MIDTERM 2

Wednesday, November 13, 2013 12:39 PM

Log n < n < n log n < n^2

- Balanced search trees better because reinforce n log n runtime
    - Binary search trees don't
- Trees
    - Preorder: roots first, depth first, left to right
    - Postorder: children first, breadth first
    - Inorder: children first, depth first
    - $1 \leq \text{leaves} \leq 2^h$

      $h \leq \text{internal nodes} \leq 2^h - 1$

      $\log_2(n+1) - 1 \leq h \leq n - 1$

      $\therefore$ For complete tree, there are $2^{\log_2(n+1)-1} = \frac{n+1}{2} \approx \frac{n}{2}$ leaves
    - Runtime are between n and log n depending on balanced or branched.
    - ```
      /** Node in T containing L,
       *  or null if none */
      static BST find(BST T, Key L) {
        if (T == null)
          return T;
        if (L.keyequals (T.label()))
          return T;
        else if (L ≺ T.label())
          return find(T.left(), L);
        else
          return find(T.right (), L);
      }
      ```
    - ```
      /** Insert L in T, replacing existing
       *  value if present, and returning
       *  new tree. */
      BST insert(BST T, Key L) {
        if (T == null)
          return new BST(L);
        if (L.keyequals (T.label()))
          T.setLabel (L);
        else if (L ≺ T.label())
          T.setLeft(insert (T.left (), L));
        else
          T.setRight(insert (T.right (), L));
        return T;
      }
      ```

- Queues and Stacks
    - Stacks: dishes
        - Push, pop
    - Queues: first in, first out
        - Enqueue, dequeue
        - Front
    - Constant time
    - Priority queue
        - Insert
        - Use key
    - Heap
        - Sorted by min/max/etc.
        - Each level needs to be filled up first
        - Can be represented as array
        - Compare/swap
        - Stick in new spots and bubble up(remove)/down(insert) until property satisfied
- Hash tables
    - Constant time for inserting, over linkedlist
    - chaining