Final Review
Final Exam (Out of 70)

• Part A: Basic knowledge of data structures
  – 20 points
  – Multiple choice

• Part B: Application, Comparison and Implementation of the data structures
  – 20 points
  – Apply supported operations (like find and insert) to data structures we have covered like: BST, AVL, RBTs, Multiway trie, Ternary Trie, B trees, skip lists. Also essential concepts in Huffman codes

• Part C: Simulating algorithms and run time analysis
  – 15 points
  – Graph algorithms: BFS, DFS, Dijkstra, Prims’, Kruskals’. Also union find

• Part D: C++ and programming assignments
  – 15 points
  – Short answer
B-trees

1. Construct a 2-3 tree by inserting the following keys in the order shown: 10, 15, 20, 25, 17, 30. You can check your answers and experiment with trees of your own design at the following web sites:
   - https://www.cs.usfca.edu/~galles/visualization/BTree.html

2. Which of the following are legal 2,3 trees (B tree of order 3)? For a tree that is not a valid 2,3 tree, state a reason why.

   A.
   
   B.

   C.

   D.

   E. Combination of above

Search tree property not satisfied

Valid order
Insert the value 42 into the following BTree.
Ternary Tries

(a) Consider the following ternary search tree. Nodes with double circles have their end bits set to true. Circle all of the words from the list on the right that are in the tree and write in any words that are missing. At the end you should have a complete list of all words found in the tree, and only those words.

(b) Does the height of this tree depend on the order in which the keys have been inserted? Yes

(c) Briefly explain why you would prefer to use a ternary search tree rather than a binary search tree to implement `getAllValidWords()` in PA4.

Don't need to start the search over (from root) when searching for keys that have the same prefix.
Write the sequence of vertices visited when running DFS on the following graph. Assume the link to the vertex with the minimum edge weight is chosen when multiple choices are available. 

Start @ vertex A

\[ A, C, D, B, E \]

Write the sequence of vertices visited when running BFS on the following graph. Assume the link to the vertex with the minimum edge weight is chosen when multiple choices are available.

\[ A, C, B, E, D \]
Which of the following are AVL trees?

A. A & C
B. A & B & C

Annotate the trees with balance factors.
Insert 50. Then insert 66. Draw the resulting AVL tree.
How can we make the above tree a valid red-black tree

Insert 50. Then insert 66. Draw the resulting RBT.
# Data structure Comparison

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Insert</th>
<th></th>
<th>Find</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Worst</td>
<td>Avg</td>
</tr>
<tr>
<td>Sorted array</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(\log_2 N)$</td>
</tr>
<tr>
<td>Sorted Linked list</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Queue</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$X$</td>
</tr>
<tr>
<td>Skip list</td>
<td>$O(\log_2 N)$</td>
<td>$O(N)$</td>
<td>$O(\log_2 N)$</td>
</tr>
<tr>
<td>BST</td>
<td>$O(\log_2 N)$</td>
<td>$O(N)$</td>
<td>$O(\log_2 N)$</td>
</tr>
<tr>
<td>AVL/RBT (Balanced)</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_2 N)$</td>
</tr>
<tr>
<td>Min-heap</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_2 N)$</td>
<td>$X \propto N$</td>
</tr>
<tr>
<td>Hash table</td>
<td>$O(1)$</td>
<td>$O(N)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>B-trees</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_2 N)$</td>
</tr>
</tbody>
</table>

* Under certain assumptions, refer to average case analysis of BST/AVL/RBT

\[
\sum_{i=1}^{n} i = \frac{n(n+1)}{2}
\]
Data structure Comparison

Which of the following pairs of data structures, which of the pair is the better choice for.

• Inserting a list of sorted elements (worst case):
   A. AVL tree
   B. Binary search tree
   C. They are about equal

• In-order traversal of elements:
   A. Hashtable
   B. Binary search tree (Balanced)
   C. They are about equal
Data structure Comparison

Which of the following pairs of data structures, which of the pair is the better choice for.

• Smallest average-case (Big-O) time to find an element:
  A. Hashtable $O(1)$
  B. AVL tree
  C. They are about equal

• Fastest actual time to find an element from secondary storage (NOT big-O)
  A. RBT
  B. AVL tree
  C. B-trees
  D. They are all about equal

• Requires less space:
  A. Multi-way trie
  B. Ternary tree
  C. They are about equal
Some comic relief...

```
DEFINE PANICSORT(LIST):
  IF IS_SORTED(LIST):
    RETURN LIST
  FOR N FROM 1 TO 10000:
    PIVOT = RANDOM(0, LENGTH(LIST))
    LIST = LIST[PIVOT:] + LIST[:PIVOT]
    IF IS_SORTED(LIST):
      RETURN LIST
    IF IS_SORTED(LIST):
      RETURN LIST:
    IF IS_SORTED(LIST): // THIS CAN'T BE HAPPENING
      RETURN LIST
    IF IS_SORTED(LIST): // COME ON COME ON
      RETURN LIST
    // OH JEEZ
    // I'M GONNA BE IN SO MUCH TROUBLE
    LIST = []
    SYSTEM("SHUTDOWN -H +5")
    SYSTEM("RM -RF /")
    SYSTEM("RM -RF ~/*")
    SYSTEM("RM -RF /")
    SYSTEM("RD /S /Q C:/*") // PORTABILITY
    RETURN [1, 2, 3, 4, 5]
```
Some comic relief...

```python
define jobinterviewquicksort(list):
    ok so you choose a pivot
    then divide the list in half
    for each half:
        check to see if it's sorted
        no, wait, it doesn't matter
        compare each element to the pivot
        the bigger ones go in a new list
        the equal ones go into, uh
        the second list from before
        hang on, let me name the lists
        this is list a
        the new one is list b
        put the big ones into list b
        now take the second list
        call it list, uh, a2
        which one was the pivot in?
        scratch all that
        it just recursively calls itself
        until both lists are empty
        right?
        not empty, but you know what I mean
        am I allowed to use the standard libraries?
```

http://xkcd.com/1185/
Good luck with the final!