Interval Trees

1. Background and Operations:

- Closed Time Interval:
  $\{x \in \mathbb{R} \mid \ell \leq x \leq h\} = [\ell, h]$

- Operations:
  1. insert(ℓ,h): Store $[\ell, h]$ in the collection.

  2. delete(ℓ,h): Delete $[\ell, h]$.

  3. search(ℓ,h): Return a stored interval that overlaps with $[\ell, h]$.

Note: Suppose I have an interval $[1, 5]$. Then, $[-1, 1]$, $[-1, 5]$, $[0, 5]$, $[1, 6]$, $[5, 8]$, $[0, 10]$, and more overlaps it. However, $[-1, 0]$, $[6, 10]$ and $[8, 13]$ does not overlap with $[1, 5]$.

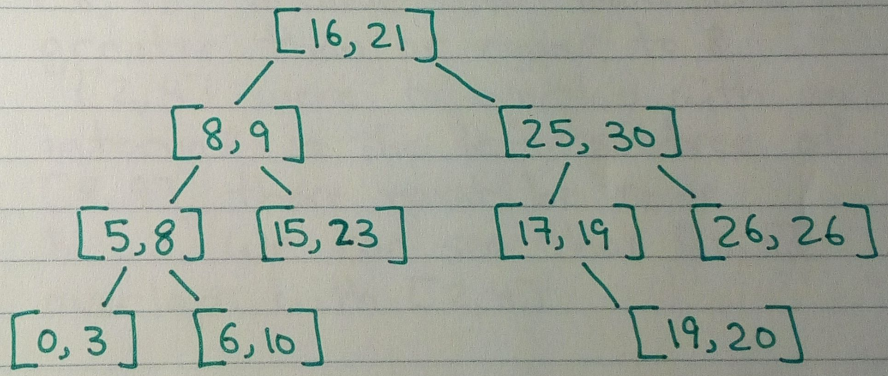
Note: To compare 2 intervals, $[a, b]$ and $[\ell, h]$:
1. If $a < \ell$, then $[a, b] < [\ell, h]$
2. If $a = \ell$ and $b < h$, then $[a, b] < [\ell, h]$.

E.g. $[1,3] < [2,5]$
E.g. $[2,3] < [2,5]$

Although this is enough for insert and delete, we need more info for search to work.

First, each node x_i stores l_i and h_i, the interval's two ends, as the key. However, this is still not enough.

Consider the interval tree below.



Suppose we want to search (10, 12). Consider [8, 9]. It doesn't overlap with [10, 12], and [10, 12] lies to the right of [8, 9].

If [10, 12] were to overlap with an interval in the left subtree of [8, 9], there must be some $hi \geq 10$. If there is an $hi \geq 10$ in the left subtree, then it is guaranteed that it overlaps with [10, 12]. We see that [6, 10] overlaps with [10, 12] in the left subtree.

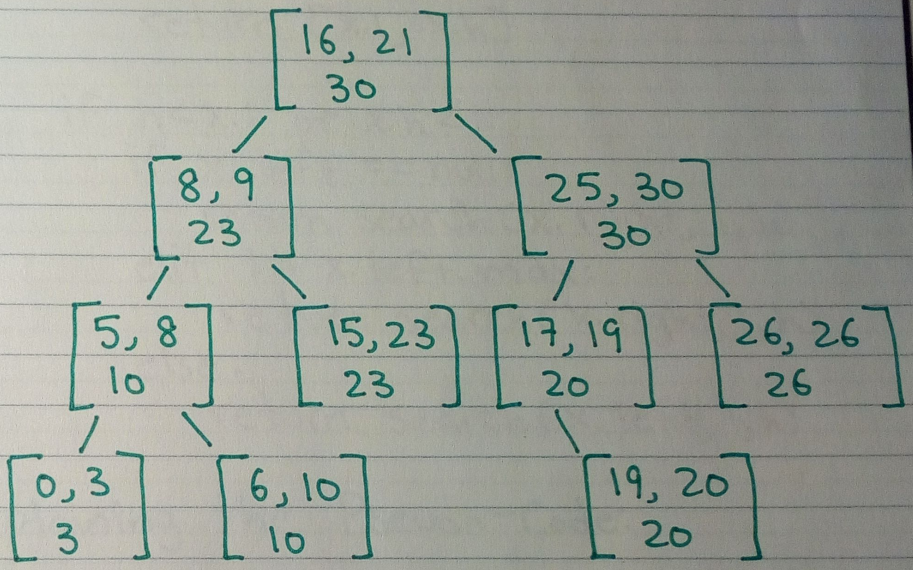
Now, suppose we want to do search (2, 4). [2, 4] does not overlap with [8, 9] and is to the left of [8, 9]. Furthermore, [2, 4] cannot overlap with any intervals in the right subtree of [8, 9], because their lower end is greater than or equal to 8. If [2, 4] were to overlap with an interval in the left subtree of [8, 9], there must be some $hi \geq 2$. We can see that [0, 3] overlaps with [2, 4].

Notice that in both cases, there needed to be an $hi \geq \ell$ in order for there to exist an overlapping interval in the left subtree.

2. Augmenting the Tree:

We can augment the tree by adding the max hi in the subtree rooted at x to each node, x.

I.e. For each node x, we add the max hi of the subtree rooted at x.



The augmented version of the tree from page 2.

3. Pseudo-Code For Search

Let x be the root of the tree.

```
Search(x, l, h):
    if x = null:          There is no overlapping
      return null         interval.

    Check if x's interval overlaps with
    [l, h].
    if x.l ≤ h and l ≤ x.h:
        return [x.l, x.h]

    if h < x.l or x.h < l:
        if x.left == null:
            return search(x.right, l, h)
        elif l > x.left.max:
            return search(x.right, l, h)
        else:
            return search(x.left, l, h)
```

4. Explaining the Pseudo-Code:

First, we check if the root is null. If it is, then there can be no overlap.

Next, if the root isn't null, we compare the root's interval with the inputted interval. We need $x.l \leq h$ and $l \leq x.h$. If only one of the two requirements is met, it wouldn't work.

For example, let $[x.l, x.h] = [1, 2]$ and let $[l, h] = [3, 4]$. Here, $x.l$ is less than or equal to h, but they don't overlap. Likewise, if $[l, h] = [-3, 0]$, $l \leq x.h$, but they don't overlap.

However, if $[x.l, x.h] = [3, 6]$ and $[l, h] = [4, 7]$, then $x.l \leq h$, $l \leq x.h$ and they overlap.

Finally, if $h < x.l$ or $x.h < l$, then we compare $[l, h]$ to x's children. If the left subtree is empty or if l is greater than the max h of the left subtree, we go to the right subtree. Otherwise, we go to the left subtree.

I.e. Consider the 2 cases below.

Case 1: We go down the right subtree. Then, there are 2 possibilities.

1. There is an overlap in the right subtree.
2. There is no overlap in either subtree. We go to the right subtree only when the left is null or $l > x.left.max$.

Case 2: We go down the left subtree. Then, one of the following must be true.

1. There is an overlap in the left subtree.
2. There is no overlap in either tree.

Now, consider these facts:

1. We went to the left subtree because $\ell \leq x.left.max$.
2. $x.left.max$ is an hi in one of the intervals in the left subtree. Let that interval be $[a, x.left.max]$.
3. Since $[\ell, h]$ doesn't overlap with any node in the left subtree, h must be smaller than a.
4. All nodes are ordered by the low value, ℓi. This means that all nodes in x's right subtree must be greater than a.

From these facts, we can deduce that $[\ell, h]$ cannot overlap with any interval in x's right subtree.