Fall 2006 CS61b Final Review

Michael Le – cs61b-tc@imail.eecs.berkeley.edu

Additional Practice Problems

Here are some additional practice problems. Due to my time constraints, solutions will not be provided to these problems. However, these problems should help you prepare for the final exam.

1. The following functions are comparable by asymptotic growth. Can you put them in order?

$$n,\ 2^n,\ n\log n,\ n^n,\ n-n^3+7n^5,\ n^2+\log n,\ n^2,\ \log n,\ n!$$

   Note, if you put function $f(n)$ to the left of $g(n)$, then it must be the case that $f(n) = O(g(n))$.

2. Consider the following naive factoring algorithm:

   For every integer p such that 1 < p < r, check if p divides r.
   If so, print "The secret message is p!" and stop, else continue.

   a) Suppose that the eavesdropper uses the above algorithm and has a computer that can carry out in one microsecond (one millionth of a second) a division between two integers of up to 100 bits each. Give an estimate of the time that it will take in the worst case to decipher the secret message if r has 100 bits.

   b) What is the worst-case time complexity of the above algorithm? Since the input to the algorithm is just one large number, r, assume that the input size N is the number of bytes needed to store r, i.e., $N = (\log_2 r)=8$, and that each division takes time $O(N)$.

3. A 3-way mergesort on an array with N elements works as follows:

   1. divide the array into 3 subarrays of size N/3
   2. recursively sort the 3 subarrays;
   3. merge the 3 subarrays together.

   Find the time complexity of 3-way mergesort. You may assume that N is a power of 3.

4. Consider the following variation of insertion sort. Assume A is the array we are sorting and that A[1], A[2], . . . , A[p] is the subarray of A which is currently sorted. Rather than sequentially searching for A[p + 1]'s position in A[1], . . . , A[p], suppose we perform a binary search. If this is the case for $1 \leq p \leq n - 1$ (where n is the size of A), then what is the time complexity of the sorting algorithm? Why?

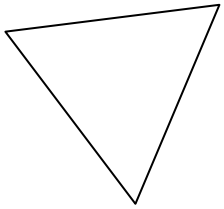
5. Consider binary trees that have single characters stored at each internal node. Draw a binary tree that will spell out the phrase: ANUPSIDEDOWNTREE when nodes are visited in preorder, and UNPADIESDNWTOERE when the **same** tree is visited in inorder.

6. Consider a search tree T storing 100,000 keys. What is the worst-case height of T in the following cases?

    a) T is a 2-3-4 tree
    b) T is a binary search tree.

7. Let G = (V,E) be a given graph where |V| = N and |E| = M. Suppose we wish to color each node of G either black or white such that no two adjacent nodes share the same color. Assume without loss of generality that the initial node visited in the graph is to be colored black. Design an $O(N + M)$ time algorithm to provide such a coloring or determine that none exists.

    Below are simple examples of a graph that is 2-colorable (on the left) and another that is not (on the right).



8. Let graph G = (V, E) be a directed graph and vertices *start*, *goal* to be in V. Assuming all edges in E are of non-negative weight, describe an efficient algorithm for finding the longest acyclic path from *start* to *goal*.

9. Give an efficient algorithm for testing whether a directed contains no cycles. What is the time complexity of the algorithm?

10. Counting sort can be considered to have three stages:

    1. Count the keys in the collection.
    2. Count the number of keys less than a given element in a collection.
    3. Build the sorted collection.

    Suppose we ran counting sort on an array. In class, the third state was done as follows:

```
for (i = 0; i < x.length; i++) {
  y[counts[x[i].key]] = x[i];
  counts[x[i].key]++;
}
```

    If we traversed the array backwards (from high index to low index), is the result still stable?

11. A sequence of stack operations is performed on a stack whose size never exceeds *k*. After every *k* operations, a copy of the entire stack is made for backup purposes. Show that the amortized cost of *n* stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

12. A sequence of n operations is performed on a data structure. The ith operation costs i if i is an exact power of 2, and 1 otherwise. What is the amortized cost of each operation?