I thought I would make a thread specifically for project specifications and questions pertaining to specifications. That way they are all relatively easy to find in one thread.

:d

I gave a similar project last semester featuring red-black trees.

Things that were missed then:

* forgetting to put a level number in front of the level
* forgetting to put an X designator for the root
* not using strcmp to order the keys
* forgetting to put the balance indicator wherever a node name appears

I will be using an AVL checker to check your trees, so the syntax has to match exactly.

I'm going to post an unfinished version of the checker soon. You should finish it out and check some hand drawn trees BEFORE you start your AVL code.

Is it acceptable to use an in-order traversal via recursion to determine the level of the nodes in our trees?

Only if your entire display of the tree is O(n).
Just to make sure that I am understanding this right and that I don't go too far before I have to turn around...

My traversal takes O(n) time and displaying the level order of my tree takes O(n) time. Together that is a combined time of O(n) + O(n). Which, if I recall correctly, is still O(n).

Is my thinking correct? I could theoretically do a traversal a dozen times and still have it be O(n)?

I am using one traversal to determine the levels of each node, the max weight of each node (furthest distance to a leaf), and min weight of each node (shortest path to a leaf).

Subject: Re: SPECIFICATIONS
Posted by lusth on Wed, 28 Sep 2016 14:09:44 GMT
View Forum Message <> Reply to Message

Seems n log n to me. log n levels and n time to find all nodes at that level.

Subject: Re: SPECIFICATIONS
Posted by btlindow on Wed, 28 Sep 2016 14:54:05 GMT
View Forum Message <> Reply to Message

Even if I am doing a constant amount of work at each level?

My traversal only visits each node once and does a constant amount of work at that node. Intuitively, to me, visiting n nodes and doing constant work would be O(n) time.

\[ T(n) = 2T(n/2) + 1 \]
\[ n^{\log \text{base 2 of } 2} = n^1 = n \]
\[ n > 1 \]
Case 1
Theta (n).

If I were to linear work at the level I could see it being n log n. Maybe I am doing this wrong?