CS 477/677 Analysis of Algorithms

Homework 3

Due September 22, 2020

Note: for the programming questions upload separate files in the .c or .cpp format. Include any specific compiling instructions in the top comments section of your files.

1. **(U & G-required) [20 points]** Consider the following algorithm.

```
ALGORITHM Enigma(A[0..n − 1])
//Input: An array A[0..n − 1]  of integer numbers
for $i \leftarrow 0$ to $n - 2$ do
    for $j \leftarrow i + 1$ to $n - 1$ do
        if $A[i] == A[j]$
            return false
return true
```

a) [5 points] What does this algorithm do?

b) [15 points] Compute the running time of this algorithm.

2. **(U & G-required) [40 points]**

(a) [20 points] Implement in C/C++ a divide and conquer algorithm for finding **the values of both the largest and the smallest element** in an array of *n* numbers. Show how your algorithm runs on the input A = [1 4 9 3 4 9 5 6 9 3 7].

(b) [10 points] What will be your algorithm's output for arrays with several elements of the largest value? Indicate the answer on the input given above.

(c) [10 points] Set up and solve a recurrence relation for the number of key comparisons made by your algorithm.

Note: Name your source file problem2.c or problem2.cpp.

3. **(U & G-required) [40 points]**

We can implement Mergesort without a recursion by starting with merging adjacent elements of a given array, then merging sorted pairs, and so on. Implement this bottom-up version of Mergesort in C/C++ and show how your algorithm runs on the input A = [1 4 9 3 4 9 5 6 9 3 7 2].

Note: Name your source file problem3.c or problem3.cpp.

4. (G-Required) [20 points] Use a loop invariant to prove that the following algorithm computes `a` raised to the power of `n`:

```
Exp(a, n)
{
     i ← 1
     pow ← 1
     while ( i ≤ n )
     {
          pow ← pow*a
          i ← i + 1
     }
     return pow
}
```

Extra credit

5. [20 points] Consider the following algorithm.

```
ALGORITHM Mystery(A[0..n – 1, 0..n]))
//Input: An n-by-n+1 matrix A[0..n – 1, 0..n] of real numbers
for i ← 0 to n-2 do
     for j ← i +1 to n − 1 do
          for k ← i  to n do
               A[j,k] ←  A[j,k] – A[i,k] * A[j,i]/A[i,i]
```

a) [15 points] Compute the running time of this algorithm (make sure to count all the primitive operations separately).

b) [5 points] What obvious inefficiency does this pseudocode contain and how can it be eliminated to speed up the algorithm?