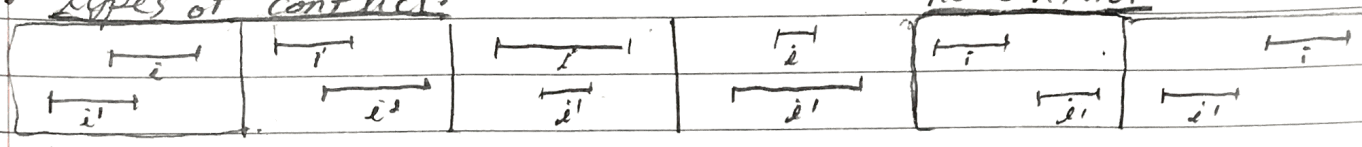
CSC263 Friday Tutorial — Feb 2, 2018

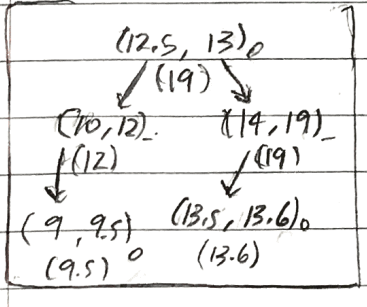
Augmented data structures

Ex. Interval trees - store intervals of time
"I want to book a conference room from 16:00 - 17:00"

Def
- closed interval: an ordered pair of real numbers $[t_1, t_2]$, $t_1 \leq t_2$
- Types of conflict:



- formal definition of overlap:
$[i \cap i' \neq \phi] \iff low[i] \leq high[i'] \;\&\&\; low[i'] < high[i]$
- Goals for solving this problem
  1) Store set S of intervals in a data structure
  2) Support the following operations in $O(\log_2 n)$ time
     - SEARCH(S, x) - INSERT(S, x) - DELETE(S, x)
     - → return pointer to y if x and y overlap, or NULL if no overlaps in S
- Step 1: Which data structure?
  AVL Trees: supports dictionary operations (search, delete, insert) in $O(\log n)$
Step 2: what info should be stored in the AVL tree?
  Each node x stores:
  - int[x] = (t_1, t_2) - balance factors - pointers
  - the max high endpoint (t_2) stored in subtree rooted at x
  Use low[x] as keys to nodes (already storing info about max)



- $max[x] = \max\{high[x], max[left[x]], max[right[x]]\}$
- INSERT/DELETE: Use AVL insert & delete, modify max field of ancestor nodes
  → constant time, done when BF updates, at most $\log_2 n$ times

```
SEARCH(T, x)
    y = root(x)
    while y ≠ NULL && (x and y do not overlap)
        if left[y] ≠ NULL && Max[left[y]] ≥ low[x]
            y = left[x]
        else:
            y = right[x]
    return y
```

// constant time check occurs $\leq \log_2 n$ times
$\therefore$ Search $\in O(\log_2 n)$

- Claim: an overlapping interval will be found if one exists in the tree.
- Loop invariant: if T contains an interval overlapping x, then the overlapping interval exists in the subtree rooted at y
- A] at root of tree, invariant trivially holds
  B] (i) left(x) = NULL, overlapping interval must be in y = right(x)
  (ii) max(left(y)) < low(x) (ie x starts later than all in left(x))
  then overlapping interval is y = right(x)
  (iii) max(left(y)) ≥ low(x)
  then some interval I in left subtree has high[I] ≥ low(x)
  everything in right subtree must start after x
  any overlapping interval must be in y = left(x)