Topic 3 : Range Query

Segment Tree Hard

Why is segment tree so hard?

[Problem Statement $\xrightarrow{\text{observation}}$ Rough Solution $\xrightarrow{\text{Set up segtree}}$ Segtree Solution]$_{\text{Observations}}$ [Segtree Solution $\xrightarrow{\text{code}}$ Solution $\xrightarrow{\text{submit}}$]$_{\text{Knowledge}}$

(1) People don't have good obsersevation

(2) People don't know enough segtree knowledge

(3) People don't know how to code

Let us address one key misconception first...

Warm-up problem: Movie Collection

Warm-up problem: Movie Collection

Knowing what to do with numbers is certainly the heart and soul of basic arithmetic. However, knowing what to do can mean rather different things. I have always been charmed by this example from an elementary school child reported a number of years ago:

I know what to do by looking at the examples. If there are only two numbers I subtract. If there are lots of numbers I add. If there are just two numbers and one is smaller than the other it is a hard problem. I divide to see if it comes out even and if it doesn' t I multiply.

Perkins, David N., Making Learning Whole: How Seven Principles of Teaching Can Transform Education

Misconception: Where is the segtree?

Observational things are good to ask:

DP: Is there some optimal substructure?
Greedy: Is there some optimal strategy?
Guessing: Is there some pattern in the answers?

These questions help you establish frameworks

Meanwhile, specific pieces of code is not a good starting point
Should I use a std::set to solve this problem?
Should I use a priority queue to ...
or

Where is the segtree?

(Do this AFTER you have a rough solution)

Tacit Knowledge : Rough Solution

Generally know what you can do

e.g. find the maximum in a bunch of numbers
sum
. . .

Do some trivial geo computation

Find some trivial combinitorial fomula

this class!! Do some trivial range query

And describe your solution in these functionalities

(We should practice this in code presentation)

Range Query (Expand your vocab on rough sols)

Ex. 1: $S[l:r] = a_l + a_{l+1} + \dots + a_{r-1}$ (sum)

(1) Combine: $S[l:m] + S[m:r] = S[l:r]$
(2) Separate: $S[l:r] - S[l:m] = S[m:r]$

$S[l:r]$

(1) Static (Find result) — Prefix Sum

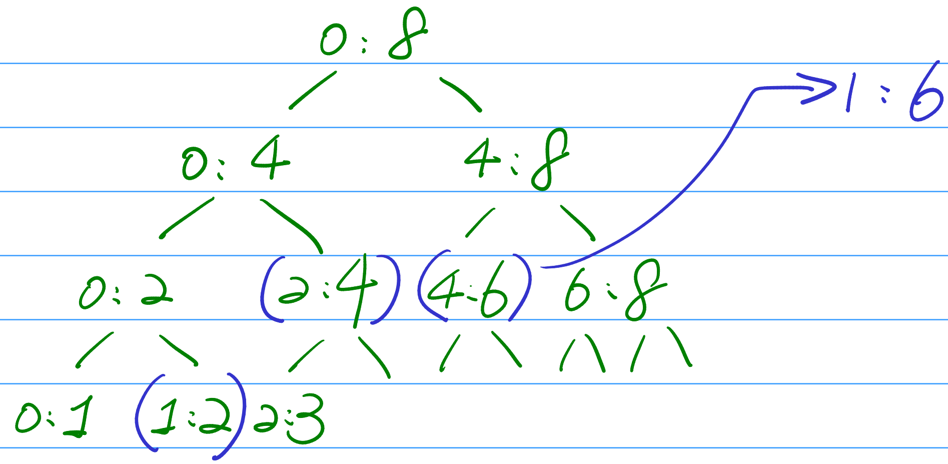
(2) Point Modify ($a_i \leftarrow v$) — Fenwick Tree

(3) Range Modify but Point Query — Fenwick Tree

(4) Range Modify & Range Query — (Segment Tree)

Only thing

Segtree w/ Lazy tag in 1 Slide



Theorem: Any range l:r can be represented with O(log n) nodes on the segtree

Therefore, when we do a[l:r] += v, we only need to tag O(log n) nodes lazily.

We can alway "push down" a tag to its two children with O(1) complexity.

Check the code sample on the website and code one yourself.

Some implementation details

$$\begin{pmatrix} l:r \\ S[l:r] \\ lazy \end{pmatrix}$$

$S[l:r]$
= original value +
lazy tag value
(it's always accurate
when visited)

```
add { push-down(l:r)
  if (l:r is in add range)
      S[l:r] += d*(r-l)
      lazy += d
      return
  do left
  do right
  update S[l:r]
}
```

```
push-down {
   if  l:r is not a leaf
       S[l:m] += lazy*(m-l)
       S[m:r] += lazy*(r-m)
       lazy[l:m] += lazy
       lazy[m:r] += lazy
   lazy = 0
}
```

Beyond Addition...

Ex. 2 : $\max[l:r] = \max(a_l, \ldots, a_{r-1})$

(1) Combine : $\max[l:r] = \max(\max[l:m], \max[m:r])$

(2) No Separate : $\max[l:r] - \max[l:m] \neq \max[m:r]$

	$S[l:r]$	$\max[l:r]$
(1) Static (Find result)	Prefix Sum	RMQ (Sparse Table)
(2) Point Modify ($a_i \leftarrow v$)	Fenwick Tree	Segment Tree
(3) Range Modify but Point Query	Fenwick Tree	"
(4) Range Modify & Range Query	Segment Tree	"

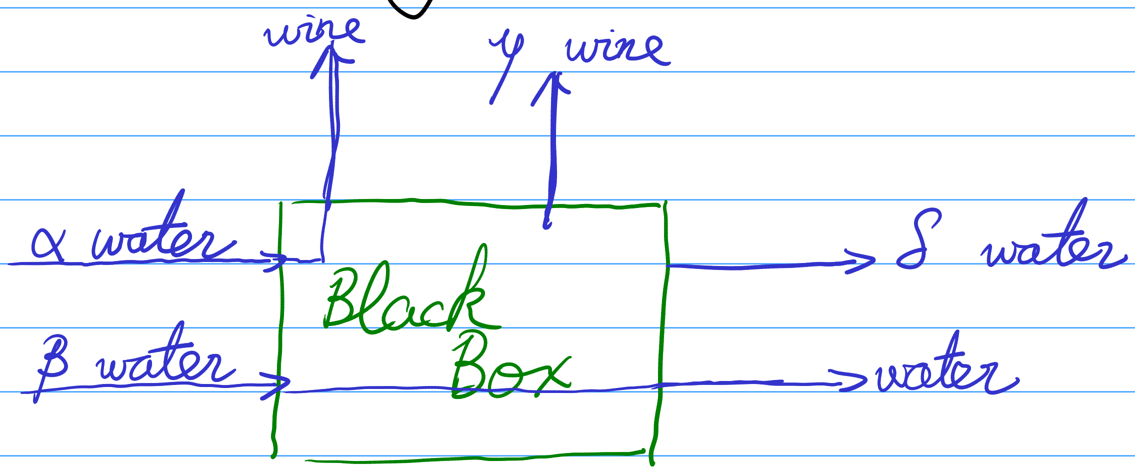
Combinable is key ☺

Wine Factory (Hard Version)

Don't ask where segtree is !

Try to understand what can be combined.

Wine Factory (Hard Version)



Observation: A list of factories is a black box

Black boxes are combinable

Editor

Paimon Segment Tree

Step 1: Observe that

$$\begin{aligned} &Query(l_k, r_k, x_k, y_k) \\ &= Query(1, r_k, x_k, y_k) - \\ &\quad Query(1, l_{k-1}, x_k, y_k). \end{aligned}$$

(Prefix on time is enough)

Step 2: Build a segtree to maintain $S[l:r]$

Step 3: Denote $S^2[l:r] = a_l^2 + \cdots + a_{r-1}^2$

Observe that when you update $S^2[l:r]$ with $+d$

$$S^2[l:r] = S^2[l:r] + 2 \cdot d \cdot S[l:r] + (r-l) \cdot d^2$$

So you can maintain $S^2[l:r]$ as well.

Paimon Segment Tree

Step 4: Denote Query $[l:r] = \sum_{t=1}^{now} S^t[l:r]$

When you update Query $[l:r]$ with $+d$

Query $[l:r]$ = Query $[l:r]$ + now $S^t[l:r]$

Step 5: At one step, update Query $[1:l]$ with $+0$
Query $[r:n]$ with $+0$
Query $[l:r]$ with $+d$

Summary

	$S[l:r]$	$\max[l:r]$
(1) Static (Find result)	Prefix Sum	RMQ (Sparse Table)
(2) Point Modify ($a_i \leftarrow v$)	Fenwick Tree	Segment Tree
(3) Range Modify but Point Query	Fenwick Tree	"
(4) Range Modify & Range Query	Segment Tree	"

Wine Factory : what state is maintainable
Editor : how to observe before segtree
Paimon : how to build segtree state from state (Advanced)

No more "Where is segtree" !!!