CSCI 33500 - Spring 2016 -
Midterm, covering chapters 1 through 5.

in class March 21st

Instructions

- Write your full name on the front page.
- All four sections are worth the same amount of points.
- Use the back of each page to draft your thoughts. Use the part under the questions to write clear, and concise but detailed answers.
- No outside material or electronics are allowed.
- Use a black or blue pen.

1 Algorithm Analysis

Consider the following mystery procedure:

```
list<Object> myst_proc( const list<Object> & L1, const list<Object> & L2)
{
   list<Object> result;

   typename list<Object>::  const_iterator iterL1 = L1.begin();
   int prev;
   while (iterL1 != L1.end() )
   {
       prev = *iterL1;
       iterL1++;
       if (prev > *iterL1)
           return result;
   }

   typename list<Object>::  const_iterator iterL2 = L2.begin();
   while (iterL2 != L2.end() )
   {
       prev = *iterL2;
       iterL2++;
       if (prev > *iterL2)
           return result;
   }

   iterL1 = L1.begin();
   iterL2 = L2.begin();
   while (iterL1 != L1.end() && iterL2 != L2.end() )
   {
       if (*iterL1 < *iterL2)
       {
          result.push_back(*iterL1);
          iterL1++;
       }
       else
       {
          result.push_back(*iterL2);
          iterL2++;
       }
   }

   while (iterL1 != L1.end() )
   {
       result.push_back(*iterL1);
       iterL1++;
   }
   while (iterL2 != L2.end() )
   {
       result.push_back(*iterL2);
       iterL2++;
   }

   return result;
}
```

1.1 Give a description of what the mystery procedure does.

1.2 Give a detailed analysis of the big-O runtime of this procedure, considering that L1 has N elements, and L2 has M elements.

1.1: The first two `while` loops verify that L1 and L2 are sorted in ascending order. If either is not, then the procedure returns an empty list.
The third `while` loops merges the two lists together. At each iteration the smallest element of the two lists is put to the back of the `result` list, and the corresponding iterator incremented. The loop stops when one of the iterators reaches the end of its corresponding list.
Finally, the remainder of the list with the largest elements is all pushed to the back of `result`. Only one of the two final `while` loops will be executed, because one of the iterators will be at its end. The final result is a *sorted* merger of the two lists L1 and L2, in ascending order.

1.2: In total, both iterators are looped through exactly twice. All the operations done inside and outside of the loops take constant time (defining and incrementing iterators, comparing elements, inserting into the `result` list, etc...). The runtime is then O(2N+2M) = O(N+M).

2 Induction

Simplified, Moore's Law states that the speed of computers doubles every two years. Let's prove that this means that the speed of computers today is greater than the sum of the speeds of *all* past computers. Proove, by induction that:

$$\sum_{i=0}^{N} 2^i < 2^{N+1}$$

State clearly your base case, what is the next step of the hypothesis you are aiming to prove, and when you use the induction hypothesis in the proof.

Base case: for N=0 we have:
$\sum_{i=0}^{N} 2^i = \sum_{i=0}^{0} 2^i = 2^0 = 1$ and $2^{N+1} = 2^{0+1} = 2$
so the hypothesis is true for N=0.

Now we assume the hypothesis true at N=k, with k≥0, we need to verify that it is true at N=k+1.
So, assuming that $\sum_{i=0}^{k} 2^i < 2^{k+1}$, we aim to prove that $\sum_{i=0}^{k+1} 2^i < 2^{k+2}$.

Proof:

$$\begin{aligned}
\sum_{i=0}^{k+1} 2^i &= 2^{k+1} + \sum_{i=0}^{k} 2^i \\
&< 2^{k+1} + 2^{k+1} \text{ (by the induction hypothesis)} \\
&< 2 * 2^{k+1} \\
&< 2^{k+2}
\end{aligned}$$

Q.E.D.

3 Trees

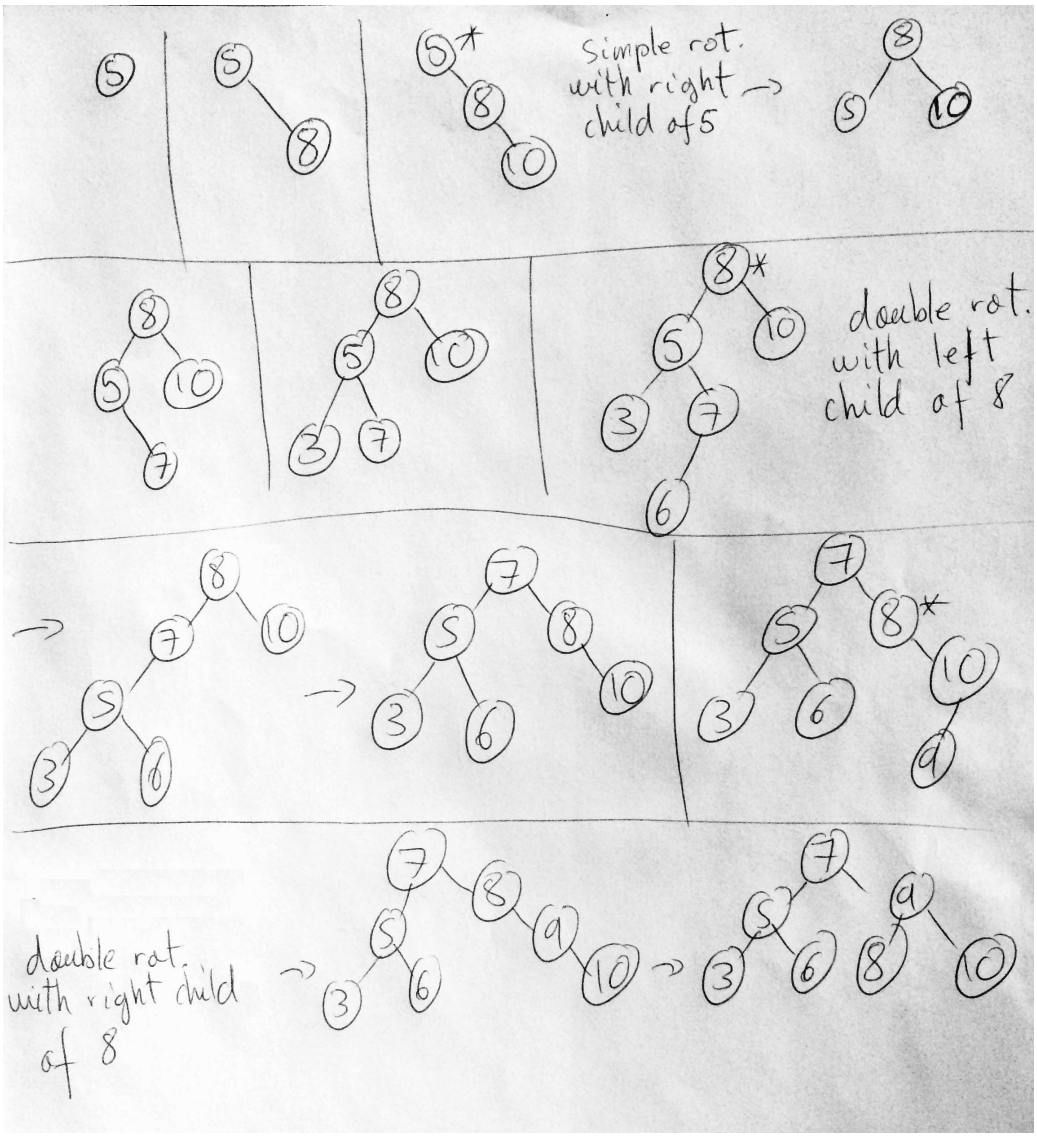
An AVL tree is a special type of binary search tree (in which a node's right descendants are all larger, the left all smaller), with an additional balance condition.

3.1 Explain the balance condition of AVL trees.

3.2 Show the steps and final result of inserting $5, 8, 10, 7, 3, 6, 9$ (in this order) into an initially empty AVL tree. After each insertion, note which nodes are imbalanced, which type of simple/double rotation with the left/right child of which imbalanced node you will use to re-balance the tree, and draw the tree before and after the rebalancing rotation.

3.1: The balance condition for AVL trees states that, for each node, the height of the left and right children of that node should not differ by more than 1.

3.2:



4 Hash Tables

Consider a hash table of size 9, using following hash function: $Hash(x) = 3 * x \bmod 9$
For example the key 20 hashes to $Hash(20) = 60 \bmod 9 = 6$, so cell position 6.

4.1 Compute the hashes of the following keys: $12, 15, 10, 19, 11, 14$

4.2 Draw the final hash table resulting from inserting the above keys in the given order. Use linear probing to resolve collisions.

4.3 Explained why this hash function is flawed.

4.1: $Hash(12) = 3 * 12 \bmod 9 = 36 \bmod 9 = 0$.
Similarly, we have:
$Hash(15) = 0,\ Hash(10) = 3,\ Hash(19) = 3,\ Hash(11) = 6,\ Hash(14) = 6$

4.2:

position	key
0	12
1	15
2	
3	10
4	19
5	
6	20
7	11
8	14

4.3: Because 9 is a multiple 3, all values will only hash to 3 positions in the table, instead of the 9 possible. This will result in more collisions than expected.