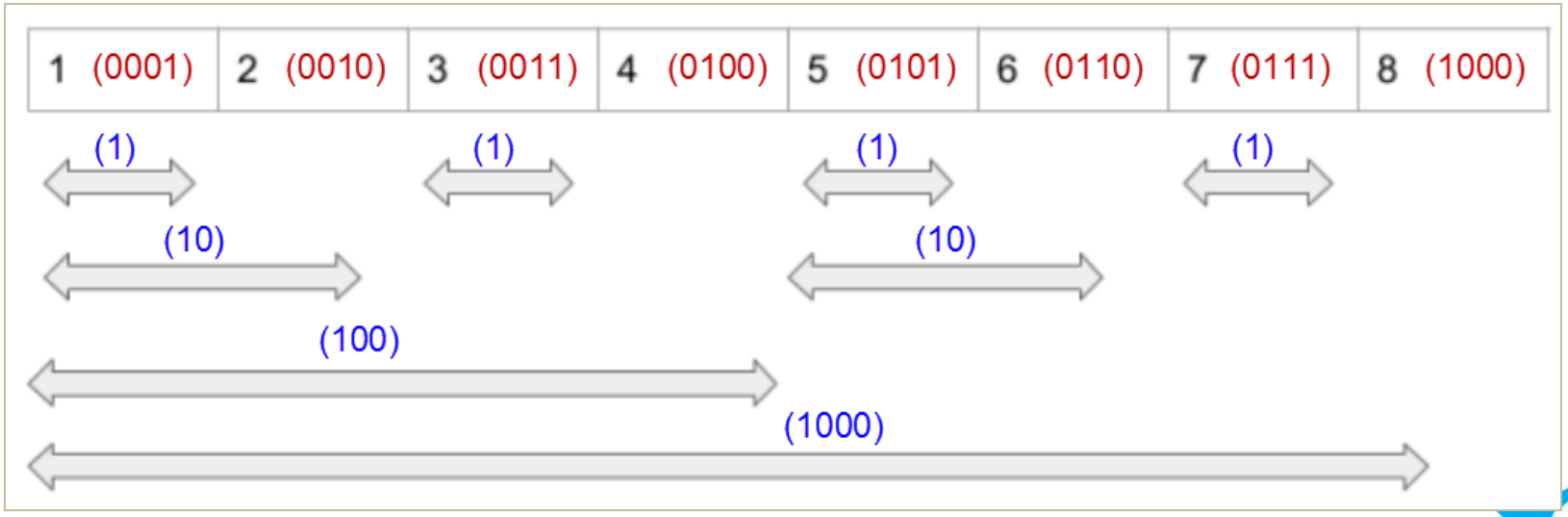
to be calculated efficiently in logarithmic **O(lgN)** time.

# Motivation

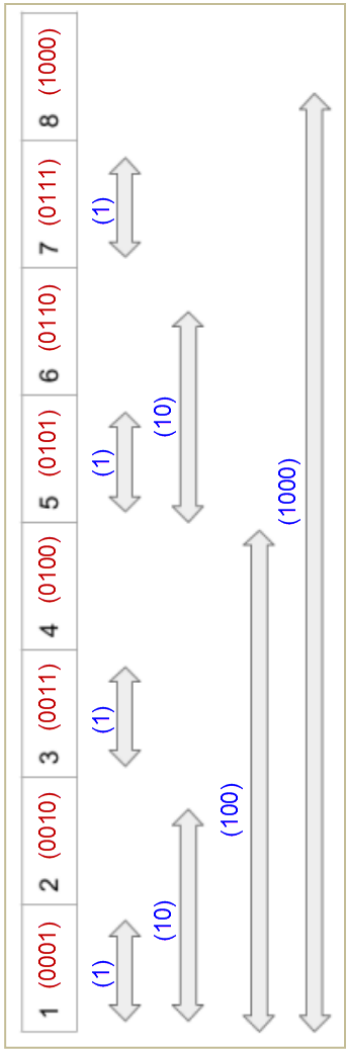
- Each integer can be represented as a sum of powers of two. In the same way, legnth of each interval (range) can be represented as a sum of powers of two.
- $8 = 2^3$ and $15 = 2^3 + 2^2 + 2^1 + 2^0$.
- 8 = 1000, 15 =1111 in binary.
- There are **logN digits** in binary representing of the number N.

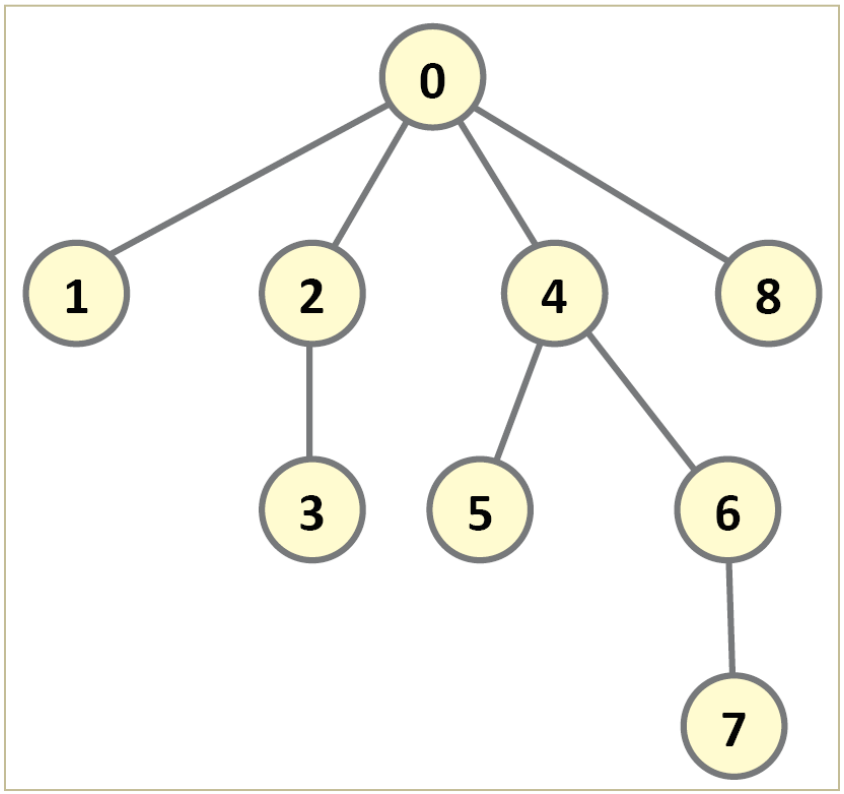
# Motivation (Continue)

- **Last set bit** of each index represent length of the range that is ending in this position. The value in that position is sum of the values in the range.



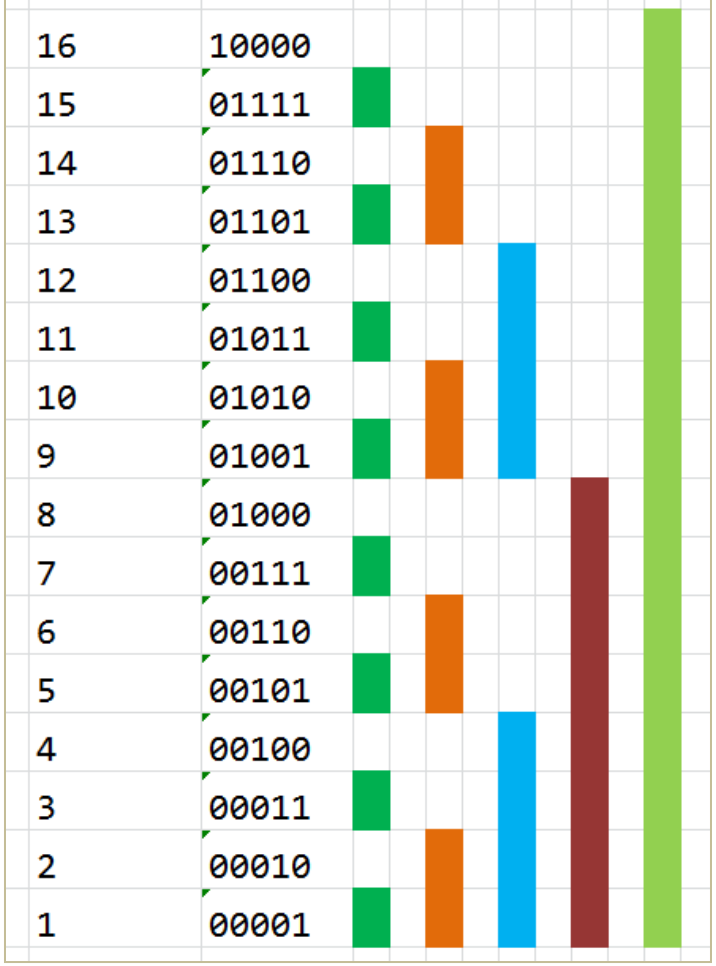
# Motivation (Continue)

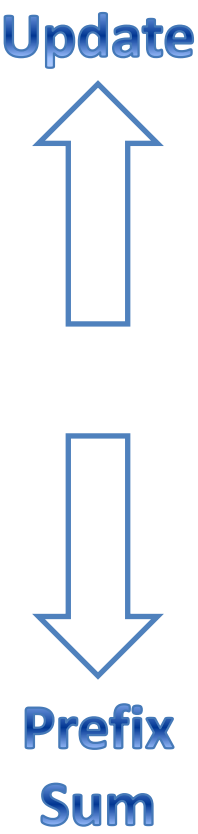


# Motivation (Continue)

| DECIMAL | BINARY |
|---|---|
| 16 | 10000 |
| 15 | 01111 |
| 14 | 01110 |
| 13 | 01101 |
| 12 | 01100 |
| 11 | 01011 |
| 10 | 01010 |
| 9 | 01001 |
| 8 | 01000 |
| 7 | 00111 |
| 6 | 00110 |
| 5 | 00101 |
| 4 | 00100 |
| 3 | 00011 |
| 2 | 00010 |
| 1 | 00001 |





# Fenwick Tree Implementation

```
struct BIT
{
    vector <int> tree;
    int n;
    //-----------------------------------------
    //Create an empty tree
    void init(int size)
    {
        n = size;
        tree.resize(n, 0);
    }
    //-----------------------------------------
    //Point update.
    void update(int i, int delta)
    {
        while (i < n)
        {
            tree[i] += delta;
            i += i & (-i); //add lowest set bit
        }
    }
    //-----------------------------------------
    int prefixSum(int i)
    {
        int sum = 0;
        while (i > 0)
        {
            sum += tree[i];
            i -= i & (-i); //subtract lowest set bit.
        }
        return sum;
    }
    //-----------------------------------------
    int rangeSum(int i, int j)
    {
        return prefixSum(j) - prefixSum(i - 1);
    }
};
```

```
//-----------------------------------------
int main()
{
    vector<int> values = {0, 3, 5, 2, 2, 7, 3, -2, 4, -1, 3};
    BIT bit;
    bit.init(values.size() + 1);

    for (int i = 1; i <values.size(); i++)
    {
        bit.update(i, values[i]);
    }
    cout << bit.rangeSum(3, 7) << endl; //12
    bit.update(5, 5);
    cout << bit.rangeSum(3, 7) << endl; //17
    return 0;
}
```

# Cumulative Frequency

```
//-----------------------------------------------
int main()
{
    vector<int> values = { 3, 5, 500, 2, 2, 7, 3, 3 };
    int maxVal = *max_element(values.begin(), values.end());
    BIT bit;
    bit.init(maxVal + 1);   //500 + 1

    bit.update(values[0], 1);   //3
    bit.update(values[3], 1);   //2
    bit.update(values[5], 1);   //7

    cout << bit.prefixSum(5) << endl;   //2

    bit.update(values[7], 1);   //3
    cout << bit.prefixSum(5) << endl;   //3

    return 0;
}
```

# Cumulative Frequency with Relative Mapping

```
void doMapping(vector<int> v, vector<int> &compVec)
{
    int n = v.size();
    vector<int> sorted = v;
    sort(sorted.begin(), sorted.end());

    compVec.resize(n);
    for (int i = 0; i < n; i++)
    {
        int pos = lower_bound(sorted.begin(), sorted.end(), v[i]) - sorted.begin();
        compVec[i] = pos + 1;
    }
}

//-------------------------------------------------
int main()
{
    vector<int> values = { 9, 2, 500, -2, 2, 5};
    vector<int> compVec; //compressed vector

    doMapping(values, compVec); //4, 2, 5, 1, 2, 3

    BIT bit;
    bit.init(values.size() + 1);

    bit.update(compVec[0], 1);  //9 --> 5
    bit.update(compVec[1], 1);  //2 -- > 3
    bit.update(compVec[2], 1);  //500 --> 6

    cout << bit.prefixSum(compVec[5]) << endl;  //1

    bit.update(compVec[1], 1);          //2 -- > 2
    cout << bit.prefixSum(compVec[5]) << endl;  //2

    return 0;
}
```

# Counting Inversions with Fenwick Tree

```
int main()
{
    vector<int> values = { 6, 5, 5000, -2, 6, };
    int N = values.size();
    BIT bit;
    bit.init(N + 1);

    vector<pair<int, int>> pairVec;

    for (int i = 0; i < N; i++)
        pairVec.push_back({values[i], i + 1});

    sort(pairVec.begin(), pairVec.end());
    reverse(pairVec.begin(), pairVec.end());

    int res = 0;
    for (auto p : pairVec)
    {
        res += bit.prefixSum(p.second);
        bit.update(p.second, 1);
    }

    cout << res << endl;
}

//5
```

# Range Update and Point Query

```
// Range Update Point Query: O(NlgN)
struct BIT
{
    vector <int> tree;
    int n;
    //-----------------------------------------
    void init(int nn)
    {
        n = nn;
        tree.resize(n, 0);
    }
    //-----------------------------------------
    //Point update.
    void pointUpdate(int i, int delta)
    {
        while (i < n)
        {
            tree[i] += delta;
            i += i & (-i); //add lowest set bit
        }
    }
    //-----------------------------------------
    //Difference array
    void rangeUpdate(int left, int right, int delta)
    {
        pointUpdate(left, delta);
        pointUpdate(right + 1, -delta);
    }
    //-----------------------------------------
    int pointQuery(int i)
    {
        int sum = 0;
        int j = i;
        while (j > 0)
        {
            sum += tree[j];
            j -= j & (-j); //subtract lowest set bit.
        }
        return sum;
    }
};
```

```
//-----------------------------------------
int main()
{
    BIT bit;
    int N = 5;
    bit.init(N+1);                      //0 0 0 0 0

    cout << bit.pointQuery(5) << endl;  //0
    bit.rangeUpdate(2, 4, 10);          // 0 10 10 10 0
    cout << bit.pointQuery(3) << endl;  //10
    bit.rangeUpdate(1, 5, -5);          //-5 5 5 5 -5
    cout << bit.pointQuery(5) << endl;; //-5
}
```

# Problems

- Sleepwalking Cows (Starleague Training)
- Balanced Photo (USACO 2017 Jan, Gold)
- Haircut (USACO 2020 US Open, Gold)
- Why Did the Cow Cross the Road III (USACO 2017 Feb, Gold)