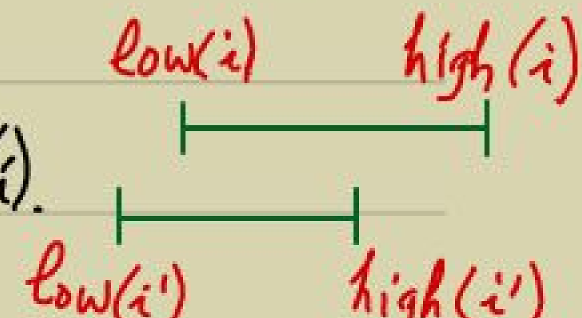
Interval Trees

- Computational geometry, or scheduling, problems require organization of intervals.

- Interval $i = [t_1, t_2]$ has the low endpoint $t_1 = low(i)$ & high endpoint $t_2 = high(i)$.

- Interval i, i' overlap if $i \cap i' \neq \phi$.
  Equivalently,
  $low(i) \leq high(i')$ & $low(i') \leq high(i)$.



- Qn: Is there a data structure where an overlapping interval can be searched in $O(\lg n)$ time?

- Ans: Let T be the set of n intervals. Organize T into an AVL tree wrt the low endpoints.

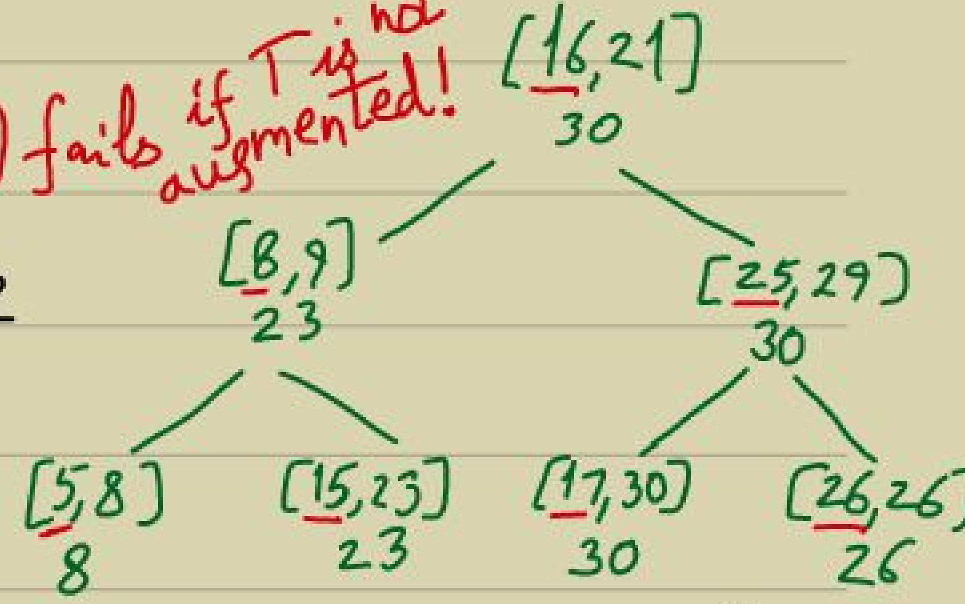
- We now need to implement:
  1) <u>Insert</u> (T, i): insert i into T.
  2) <u>Delete</u> (T, i): delete i from T.
  3) <u>Search</u> (T, i): return a pointer to a node $x \in T$ that overlaps with i.

- To search for i, just having $low(v)$, in every node $v \in T$, is not enough.

Augmented AVL

We also store <u>$max(v)$</u> $:=$ the maximum value across all the intervals in $tree(v)$.

Example: Searching for $[30,30]$ fails if T is not augmented!



▷ Insert (T, i) can be done in $O(\lg n)$ time.

Pf: One needs to change $max(v)$ in only $depth(T)$ many ancestors, while inserting i. $\square$

▷ Similarly, for Delete (T, i).

- The pseudocode for Search(T, i) is mainly guided by "$low(i) < max(left(v))$":

- $v \leftarrow root(T)$;
- while (i does not overlap $int(v)$) {
  - if ($low(i) \leq max(left(v))$)
    then $v \leftarrow left(v)$;
    else $v \leftarrow right(v)$; }
- return v;

- <u>Caution</u>: Handle the boundary conditions like – $v = NULL$ or $left(v) = NULL$ or $right(v) = NULL$.

<u>Exercise</u>: Show that it correctly finds a $v \in T$ s.t. $int(v) \cap i \neq \phi$ in $O(\lg n)$ time.

<u>Hint 1</u>: Loop invariant – If i overlaps with some interval in T, then " " " "
" " tree(v).

Hint 2: If $low(i) \leq max(left(v))$ & i overlaps with some interval in $tree(v)$, then i overlaps with someone in $left(v)$.

Proof:

- Otherwise, it means that $\forall u \in left(v)$, $i \cap int(u) = \phi$.

$\Rightarrow low(i) > high(int(u))$ OR

$high(i) < low(int(u))$

$\Rightarrow \exists u \in left(v), high(i) < low(int(u))$ [$\because low(i) \leq max(left(v))$]

$\Rightarrow low(i) < high(i) < low(int(u))$.

- This means that $high(i) < low(int(u'))$, $\forall u' \in tree(right(v)) \cup \{v\}$. [$\because$ T uses low endpoints]
- Hence, i does not overlap with any interval in $tree(v)$. $\square$

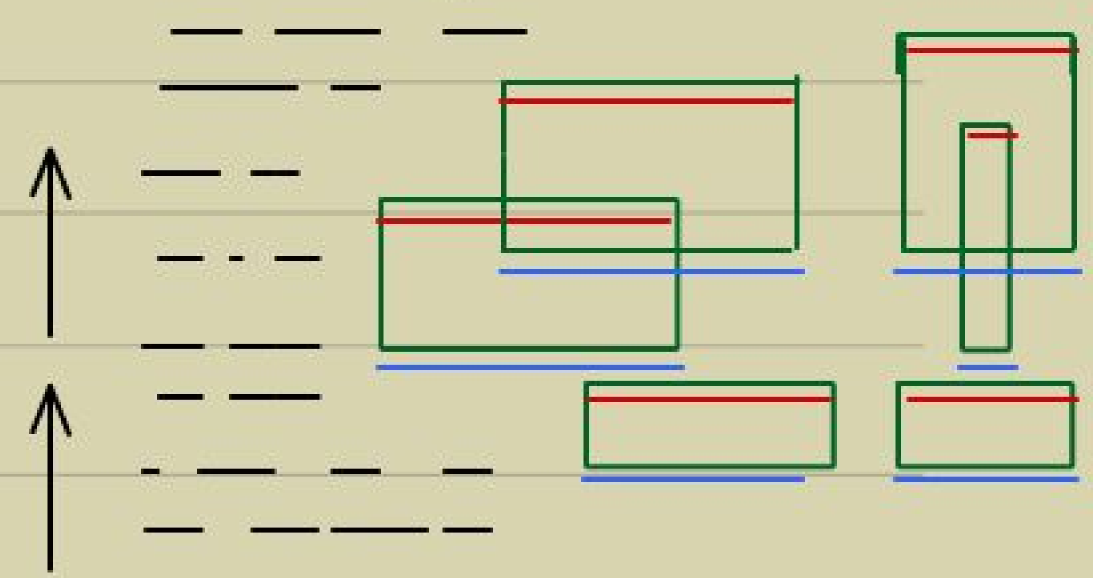
Apply to Rectangle Overlap

- Input: A list L of axis-parallel rectangles (n of them via $2n$ points).

- Output: YES if two of them overlap

Qn: Can you solve in time less than $O(n^2)$?

- Idea: (Virtual line sweep!)
  - Order the red & blue edges wrt y-coordinates in an array A.



  - Pick an edge e from A, in order.
  - If e is blue: Check whether e overlaps with an edge in an interval tree T; if no, Insert(T,e).

- If e is red: Let e' be the associated blue edge. Search & Delete e' from T. If $e' \notin T$ then OUTPUT OVERLAP.
Else goto the next e in A.

Exercise: Write the pseudocode & prove that it works in time $O(n \lg n)$.