Segment Trees

Arthur Conmy[*][†]

April 2022

Think about how to solve one of the below problems (perhaps for smaller bounds). We will discuss an approach able to solve both problems.

Problem (*Dynamic Range Sum Queries*)
Given an array of n integers $x_1 x_2 ... x_n$, process q queries of the following types:

1. Update the value at position k to u.
2. Output the sum of values in the range $[a, b]$.

The bounds are $n, q \leq 2 \times 10^5$ and $x_i, u \leq 10^9$.

Problem (*LCAs in a Tree*)
You are given a tree of n vertices, rooted at vertex 1. Given q queries `u v`, for each query output the lowest common ancestor. Again, $n, q \leq 2 \times 10^5$

The lowest common ancestor of two vertices u and v is the lowest vertex that's on the path from 1 to u and from 1 to v:

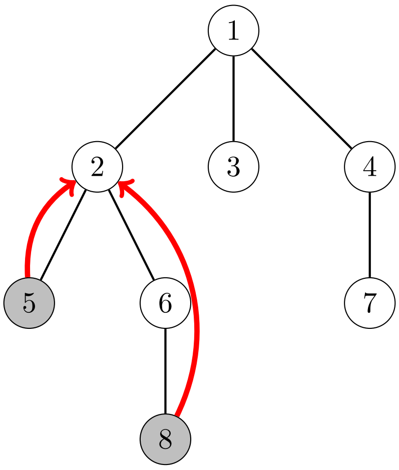


Figure 1: The LCA of vertices 5 and 8 is vertex 2.

[*] `asc70@cam.ac.uk` for feedback or help or anything.

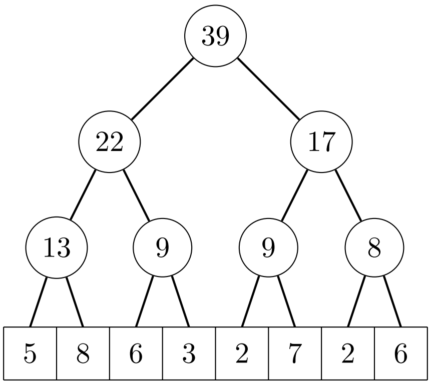
[†] This is based on *Competitive Programmer's Handbook*, https://cses.fi/book/book.pdf, Antti Laaksonen.

Segment Trees

A segment tree is a binary tree such that the nodes on the bottom level of the tree correspond to the array elements, and the other nodes contain information needed for processing range queries[1].

0	1	2	3	4	5	6	7
5	8	6	3	2	7	2	6

The corresponding segment tree is as follows:



i.e, each vertex has value the sum of the two vertices below it in this case.

Question. Can you see how to perform the range and the sum queries given the data structure of this form?

You should be able to see a way to perform both range queries and updates in $O(\log n)$. For the Dynamic Range Queries problem this leads to an $O(n \log n)$ solution! Segment trees often allow us to support both updating and querying efficiently, rather than just one of them.

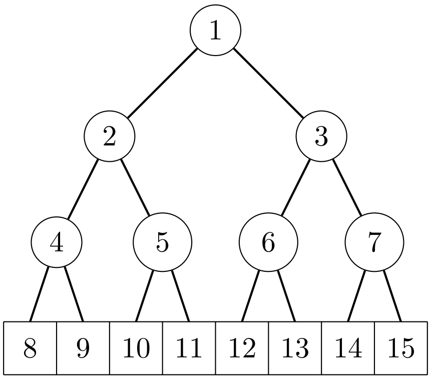
Implementation

I'd recommend sticking quite close to the following segment tree implementation, perhaps even memorising the `query` function[2].

We store the segment tree as an array `tree` of length $2n$, where the array is kept in the array `tree[n], tree[n+1], ..., tree[2*n - 1]`, and the indices correspond to the binary tree vertices as shown:

[1] In this section, we assume that the size of the array is a power of two, because it is convenient to build a segment tree for such an array. You might want to think about how we could use a power of two segment tree when our array doesn't have length that's a power of two.

[2] It turns out a more complicated implementation is needed to allow for both range and point updates and queries to be supported; see 'lazy propagation' in **?**. I don't memorise this implementation, but rewrite it, and it takes a fair bit longer for me.



This has the very nice propery that where defined, vertex i has children $2i$ (which is even) to the left and $2i + 1$ (which is odd) to the right, and parent $\lfloor i/2 \rfloor$.

Question. How can we implement the function `void update(int index, int new_value)` that updates a value in our sum segment tree?

It's slightly trickier to implement the sum function[5]:

```
int sum(int a, int b) {
    a += n; b += n;
    int s = 0;
    while (a <= b) {
        if (a%2 == 1) s += tree[a++];
        if (b%2 == 0) s += tree[b--];
        a /= 2; b /= 2;
    }
    return s;
}
```

We can implement the range maximum query for the LCA problem by replacing the `s += tree[a--]` line with `s = max(s, tree[a--])` and similar for the line below.

Problems

Implement Dynamic Range Sum Queries first: https://cses.fi/problemset/task/1648. If you get stuck implementing any part of the segment tree, you should be able to find all the parts needed (and more explanation) in https://cses.fi/book/book.pdf.

Problem. 1: https://cses.fi/problemset/task/1651

Problem. 2: https://cses.fi/problemset/task/1749

Problem. 3: https://cses.fi/problemset/task/1143

Problem. 4: https://cses.fi/problemset/task/1144

Problem. 5: https://codeforces.com/contest/1195/problem/F

Problem. 6: https://codeforces.com/contest/1467/problem/E

Problem. 7: https://oj.uz/problem/view/IOI13_game http://www.ioi2013.org/wp-content/uploads/tasks/day2/game/game.pdf

[5] I commited this to memory at first, but so long as you've got the basic structure the checks modulo 2 and whether to increment or decrement should be rederivable, as I can go through.