set $S$ of intervals

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

Set $S$ of intervals

$\lambda_0(x) = a$
$\lambda_i(x) = b$
Interval Trees

Set $S$ of intervals

Query: given an interval $x$,
return any interval in the set $S$ that partially overlaps $x$ (if one exists)
types of overlap:
1) "smaller"
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: $lo[s_i]$ 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 \( hi[s_i] \) ?

if \( lo[s_i] < lo[x] \)

and \( hi[s_i] > lo[x] \)

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 \( \text{lo}[x] \leq \text{hi}[x] \) then overlap
First comparison: \(lo[s_i] \text{ vs } lo[x]\)

- Is there some large enough \(hi[s_i]\)?
- Is there some small enough \(lo[s_i]\)?

If \(lo[s_i] \leq lo[x]\) AND \(hi[s_i] > lo[x]\) then overlap

If \(lo[s_i] > lo[x]\) AND \(\leq hi[x]\) then overlap

Store max of \(hi[s_i]\)

Store min of \(lo[s_i]\)
Store a balanced BST on \( lo[s_i] \)
Store a balanced BST on lo[si]

Compare lo[x] to nodes,
splits into 2 groups: < & >
Store a balanced BST on $lo[Si]$

Compare $lo[x]$ to nodes,

splits into 2 groups: $< \& >$
Store a balanced BST on $lo[s_j]$

Compare $lo[x]$ to nodes,
splits into 2 groups: $<$ & $>$

$$\text{if } lo[x] < lo[s_j] \quad (\text{and } hi[x] < lo[s_j]) \quad \Rightarrow \quad \frac{x}{s_j}$$
if \( \text{lo} \[x\] < \text{lo} \[s_j\] \) (and \( \text{hi} \[x\] < \text{lo} \[s_j\] \) )

Notice that \( \text{lo} \[s_j\] \leq \text{lo} \[s_i\] \) for the entire subtree of nodes to the right.

Store a balanced BST on \( \text{lo} \[s_i\] \).

Compare \( \text{lo} \[x\] \) to nodes, splits into 2 groups: < & >
Store a balanced BST on \( l0[i,j] \)

Compare \( l0[x] \) to nodes, \( x < k \)

splits into 2 groups: \( x < k \)

Notice that \( l0[i,j] \leq l0[i,j] \)

for the entire subtree of nodes to the right
Store a balanced BST on $lo[s_i]$

Compare $lo[x]$ to nodes, splits into 2 groups: $<$ & $>$

If $lo[x] < lo[s_j]$ (and $hi[x] < lo[s_j]$)

Notice that $lo[s_j] \leq lo[s_i]$ for the entire subtree of nodes to the right

If $lo[x] > lo[s_k]$ (and $lo[x] > hi[s_k]$)
Store a balanced BST on $lo[si]$

Compare $lo[x]$ to nodes, splits into 2 groups: $< & >$

- If $lo[x] < lo[s_j]$ (and $hi[x] < lo[s_j]$)
  - Notice that $lo[s_j] \leq lo[si]$ for the entire subtree of nodes to the right

- If $lo[x] > lo[sk]$ (and $lo[x] > hi[sk]$)
  - Compare $lo[x]$ to max of $hi[si]$ of the entire subtree of nodes to the left stored at $sk$ (augment)
Store a balanced BST on \( lo[si] \)

Compare \( lo[x] \) to nodes, splits into 2 groups: \(< \& >\)

if \( lo[x] < lo[sj] \) (and \( hi[x] < lo[sj] \))

Notice that \( lo[sj] \leq lo[si] \) for the entire subtree of nodes to the right

if \( lo[x] > lo[sk] \) (and \( lo[x] > hi[sk] \))

Compare \( lo[x] \) to \( \max \) of \( hi[si] \) of the entire subtree of nodes to the left stored at \( sk \) (augment)
Store a balanced BST on \(lo[si]\)

Compare \(lo[x]\) to nodes, splits into 2 groups: \(<\) & \(>\)

\[
\text{if } \quad lo[x] < lo[s_j] \quad \{ \quad \text{x} \quad \text{s_j} \\
(\text{and } hi[x] < lo[s_j]) \}
\]

Notice that \(lo[s_j] \leq lo[si]\)

for the entire subtree of nodes to the right

... go left.

\[
\text{if } \quad lo[x] > lo[Sk] \\
(\text{and } lo[x] > hi[Sk]) \quad \text{Sk}
\]

compare \(lo[x]\) to max of \(hi[si]\)

of the entire subtree of nodes to the left

stored at \(Sk\) (augment)

depends
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

ID:
SEARCHING FOR OVERLAPPING INTERVALS

1D:

compare w/ root first

query segment
SEARCHING FOR OVERLAPPING INTERVALS

**Case 1**

IF NO OVERLAP

\( R < x \)
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

right subtree can't overlap

case 1

R < x < W
SEARCHING FOR OVERLAPPING INTERVALS

ID:

IF NO OVERLAP
right subtree can't overlap

CASE 1

keep searching LEFT

R < x < W
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF NO OVERLAP

L case 2 R
SEARCHING FOR OVERLAPPING INTERVALS

IF $Z \gg L$

IF NO OVERLAP

CASE 2

L \quad R
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF \( Z \geq L \)

- search left

IF NO OVERLAP

\[ L \rightarrow \text{case 2} \rightarrow R \]

\[ \exists y, z' \]

s.t.

\[ y < L < z' \]

\{ guaranteed overlap \}
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF $z \geq l$
- search left

IF NO OVERLAP

CASE 2

$z$

E $yz'$

s.t. $y < l < z'$

\{ guaranteed overlap \}

else $(z < l)$
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF $Z \geq L$
- search left

ELSE $y\leq z'$
- guaranteed overlap

ELSE $(z < L)$
- no overlap to left
- search right
How can we update the max right end of subtree?
BST on $\{s_1\}$
BST

4 ——— 8 ——— 5 ——— 11 ——— 19 ——— 17 ——— 19

7 ——— 10 ——— 5 ——— 11 ——— 15 ——— 18 ——— 21 ——— 23
augmented BST

\[
\text{max}(t) = \max \left\{ h_i(t), \max(t_L), \max(t_R) \right\}
\]
\[ \text{max}1, \text{max}2, \text{max}3 : \text{unchanged by rotation} \]
\[ \text{max}(A) \ & \ \text{max}(B) : \text{trivial to update} \]

we can maintain a balanced BST augmented w/ max value of subtrees
Find-Overlap(x, S)

\[ y \leftarrow \text{root interval in } S \]

while \( y \neq \emptyset \) & \( \left(\frac{\text{lo}(x) > \text{hi}(y)}{\text{lo}(y) > \text{hi}(x)}\right) \) \} while no trivial solutions: (no overlap) -or- (x_y overlap)

\[ \ldots \]

return y
Find-Overlap(x, S)

\[
\begin{align*}
y &\leftarrow \text{root interval in } S \\
\text{while } y \neq \emptyset & \land \left( \begin{array}{c}
\text{no overlap} \\
\left| lo(x) \right| > hi(y) \\
\text{or} \\
\left| lo(y) \right| > hi(x)
\end{array} \right) \\
\text{return } y
\end{align*}
\]

\[\text{no } xy \text{ overlap} \quad \text{so keep searching} \]

\[\text{while no trivial solutions:} \quad \text{(no overlap)} - \text{or} - \text{(xy overlap)}\]
Find-Overlap(x, S)

y ← root interval in S

while y ≠ ∅ & \(\text{lo}(x) > \text{hi}(y)\) or \(\text{lo}(y) > \text{hi}(x)\) do

while no trivial solutions:

(\text{no overlap}) - OR - (x y overlap)

return y

For node p:
- left/right child = \(p_L, p_R\)
- if no child, use ∅
- \(\text{max}(p) = \text{max}(\text{hi}[s_i])\) in subtree(p)
Find-Overlap($x, S$)

\[ y \leftarrow \text{root interval in } S \]

while \( y \neq \emptyset \) & \( (\text{lo}(x) > \text{hi}(y) \text{ or } \text{lo}(y) > \text{hi}(x)) \)

if \( y_L \neq \emptyset \) & \( \text{lo}(x) \leq \max(y_L) \)

\[ y_L \text{ exists} \]

return \( y \)

For node \( p \): 
- left/right child = \( P_L, P_R \)
- if no child, use \( \emptyset \)
- \( \max(p) = \max(hi[S_i]) \) in subtree(p)

\[ \text{while no trivial solutions: } \]
\( (\text{no overlap}) - \text{OR-} (xy \text{ overlap}) \)

\[ \text{guaranteed overlap in left tree} \]

why?
Find-Overlap\((x, S)\)

\[ y \leftarrow \text{root interval in } S \]

while \( y \neq \emptyset \) \& \( \begin{cases} \text{lo}(x) > \text{hi}(y) \quad \text{or} \\ \text{lo}(y) > \text{hi}(x) \end{cases} \) \}

\[
\begin{cases}
\text{if } y_L \neq \emptyset \text{ \& } \text{lo}(x) \leq \max(y_L) \\
\text{then } y \leftarrow y_L
\end{cases}
\]

return \( y \)

For node \( p \):
- \( \text{left/right child} = \text{P}_L, \text{P}_R \)
- if no child, use \( \emptyset \)
- \( \max(p) = \max(\text{hi}[S_i]) \) in \( \text{subtree}(p) \)

\[
\text{while no trivial solutions: (no overlap) - OR - (xy overlap)}
\]

\[
\text{guaranteed overlap in left tree because it's a BST.} \\
\text{...so just search left.}
\]
For node \( p \):
- left/right child = \( P_L, P_R \)
- if no child, use \( \emptyset \)
- \( \max(p) = \max(hi[\text{subtree}(p)]) \)

\[
\text{Find-Overlap}(x, S) \\
y \leftarrow \text{root interval in } S \\
\text{while } y \neq \emptyset \text{ or } \bigg( \text{lo}(x) > \text{hi}(y) \text{ or } \text{lo}(y) > \text{hi}(x) \bigg) \\
\text{while no trivial solutions: (no overlap) OR (xy overlap)} \\
\text{if } y_L \neq \emptyset \text{ and } \text{lo}(x) \leq \max(y_L) \\
\text{then } y \leftarrow y_L \\
\text{else} \\
\text{return } y
\]
Find-Overlap(x, S)

\[\text{root interval in } S\]

\[y \leftarrow \text{root interval in } S\]

while \(y \neq \emptyset\) & \((\text{lo}(x) \geq \text{hi}(y)) \text{ or } (\text{lo}(y) \geq \text{hi}(x))\) & \(x \neq \emptyset\)

\[\emptyset \leftarrow \emptyset\]

\[
\begin{align*}
\text{if } y_z \neq \emptyset \text{ & } y_z \leq \max(y_z(z))
\text{then } y \leftarrow y_z
\end{align*}
\]

\[3 \text{ guaranteed overlap in left tree}\]

\[3 \text{ definitely nothing in left tree: must still try right}\]

\[\text{done}\]

For node \(p\):

- if no child, use \(P_l, P_r\)
- \(\max(p) = \max(h, [S_l, S_r])\) in subtree \(p\)
Time = height = $O(\log n)$

All $k$ overlaps: ?