CSCI 3230, Fall 2018 Assignment 1

Posted: August 12, 2018 **Due: 11:00 pm, August 31, 2018 (Friday)**

(Mandatory assignment cover-sheet; without it, your work will not be marked.)

Submitting student: Blair Weaver Your mark: ____ out of 100

Collaborating classmates: _____

Other collaborators: _____

References other than the textbook and handouts:

www.geeksforgeeks.org

Regardless of the collaboration method allowed, you must always properly acknowledge the sources you used and people you worked with. Your professor reserves the right to give you an exam (oral, written, or both) to determine the degree that you participated in the making of the deliverable, and how well you understand what was submitted. For example, you may be asked to explain any solution that was submitted and why you choose to write it that way. This may impact the mark that you receive for the deliverable.

So, whenever you submit a deliverable, especially if you collaborate, you should be prepared for an individual inspection/walkthrough in which you explain what every line of assignment does and why you choose to write it that way.

1. **(10 points)** Suppose we are comparing implementations of two algorithms $\mathcal{A}$ and $\mathcal{B}$ on the same machine. For inputs of size n, the algorithm $\mathcal{A}$ runs in $8n^2$ steps, while the algorithm $\mathcal{B}$ runs in $64n \log n$ steps. For which values of n does the algorithm $\mathcal{A}$ beat the algorithm $\mathcal{B}$?

Solution:

$8n^2 = 64n \log n$ (intersection point)

$n = 8 \log n$

$2^{n/8} = n$

since we want A to beat B,

want $2^{n/8} < n$

For whole numbers of n, this would be true for

$n < 44$

$2^{43/8} < 43$

2. **(10 points)** Rank the following functions by asymptotic growth rate from the smallest to the largest.

(a)

$$n + 1000, 2n^2 + 100, 2^n + n^2 + 10n, 4^n, n \cdot 3^n, (5/2)^n, 8n^2, n!, 2^{\log n}, 4^{\log n}, (n+1)!, n^{2/3}$$

(b)

$$4n \log n + 2n, 2^{10}, 2 \log n, 3n + 100 \log n, 4n, 2^n, n^2 + 10n, n^3, n \log n.$$

Solution:

A) $n^{2/3}$, $n + 1000$, $2n^2 + 100$, $8n^2$, $n!$, $(n+1)!$, $2^{\log n}$, $4^{\log n}$, $2^n + n^2 + 10n$, $n \cdot 3^n$, 4^n

B) 2^{10}, $2 \log n$, $3n + 100 \log n$, $4n$, $n \log n$, $4n \log n + 2n$, $n^2 + 10n$, n^3, 2^n

3. **(20 points)** What is the asymptotic complexity of the following methods, in terms of the Big-O notation?

(a)
```
void methodA(int n) {
  for (int i = 1; i < n; i++) {
    for (int j = n; j > n - i; j--){
      System.out.println(j);
    }
  }
}
```
$O(n^2)$

(b)
```
void methodB(int n) {
  for (int i = n; i > 0; i--) {
    for (int j = 0; j < i; j++) {
      fun(n);
    }
  }
}
void fun(int n){
  for (int i = 1; i < n; i = i * 3){
    System.out.println(i);
  }
}
```
$O(n^2 \log n)$

(c)
```
void methodC(int n) {
  for (int i = n; i <= n; i++) {
    for (int j = n; j > 1; j = j / 2) {
      System.out.println(j);
    }
  }
}
```
$O(\log n)$

(d)
```
void methodD(int n, char ch) {
  if (ch == 'a') {
    for (int s=1; s<n; s++)
      for (int t=1; t<n; t=t*4) {
        System.out.println(t);
      }
  }
  else if (ch == 'b') {
    for (int i=0; i<n; i++) {
      System.out.println(i);
    }
  }
  else if (ch == 'c') {
    for (int i=0; i<n; i++) {
      System.out.println(i);
```
$O(n^2)$

```
    }
  } else {
    for (int k=n; k>1; k--)  // 4, n-1
      for (int m=1; m<k; m++) { // 5, n^2
        System.out.println(m);
      }
  }
}
```

Solution:

a) $O(n^2)$

b) $O(n^2 \log n)$

c) $O(\log n)$

d) $O(n^2)$

4. **(20 points)** Write a short recursive Java method that rearranges an array of integer values so that all the even values appear before all the odd values.

   (Include screenshots of your return results and submit your **java code** in a separate file.)

   Solution:

5. **(20 points)** Suppose you are given an array, A, containing n distinct integers that are listed in increasing order.

   (a) **(15 points)** Given a number k, describe a recursive algorithm to find two integers in A that sum to k, if such a pair exists.

   (Include screenshots of your return results and submit your **java code** in a separate file.)

   (b) **(5 points)** What is the running time of your algorithm?

Solution:

B) $\Theta(n^2)$

6. **(20 points)** Given two arrays of the same size, each consisting of n positive integers, write a Java program to determine how many *unique* integers the second array have. Here a unique integer means that it shows only once in the second array but does not show in the first array. For example, if the first array is $[1, 5, 3, 6, 8]$ and the second array is $[2, 4, 2, 5, 9]$ (with $n = 5$), the second array has two unique integers (4 and 9).

   What is the complexity of the program? Measure the runtime of the program for array sizes of 10, 100, 1000 and 10000. Use *System.nanoTime()* to get time measurements.

   (Include screenshots of the program running and submit your **java code** in a separate file.)

   Solution:

$\Theta(n^2)$ since worst case for both methods is $\Theta(n^2)$