Segment Tree

Given array of N nodes.

query: find sum L–R

TC: $O(\log N)$

SC: $O(N)$

Update: $O(\log N)$

Segment tree of type of query.

Sum, Min, Max, Gcd, Lcm

TC: $O(\log N)$

SC: $O(N)$

Array L–R, find sum or min

Hint → $N/2$

$f(x) = 2x + N/2$

$\frac{f(x)}{dx} = 2 - \frac{N}{x^2} = 0$

$x^2 = \frac{N}{2}$

$x = \sqrt{\frac{N}{2}}$

$x \to$ Hidden $\to \sqrt{N}$

$f(\sqrt{N}) = 2\sqrt{N} + \frac{N}{\sqrt{N}} = 3\sqrt{N}$

TC for query $\to O(\sqrt{N})$

$N! \to N^N$

$\sqrt{N} \to N^{1/2}$

TC: $O(\text{query}\ \sqrt{N} + N)$

SQUARE ROOT DECOMPOSITION

Segment Tree (Binary tree)

→ help you in any type query problem.

Array: [5, 2, 3, 4, 1, 7, 8, 9, 6, 3, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

$\sqrt{N} = 4$

$\log^2 N$

Query tree

query → (L, R) → Segment [S, E]

Segment Overlap

1) Complete Overlap

L<=S && R>=E

S ... E

2) No Overlap

L>E || R<S

S ... E

3) Partial Overlap

L<=S && R<E

S ... E

→ ans = (query left, query right)

ans = sum for any of one no.

$10 + 2 + 3 + \dots + N = \frac{N(N+1)}{2}$

$1 + 2 + 3 + \dots + N = \frac{N(N+1)}{2}$

$N = 2^{\log_2 N}$

$\frac{N(N+1)}{2}$

$\frac{N(N+1)}{2} \to N^2$

ans → $O(N)$

So if A is any → 4N

Height of a segment tree: $O(\log_2 N)$

Analysis – TC for Segment

TC

- Per query → $O(\log_2 N)$
- Update query → $O(\log_2 N)$
- Creating ST → $O(N)$

SC

$O(N + 2N + 4N)$

$N + 2N + 4N$

$7N$

7×10^5

10^6

7×10^6

10^7

Code:

1) Building → ST

void build (int, int, int) {

if (s == e) {

T[ind] = A[s];

}

TC: $O(N)$

m = (s+e)/2;

Left = 2*ind;

Right = 2*(ind+1);

build (left, s, m);

build (right, m+1, e);

T[ind] = min (T[left], T[right]);

}

2) Query → Min

int query (int, int, int, int) {

if (R<=S || L>=E) // ans

int m = (S+E)/2;

if (L<=S && R>=E)

int ans = T[ind];

TC: $O(\log_2 N)$

m = (S+E)/2;

Left = 2*ind;

Right = 2*(ind+1);

ql = query (left, S, m, L, R);

qr = query (right, m+1, E, L, R);

int ans = min (ql, qr);

}

3) Update

void update (int, int, int, int, int) {

if (s == e) {

T[ind] = val;

A[s] = val;

}

TC: $O(\log_2 N)$

m = (s+e)/2;

Left = 2*ind;

Right = 2*(ind+1);

if (ind <= m) {

update (left, s, m, ind, val);

}

else {

update (right, m+1, e, ind, val);

}

T[ind] = min (T[left], T[right]);

}

Update(1, 1, N, ind, val);

query (1, 1, N, L, R);

build (1, 1, N);

Sum, min, max, xor, GCD

Prime

$\sqrt{N}$