Theory Homework 1

Honor Data Structures and Algorithms in JAVA (3137)
Shlomo Hershkop
Department of Computer Science
Columbia University
Fall 2013

Out: Monday Sept 16
Due: Monday, Sept 30 (11 pm).

Goal: review recursion, running time, analysis, linked lists.

Theory (100 points)

1. Why do we use a simplified model to model runtime when the real world runtimes would be affected by the language, machine, coding style and CPU instructions ?

2. You are evaluating two sorting algorithms. You expect that your O(n log n)-time is always faster than your $O(n^2)$-time algorithm. To make sure, you implement both algorithms and run the two algorithms on a couple of randomly generated data sets of different sizes. Unfortunately you find that sometimes if around n = 50 the $O(n^2)$-time algorithm actually runs faster, and mostly when n >=50 is the O(nlogn)-time one is faster. Explain why this scenario is possible. You may give numerical examples to illustrate your point.

3. Here is an example of a recursive fib function:

```
public int fib (int n)
{
  if (n < 1)
    return 1;
  else
    return fib(n-1) + fib(n-1);
}
```

   We made the claim that tail recursion is slow. Run this code (or any other tail recursive methods for 100, 200, 500, and 1000 and calculate the time it needs to run (show code and timings). Can you turn it into non tail recursion like we did in class with the others ?

4. Given an array of n elements, Algorithm C executes an O(n) time computation for each even number location and O(log n) time computation for each odd number

location in the array.

a) what is the best case and worst case running time for Algorithm C ?
b) what would change if the value of the location was used ? ie a location with odd value ran at O(log n) ?

5. Assume I want an array of size N with all the number from 1 to N in random order (each appearing only once). Every time you run the algorithm the ordering should be different. Describe 2 algorithms for generating the random ordered array and give the runtime in terms of Big-O for each of them.

6. Given a set of unordered numbers (can have duplicates) and a value K, **describe** an algorithm and **give the running time** for finding if any two elements in the list a,b can be subtracted from each other (either a-b or b-a) to give you K. (that is a-b=k or b-a=K).

7. Arrange the following expressions by growth rate from slowest to fastest. Please describe how you got your answers (google is not a valid answer)
   a) $10n^2$
   b) $\log_3 n$
   c) 2^n
   d) $10n$
   e) 8
   f) $\log_2 n$
   g) $n \log_2 n$
   h) $n!$

8. Give the exact and Big-Oh running time for each of the following program fragments:

a)

```
sum = 0;
for (i = 0; i < 3; i++)
  for (j = 0; j < n; j++)
    sum++;
```

b)

```
sum = 0;
for (i = 0; i < n*n; i++)
 { sum++;  }
```

c)

Assume the array contains n values.

```
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++)
    array[i] = random(n);     // random() takes constant time
  sort(A, n);                 // sort takes n log n time
}
```

```
d)
sum = 0;
if (EVEN(n))                  // EVEN(n) is true iff n is even
  for (i = 0; i < n; i++)
    sum++;
else
  sum = sum + n;
```

9. Assume I have a linked list of size n. Describe an algorithm for deleting all **odd** elements in a given list. (Odd refers to first, third, fifth etc position in the list NOT the value of the items). Describe the algorithm and give the runtime. also be specific on implementation details.

10. Suppose you have a list of N numbers, and some of the values repeat. Describe an algorithm and give the runtime for an algorithm which finds if any one value repeats itself more than N/2 times. (there might be one or zero of these). For example if your list is (4,4,3,2,3,3,2,3,3,3) then the element you want to identify is "3".

11. Why does implementation matter in Data structures ?