Binary Index Tree or Fenwick Tree

Given an array of n integers, your task is to process q queries of the following types:
1. update the value at position k to u
2. what is the sum of values in range [a,b]?

CSES-1648

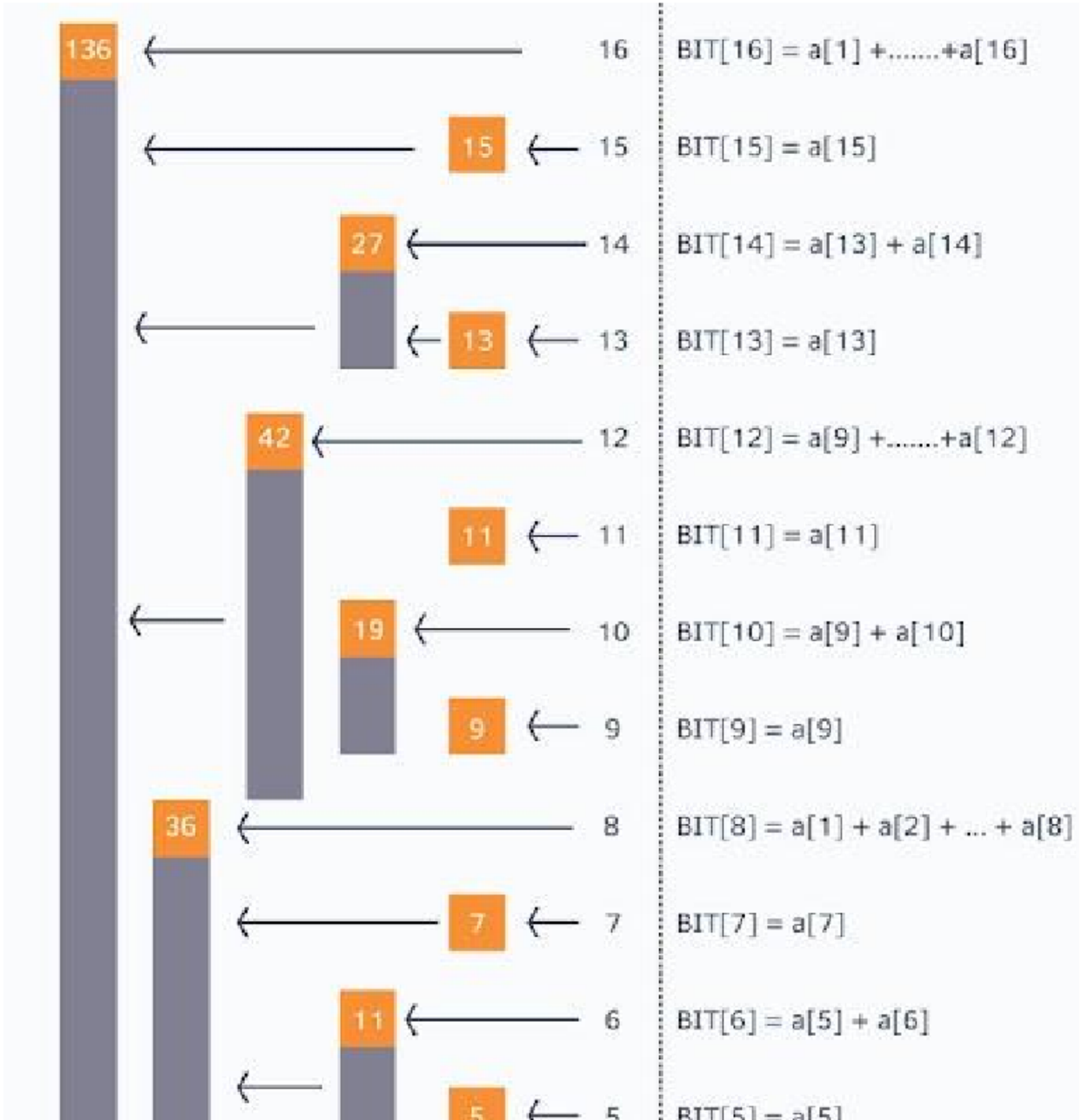
Features of BIT :
1. Update value at specifix index
2. Prefix sum/max/min etc
3. Range sum

Time Complexity :
Per Query - O(logn)
To Build - O(Nlogn)

Space Complexity : O(N)

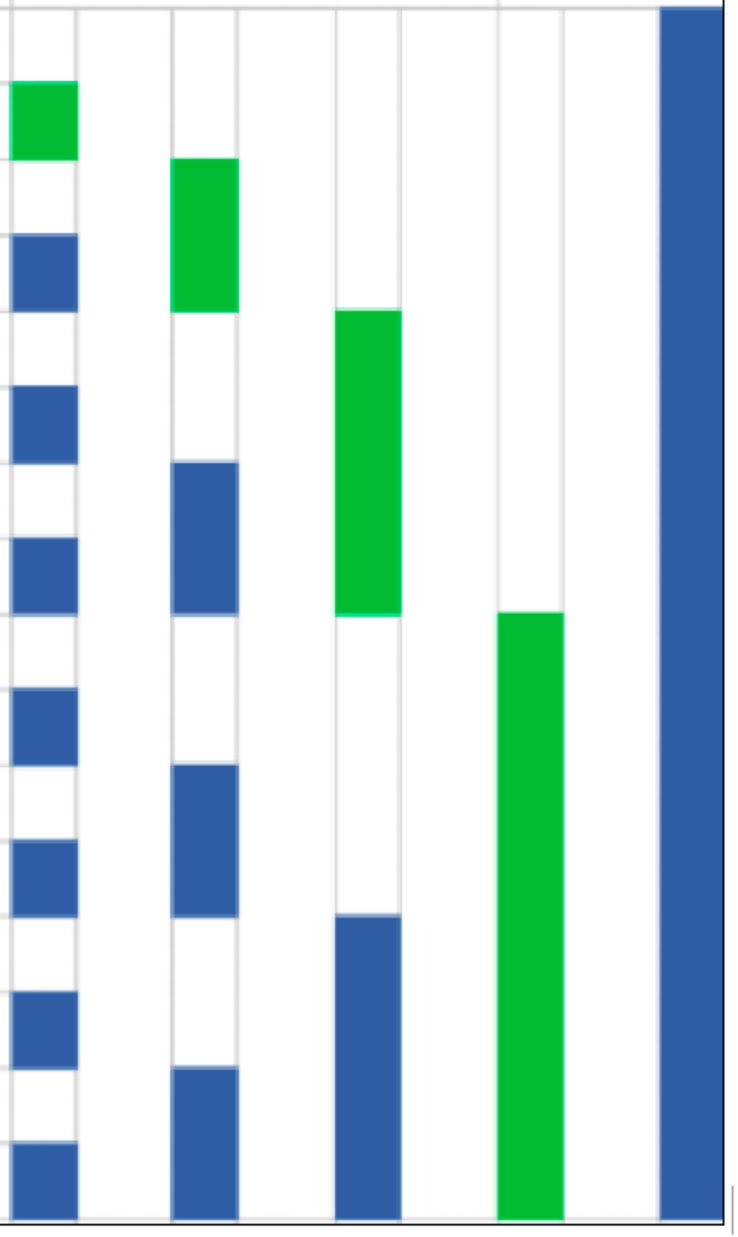
Code :

```
int  bit_tree[M+2]; // sob 0

void update(int idx,int  val)          //set a[idx]+=val;
{
    while(idx < M){
        bit_tree[idx] += val;
        idx += (idx&-idx);
    }
}

int  qry(int idx)        //returns the prefix sum from 0 to idx
{
    int  ret = 0;
    while(idx > 0){
        ret += bit_tree[idx];
        idx -= (idx&-idx);
    }
    return ret;
}
main()
{
    memset(bit_tree,0,sizeof bit_tree);
    for(int i=1;i<=n;i++)update(i,a[i]); // nlogn

}
```

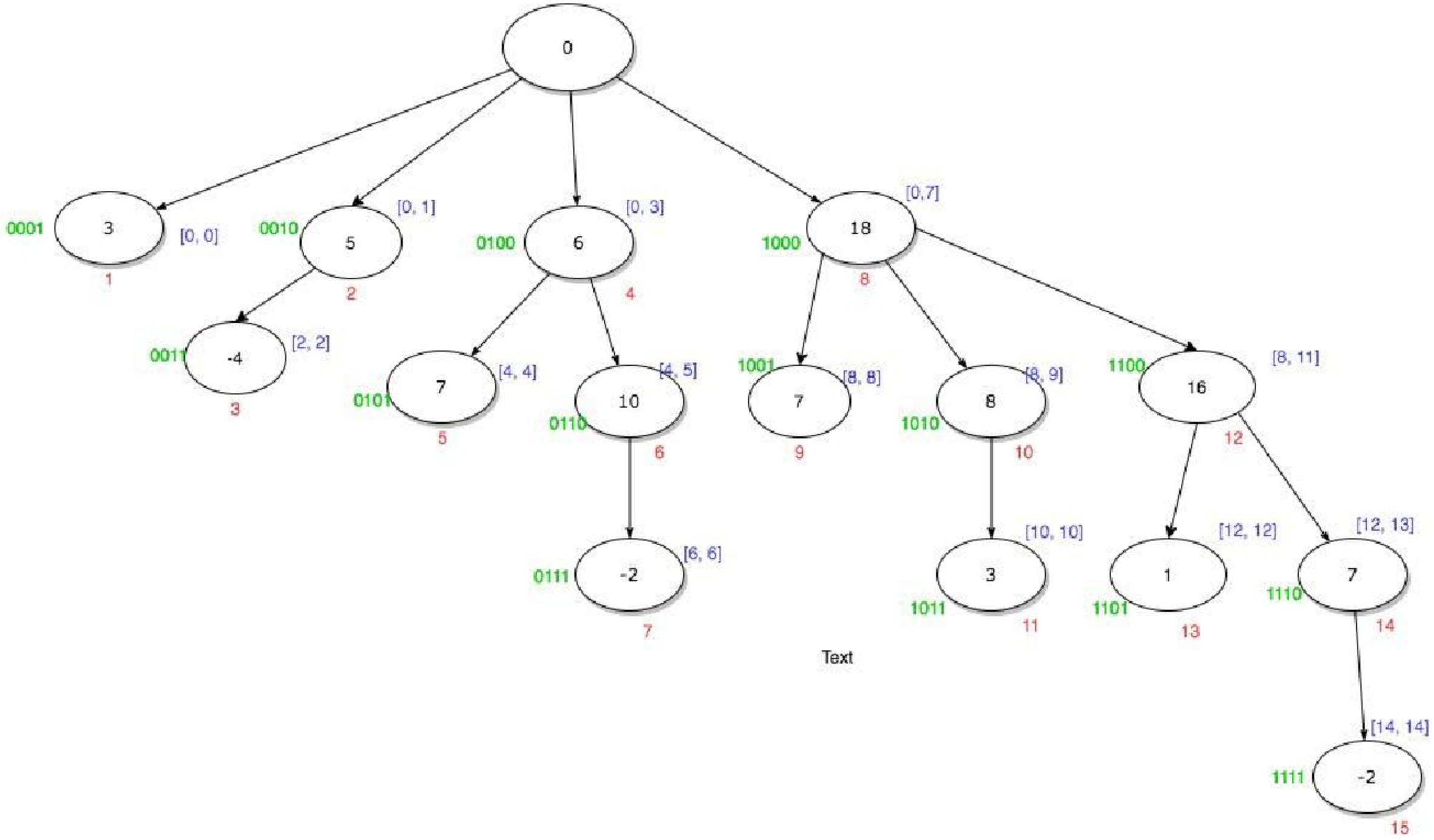


$1 = 2^0 + 0$
$2 = 2^1 + 0$
$3 = 2^1 + 2^0 + 0$
$4 = 2^2 + 0$
$5 = 2^2 + 2^0 + 0$
$6 = 2^2 + 2^1 + 0$
$7 = 2^2 + 2^1 + 2^0 + 0$
$8 = 2^3 + 0$
$9 = 2^3 + 2^0 + 0$
$10 = 2^3 + 2^1 + 0$
$11 = 2^3 + 2^1 + 2^0 + 0$
$12 = 2^3 + 2^2 + 0$
$13 = 2^3 + 2^2 + 2^0 + 0$
$14 = 2^3 + 2^2 + 2^1 + 0$
$15 = 2^3 + 2^2 + 2^1 + 2^0 + 0$

number	binary representation	range of responsibility
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	



$1 = 2^0 + 0$
$2 = 2^1 + 0$
$3 = 2^1 + 2^0 + 0$
$4 = 2^2 + 0$
$5 = 2^2 + 2^0 + 0$
$6 = 2^2 + 2^1 + 0$
$7 = 2^2 + 2^1 + 2^0 + 0$
$8 = 2^3 + 0$
$9 = 2^3 + 2^0 + 0$
$10 = 2^3 + 2^1 + 0$
$11 = 2^3 + 2^1 + 2^0 + 0$
$12 = 2^3 + 2^2 + 0$
$13 = 2^3 + 2^2 + 2^0 + 0$
$14 = 2^3 + 2^2 + 2^1 + 0$
$15 = 2^3 + 2^2 + 2^1 + 2^0 + 0$


0
0001
3
[0, 0]
1
0010
5
[0, 1]
2
0011
-4
[2, 2]
3
0100
6
[0, 3]
4
0101
7
[4, 4]
5
0110
10
[4, 5]
6
0111
-2
[6, 6]
7
1000
18
[0,7]
8
1001
7
[8, 8]
9
1010
8
[8, 9]
10
1011
3
[10, 10]
11
1100
16
[8, 11]
12
1101
1
[12, 12]
13
1110
7
[12, 13]
14
1111
-2
[14, 14]
15

Text

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

Query(11) = a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] +a[11]

number	binary represent
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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

Query(11) = a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] +a[11]

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

number	binary represent
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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

Query(11) = a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] +a[11]

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

10 => (1010)
last bit = 2^1 = 2
tree[10] = a[10]+a[9]
Subtract the last on bit;

```
 1010
-0010
------
 1000 => 8
```

number	binary represent
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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

Query(11) = a[1] + a[2] + a[3] + a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10] +a[11]

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

10 => (1010)
last bit = 2^1 = 2
tree[10] = a[10]+a[9]
Subtract the last on bit;

```
 1010
-0010
------
 1000 => 8
```

8 => (1000)
last bit = 2^3 = 8
tree[8] = a[8]+a[7]+...+a[1]
Subtract the last on bit;

```
 1000
-1000
------
 0000 => 0
```

number	binary represent
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

```
int  qry(int idx)
{
    int  ret = 0;
    while(idx > 0){
        ret += bit_tree[idx];
        idx -= (idx&-idx);
    }
    return ret;
}
```

Query(11) = a[1] + a[2] + a[3] + a[4] + ... + a[9] + a[10] + a[11]
ret=bit_tree[11]+bit_tree[10]+bit_tree[8]

idx & -idx = 11 & (-11)

idx = 11 => 1011
-idx= -11=> 0101 (2's compliment)
.....
idx & -idx = 0001
idx - = (idx & -idx)
idx=11-1;
idx=10

How to find 2's Compliment ?

11 = > 1011
invert=> 0100
+ 1 => 0001
.....
2's com=> 0101

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

10 => (1010)
last bit = 2^1 = 2
tree[10] = a[10]+a[9]
Subtract the last on bit;

```
 1010
-0010
------
 1000 => 8
```

8 => (1000)
last bit = 2^3 = 9
tree[8] = a[8]+a[7]+...+a[1]
Subtract the last on bit;

```
 1000
-1000
------
 0000 => 0
```

number	binary represent
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

```
int  qry(int idx)
{
    int  ret = 0;
    while(idx > 0){
        ret += bit_tree[idx];
        idx -= (idx&-idx);
    }
    return ret;
}
```

Query(11) = a[1] + a[2] + a[3] + a[4] + ... + a[9] + a[10] + a[11]
ret=bit_tree[11]+bit_tree[10]+bit_tree[8]

idx & -idx = 10 & (-10)

idx = 10 => 1011
-idx= -10=> 0110 (2's compliment)
.....
idx & -idx = 0010 => 2
idx - = (idx & -idx)
idx=10-2;
idx=8

How to find 2's Compliment ?

10 = > 1010
invert=> 0101
+ 1 => 0001
.....
2's com=> 0110

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

10 => (1010)
last bit = 2^1 = 2
tree[10] = a[10]+a[9]
Subtract the last on bit;

```
 1010
-0010
------
 1000 => 8
```

8 => (1000)
last bit = 2^3 = 9
tree[8] = a[8]+a[7]+...+a[1]
Subtract the last on bit;

```
 1000
-1000
------
 0000 => 0
```

number	binary represent
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

```
int  qry(int idx)
{
    int  ret = 0;
    while(idx > 0){
        ret += bit_tree[idx];
        idx -= (idx&-idx);
    }
    return ret;
}
```

Query(11) = a[1] + a[2] + a[3] + a[4] + ... + a[9] + a[10] + a[11]
ret=bit_tree[11]+bit_tree[10]+bit_tree[8]

idx & -idx = 8 & (-8)

idx = 8 => 1000
-idx= -8=> 1110 (2's compliment)
.....
idx & -idx = 1000 => 8
idx - = (idx & -idx)
idx=8-8;
idx=0

How to find 2's Compliment ?

8 = > 1000
invert=> 0111
+ 1 => 0001
.....
2's com=> 1110

11 => (1011)
last bit = 2^0 = 1
So, tree[11] = a[11]
Subtract the last on bit;

```
 1011
-0001
------
 1010 => 10
```

10 => (1010)
last bit = 2^1 = 2
tree[10] = a[10]+a[9]
Subtract the last on bit;

```
 1010
-0010
------
 1000 => 8
```

8 => (1000)
last bit = 2^3 = 9
tree[8] = a[8]+a[7]+...+a[1]
Subtract the last on bit;

```
 1000
-1000
------
 0000 => 0
```

number	binary represent
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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Table 1.1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

Table 1.2 – table of responsibility

Update(idx,val) = Update(5,100) => a[5] = a[5] + 100 ;

a[5] is contained in ,
=> tree[5], tree[6], tree[8], tree[16]
So, To update a[5] , We have to update this 4 tree values.

number	binary represent
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

```
void update(int idx,int  val)
{
    while(idx < M){
        bit[idx] += val;
        idx += (idx&-idx);
    }
}
```

Update(idx,val) = Update(5,100) => a[5] = a[5] + 100 ;

idx=5;
tree[5]+=100

a[5] is contained in ,
=> tree[5], tree[6], tree[8], tree[16]
So, To update a[5] , We have to update this 4 tree values

idx & -idx = 5 & (-5)

idx = 5 => 0101
-idx= -5=> 1011 (2's compliment)
.....
idx & -idx = 0001 => 1
idx + = (idx & -idx)
idx=5+1;
idx=6

How to find 2's Compliment ?

5 = > 0101
invert=> 1010
+ 1 => 0001
.....
2's com=> 1011

number	binary represent
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

```
void update(int idx,int  val)
{
    while(idx < M){
        bit[idx] += val;
        idx += (idx&-idx);
    }
}
```

idx=6;
tree[6]+=100

Update(idx,val) = Update(5,100) => a[5] = a[5] + 100 ;

a[5] is contained in ,
=> tree[5], tree[6], tree[8], tree[16]
So, To update a[5] , We have to update this 4 tree values

idx & -idx = 6 & (-6)

idx = 6 => 0110
-idx= -6=> 1010 (2's compliment)

.....

idx & -idx = 0010 => 2
idx + = (idx & -idx)
idx=6+2;
idx=8

How to find 2's Compliment ?

6 = > 0110
invert=> 1001
+ 1 => 0001

.....

2's com=> 1010

number	binary represent
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

```
void update(int idx,int  val)
{
    while(idx < M){
        bit[idx] += val;
        idx += (idx&-idx);
    }
}
```

Update(idx,val) = Update(5,100) => a[5] = a[5] + 100 ;

idx=8;
tree[8]+=100

a[5] is contained in ,
=> tree[5], tree[6], tree[8], tree[16]
So, To update a[5] , We have to update this 4 tree values

idx & -idx = 8 & (-8)

idx = 8 => 1000
-idx= -8=> 1000 (2's compliment)
.....
idx & -idx = 1000 => 8
idx + = (idx & -idx)
idx=8+8;
idx=16;

How to find 2's Compliment ?

8 = > 1000
invert=> 0111
+ 1 => 0001
.....
2's com=> 1000

number	binary represent
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

```
void update(int idx,int  val)
{
    while(idx < M){
        bit[idx] += val;
        idx += (idx&-idx);
    }
}
```

Update(idx,val) = Update(5,100) => a[5] = a[5] + 100 ;

idx=16;
tree[16]+=100

a[5] is contained in ,
=> tree[5], tree[6], tree[8], tree[16]
So, To update a[5] , We have to update this 4 tree values

idx & -idx = 16 & (-16)

idx = 16 => 10000
-idx= -16=> 10000 (2's compliment)
.....
idx & -idx = 10000 => 16
idx + = (idx & -idx)
idx=16+16;
idx=32;

How to find 2's Compliment ?

16 = > 10000
invert=> 01111
+ 1 => 00001
.....
2's com=> 10000

number	binary represent
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