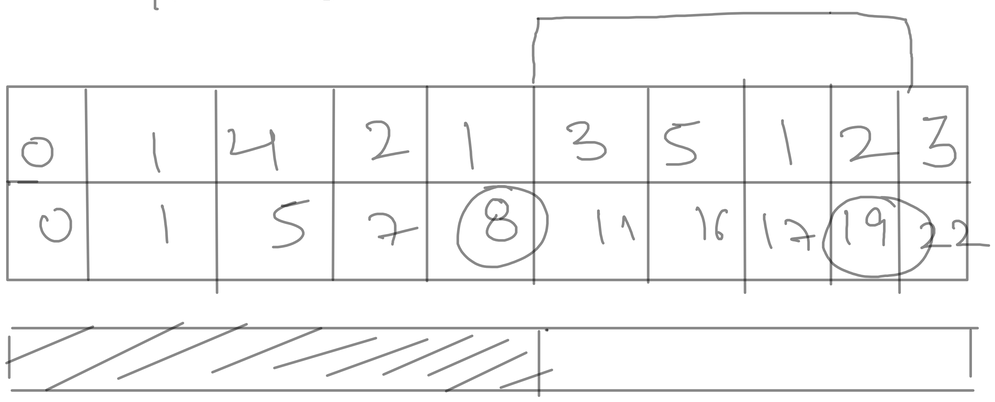
# Fenwick Tree
(BIT)

Given some sequence
+
qieies determine range sum [a,b]

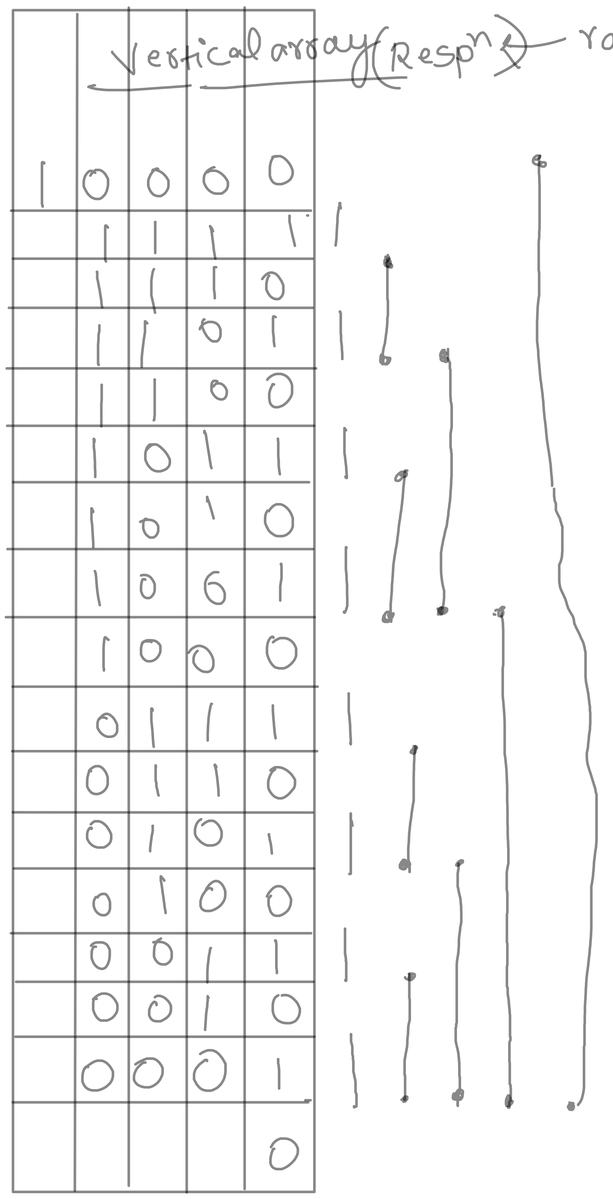



Normal method → O(N)

Using prefix Sum → O(1)

Dynamic
- O(1) Query time
- O(n) recompute time.

⇒ Thus, Fenwick tree.




* F.T is organized by lowest one bit. (range of responsibility)

## prefix Sums

Ex: Prefix Sum till 11th index.

1011 → V[11]
↳ 1010 → V[10]
↳ 1000 → V[8]
↳ 0000 → V[0]
↑
as soln 1 based, it's useless.

⇒ pre[11] = V[11] + V[10] + V[8];

Steps
— Start at current cell;
— Staircase down until no bits are left;
— i.e substract lowest one bit each time to get to next index in F.T (To move down)

## runtime?

— loop runs in at most making zero of all set bits
— No of bits are O(log n)
— complity = O(log n)

## update in fenwick tree:

For Ex:
Δ is change = Vnew - Vold.
arr[9] is changed by 'Δ'
→ V[9] will also change and also all the V[ ] in tree that took responsibity of '9'

→ 1001
1010 [look in tabl
1100
10000

These are nodes in F-T resp for '9'. So we need to add Δ to them.

→ how to find? (To move up)

Soln add '1' to lowest 1 bit.

1001 ✓
+ 1
= 1010 ✓
+ 10
1100 ✓
+1
10000 ✓

Find lwst bit.
i & -i

Ex:
i = 1010
-i ⇒ 0101 +1
-i = 0110
(i & -i)
= (1010) (0110)
= (0010) ✓

— must update all the owners of us.
— move by adding lowest one bit to current index.
— loop runs exactly as number of 1's bits.
— O(log n) runtime bounded.

# CODE TIME

```
vector<int>t(N,0);
void bit_update( int i, int delta, int size) {
        while(i <= size) {
            t[i] += delta;
            i += (i & -i);
        }
}
int pre_sum (int i ) {
    int sum = 0;
    while ( i > 0 ) {
        sum += t[i];
        i -= (i & -i);
    }
    return sum ;
}

int range_sum(int l, int r) {
    return pre_sum(r) - pre_sum(l - 1);
}



void solve() {
    int n; cin >> n;
    vector<int> a(n+1,0);
    for(int i = 1; i<=n; i++ ) {
        cin >> a[i];
        bit_update(i, a[i], n);
    }
```

by how much value changes.

keep in mind to use <= and not <.

if ( l > r) return 0 ;

(This is the best part of code.)

# Questions Based on BIT

(2-1)=1

① Counting Inversions. (Also try with Merge sort)

permutation

7 6 2 3 1 4 5

↑ ↑

(6,1) is Inversion.

7 6 2 3 1 4 5

6 5 1 1 0 0 0

(update)

1 1

6 1 ← (query)

5

4

3

2

1

7 6 2 3 1 4 5

↑ Sum=

how many values are greater than a[i], before i.

for 6 → 1

for 2 → 2

for 3 → 2

for 1 → 4

for 4 → 2

for (5) → 2

+ ans

MII.

7 6 2 3 1 4 5

How many no. ahead of me are smaller than me.

7

6

5 1

4 1

3 1

2

1 1

⟩⟩ implementation

Q. $i<j<k$, $a[i]<a[j]<a[k]$ find s.c triplets.

hint we 2 BIT.

## Q2. Find $K^{th}$ tallest :

$h \rightarrow [1, 10^6]$

update : changes no of people of a particular he.

Quer: Find $K^{th}$ tallest $K [1, 10^{18}]$

Ideas :

K → 1 2 3 4 5 6

f → 8 2 10 100 1 2

Pre → 8 10 20 120 121 123

Ex: (K=60), b = 122 ✓

So, i can maintain, a variable prefix sum with BIT, and my query using binary search.

So complexity → $(\log n)(\log n) = (\log n)^2$

B.S ↑ BIT ↑

Space complexity → O(N)

Can we do faster?

Yes we can:

By doing binary search directly on BIT: find?

For Ex: (sum = 20)

9 1
10 2
11 3
12 4
→ 8 25
7 2
6 ~~7~~
5 3
→ 4 16
3 6
2 ~~20~~
1 8

keep starting pointer at max power of 2 : i.e 8 (ans)

⇒ We can see that the value in larger. So we should go to next smaller power of 2 ⇒ 4

* ~~value~~ t[4] < 20, So this needs to included in our found sum.

Sm += 16

* Now we need to reach above it.

* i 4+2 = 6 ↑ next power of 2

* check t[6] → 7, ~~16~~ 7+7 > 20 so, No

4+1 = 5

t[5] → 3, 16+3 < 20 ✓

(pos = 5, sm = 19)

* Now we are finished with all powers of 2, so we can stop;

index with value less than sum is pos, with sum as sm.

## Code:

```
int bS(int n, int sum) {
      // pos < val;
       int sm = 0, pos = 0;
       for ( int i = (int)log2(n); i >= 0; i--) {
             if ( ( pos + (1 << i) )  < n  && sm + t[pos + (1 << i)] < sum ) {
                   sm += t[pos + (1 << i) ];
                   pos += (1 << i );
             }
       }
             return pos+1;

}
```

Link to this : https://codeforces.com/blog/entry/61364 and this https://codeforces.com/blog/entry/11275?locale=en

https://codeforces.com/contest/992/problem/E