



As we can
have at max $\log_2 n$
bits is a number
thus loop will go
 $\log_2 n$ times.

```

int query ( int id )
{
    int ans = 0 ;
    while ( id > 0 ) {
        ans += bit [ id ]
        id -= ( id & -id )
    }

    return ans ;
}

```

Query sum from

$$\begin{array}{rcl}
 \text{id} & & 0111 \\
 \& -\text{id} & \Rightarrow \& 1001 \\
 \hline
 \text{id} \& -\text{id} & \underline{0001}
 \end{array}$$

$$\begin{array}{rcl}
 \text{id} \checkmark & = & 0111 \\
 \text{---} (\text{id} \& -\text{id}) & = & \underline{-0001} \\
 \hline
 \text{id} - (\text{id} \& -\text{id}) & = & 0110
 \end{array}$$

Flipped last

So just rightmost set bit is flipped

```
int query(int id){
    int ans = 0;
    while(id > 0){
        ans += bit[id];
        id -= (id & -id);
    }
    return ans;
}
```

```
// Always remember BIT is 1 based
```

```
vector<int> bit;
```

```
void update(int id, int val){
```

```
while(id <= n){
```

```
bit[id] += val;
```

```
id += (id & -id);
```

Now just
add it at
exponentially
increasing intervals

① $\frac{1}{5}$

```
vector<int> bit; // BIT array

vector<int> a; // Original input array

void update(int id, int val) {
    while (id < bit.size()) {
        bit[id] += val;

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

int query(int id) {
    int ans = 0;

    while (id > 0) {
        ans += bit[id];

        id -= (id & -id);
    }
}
```

```

return ans;
}

int main() {
    int n, q;
    cin >> n;

    bit = vector<int>(n + 1, 0);

    a = vector<int>(n + 1);

    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        update(i, a[i]);
    }

    cin >> q;

    while (q--) {
        int typeOfQuery;
        cin >> typeOfQuery;

        if (typeOfQuery == 1) {
            int L, R;
            cin >> L >> R;

            int ans = query(R) - query(L - 1);

            cout << ans << endl;
        } else {
            int id, val;
            cin >> id >> val;

            update(id, -a[id]); // Firstly make it back to 0
            a[id] = val; // Update your original input array 'a'
            update(id, a[id]); // Update the BIT array
        }
    }

    return 0;
}

```

<https://leetcode.com/problems/range-sum-query-mutable/>