Interval Trees

Set $S$ of intervals

$4 \rightarrow 8 \rightarrow 11 \rightarrow 15 \rightarrow 18 \rightarrow 21 \rightarrow 23$
set $S$ of intervals

\begin{align*}
&4 \quad 8 \\
&5 \quad 11
\end{align*}

\begin{align*}
&7 \quad 10 \\
&15 \quad 18 \\
&17 \quad 19 \\
&21 \quad 23
\end{align*}

\begin{align*}
&x \\
&a \quad b
\end{align*}

\begin{align*}
&lo[x] = a \\
&hi[x] = b
\end{align*}
INTERVAL TREES

set $S$ of intervals

$\{ 4 \longrightarrow 8, 5 \longrightarrow 11, 7 \longrightarrow 10, 15 \longrightarrow 18, 17 \longrightarrow 19, 21 \longrightarrow 23 \}$

Query: given an interval $x$, return any interval in the set $S$ that partially overlaps $x$ (if one exists)

$lo(x) = a$

$h_i(x) = b$
types of overlap:
1) "smaller"

\[ \text{lo}[x] \quad \times \quad \text{hi}[x] \]

\[ s_i \]
types of overlap:
1) "smaller"
2) "bigger"
Types of overlap:
1) "smaller"
2) "bigger"
3) "left" & "right"
types of overlap:
1) "smaller"
2) "bigger"
3) "left" & "right"

First comparison: \(lo[s_i]\) vs \(lo[x]\)
types of overlap:
1) "smaller"
2) "bigger"
3) "left" & "right"

First comparison: $lo[s_i] \text{ vs } lo[x]$

is there some large enough $hi[s_i]$?
types of overlap:
1) “smaller”
2) “bigger”
3) “left” & “right”

First comparison: \(lo[s_i] vs lo[x]\)

is there some large enough \(hi[s_i]\) ?

\[\text{if } lo[s_i] < lo[x] \text{ AND } hi[s_i] > lo[x] \text{ then overlap}\]
types of overlap:
1) "smaller"
2) "bigger"
3) "left" & "right"

First comparison: $lo[s_i]$ vs $lo[x]$

is there some small enough $lo[s_i]$?

If $lo[s_i] > lo[x]$ AND $\leq hi[x]$
then overlap
types of overlap:
1) "smaller"
2) "bigger"
3) "left" & "right"

First comparison: $\text{lo}[s_i] \, vs \, \text{lo}[x]$

- is there some large enough $\text{hi}[s_i]$?
- is there some small enough $\text{lo}[s_i]$?

If $\text{lo}[s_i] < \text{lo}[x]$ AND $\text{hi}[s_i] > \text{lo}[x]$ then overlap

If $\text{lo}[s_i] > \text{lo}[x]$ AND $\leq \text{hi}[x]$ then overlap
SEARCHING FOR OVERLAPPING INTERVALS

ID:

BST w/ LEFT ENDS as KEYS
SEARCHING FOR OVERLAPPING INTERVALS

BST w/ LEFT ENDS as KEYS

MAX RIGHT END OF SUBTREE
SEARCHING FOR OVERLAPPING INTERVALS

**1D:**

```

```

calculate w/ root first

query segment
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

\( R < x \)

CASE 1
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

right subtree can't overlap

case 1

$L < x < W$

$R < x < W$
SEARCHING FOR OVERLAPPING INTERVALS

IF NO OVERLAP
right subtree can't overlap

R < x < W

keep searching LEFT

case 1
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

L case 2 R
SEARCHING FOR OVERLAPPING INTERVALS

IF \( Z \gg L \)

IF NO OVERLAP

\( L \) case 2 \( R \)

?
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF $z > l$
- search left

IF NO OVERLAP

$\exists \ y'$
- s.t. $y < l < z'$
  - guaranteed overlap

L case R
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF $Z \geq L$

search left

IF NO OVERLAP

L \quad \text{case 2} \quad R

\exists \ yz' \quad \text{s.t.} \quad y < l < z'

\{\text{guaranteed overlap}\}

\text{else} \ (z < l)
SEARCHING FOR OVERLAPPING INTERVALS

IF \( Z > L \) search left

IF NO OVERLAP

ELSE \( (Z < L) \)
  - NO overlap to left
  - search right
How can we update the MAX RIGHT END of a subtree?

BST w/ LEFT ENDS as KEYS
augmented BST

$$\max(t) = \max \left\{ h_i(t), \max(t_L), \max(t_R) \right\}$$
max1, max2, max3: unchanged by rotation
max(A) & max(B): trivial to update

we can maintain a balanced BST augmented w/ max value of subtrees