Algorithms Class Outline for Quiz 1

Use basic math formulas from Appendix: geometric series, logs, sums

Algorithm analysis
- Worst case analysis
- Best Case analysis
- Simple average case analysis
- Simple lower bounds on algorithms

Growth order of functions:
- sequence of orders for common functions
- simplifying order expressions
- big oh, theta, and omega

Recurrence relations:
- Master theorem! Be able to classify recurrence relations and know when to use it!
- Creating recurrence relations T(n) = f(n) + terms involving T, by analyzing actual algorithms
- Distinguish nonrecursive term f(n) from recursive terms and from T(n)!
- Formulas for T(n) as sum if T(n) = T(n-1) + f(n)
- Formulas discovered from simple patterns generated by recurrence relations
- Examples from book leading to recurrence relations

Seeing situations where recursion is appropriate

Searching
- linear, binary

Sorting
- Quicksort,
  - worst case, best case, order of average case
- MergeSort
  - basic and enhancements
  - Worst case, best case
- Insertion Sort, Selection Sort, Heapsort, Radix Sort, Counting Sort

Choose best sort for situation.

Classify problems and model them based on all these concepts

Amortizing is NOT on the quiz.

Two review problems are on the next page, with small hints on the bottom of the page, and answers on the third page,

Planned Quiz Ground Rules:, for quiz *sent out electronically by Friday morning, Feb 22*

Algorithms Take Home Quiz 1 Due *on paper* at the start of class Feb 25, 2013 Dr Harrington

You may use previously prepared notes on two sides of an 8.5 x 11 inch piece of paper. You do not need to do the quiz all at one sitting, but once you *look* at it, finish it before discussing algorithms class or looking at an algorithm text or reference or your notes. I hope it takes around an hour of active work, but if you need more time, you may take **up to three hours**. Calculators are not allowed. You are encouraged to use a computer ***only*** as a word processor to print as much as is convenient of your textual answers, and then add anything else in handwriting. Do ***not*** use a computer for algorithms reference or for running code.

I have read and followed the instructions above, and worked alone, using only the allowed notes:

Signature:_____Name printed _____

Approximate number of minutes working on the quiz _____

Review/extra problems

1. Suppose a[0], a[1], and a[2] are distinct, and all orderings of their sizes are equally likely. What is the average number of comparisons of array elements in sort3?

```
public static void sort3(int[] a)
{
   int temp;
   if (a[0] < a[1]) {                          //1
      if (a[1] > a[2])                         //2
        if (a[0] < a[2]) {                     //3
           temp = a[1]; a[1] = a[2]; a[2] = temp;
        }
        else {
           temp = a[0]; a[0] = a[2]; a[2] = a[1]; a[1] = temp;
        }
   }
   else if (a[0] < a[2]) {                     //4
           temp = a[0]; a[0] = a[1]; a[1] = temp;
      }
      else if (a[1] < a[2]) {                  //5
         temp = a[0]; a[0] = a[1]; a[1] = a[2]; a[2] = temp;
      }
      else {
         temp = a[0]; a[0] = a[2]; a[2] = temp;
      }
}
```

2. The recursive function below is admittedly useless, but there is a recurrence relation for the asymptotic order of the time, T, in terms of n, in the form

$$T(n) = aT(n/b) + \Theta(n^d)$$

What are a, b, and d? Explain. Answers: a _____ b _____ d _____

```
//Assume A has n elements
static double p2r(double[] A, int n, int k) {
    if (n <= 1)
        return 0.0;
    else {
        double sum = 0;
        for (int i = 0; i < n; i += 7)
            sum += A[i]*k;
        return sum + p2r(A, 2*n/9, k*1 % n) + p2r(A, 2*n/9, k*2 % n) +
                     p2r(A, 2*n/9, k*4 % n) + p2r(A, 2*n/9, k*8 % n);
    }
}
```

1. Hint: small, finite problem, count by hand – no shortcuts.

2. Hint: you only care about time order; ignore all but the little bit that matters.

Solutions on next page

Solutions to review problems

1: There are 6 equally likely permutations of the order, so each has 1/6 probability.
We can count the number of comparisons for each order. On each line I show an initial order of the elements, the labels of the if statements executed, and the count:
a[0] < a[1] < a[2]: 1, 2, total 2
a[0] < a[2] < a[1]: 1, 2, 3, total 3
a[2] < a[0] < a[1]: 1, 2, 3, total 3
a[2] < a[1] < a[0]: 1, 4, 5, total 3
a[1] < a[2] < a[0]: 1, 4, 5, total 3
a[1] < a[0] < a[2]: 1, 4, total 2

average = (2 + 3 + 3 + 3 + 3 + 2)/6 = 8/3

2: a=4 b=9/2 or 4.5 d=1
The coefficient a is the number of direct recursive calls. The return statement has 4 recursive calls.

The denominator b is the consistent factor by which the size of the problem is chopped in the recursion. 2n/9 = n/(9/2).

The rest of the time for the function comes from the parts outside the recursive calls. Everything is O(1) except for the for loop. There the increment is 7, so the number of repetitions is about n/7, which is $\Theta(n) = \Theta(n^1)$ so d is 1.