COMS 4231: Analysis of Algorithms I, Fall 2021

Problem Set 1, due Friday October 1, 11:59pm in Gradescope

Please follow the homework submission guidelines posted on Courseworks.

For all algorithms that you give, show their correctness and analysis of the running time. All times are worst-case, unless specified otherwise.

Problem 1. (12 points) For each of the following two code fragments, give its asymptotic running time in the form $T(n) = \Theta(f(n))$ for an appropriate function $f(n)$, and justify your answers. The function $f(n)$ in each case should have the form n^c for a suitable constant c. The input in both cases is an array $A[1\ldots n]$ of numbers.

(a)
```
s = 0;
for i=1 to n
    for j= i to n
        if (A[i] > A[j])  then s = s+1
return s
```

(b)
```
s = 0;
for i=1 to n
    for j= i to n
        for k= i to j
            if (A[k] > A[i]+ A[j])  then s = s+1
return s
```

Problem 2. (20 points) Order the following functions by order of growth, that is, if a function f is listed before a function g then $f(n)=O(g(n))$. Group together functions that have the same order of growth, i.e. $f(n)=\Theta(g(n))$. All logarithms are to the base 2, unless explicitly specified otherwise. Justify only the functions that you have grouped together (if any). You do not need to justify the rest of your ordering.

$4n^2 + 7n$, $\sqrt[4]{n}$, $\log(n^4)$, $\log\log n$, $(\log n)^2$, $n(\log n)^2$, $n!$, $4^{\log n}$,

$4^{\sqrt{n}}$, $2^{(\log n)^2}$, $\sqrt{4^n}$, 10, 2^{n+2}, 3^n, $n^{\log n}$, n^4, $\log_{10} n$, n^n

Problem 3. (18 points) Solve the following recurrences asymptotically. For each recurrence show *$T(n)=\Theta(f(n))$* for an appropriate function *f*. Assume that *$T(n)$* is constant for sufficiently small *n.* Justify your answers. You may use the master theorem (if applicable) or any of the other methods.

a. $T(n) = 4T(n/2) + n^2$
b. $T(n) = 3T(n/4) + n$
c. $T(n) = 3T(n/2) + n\log n$
d. $T(n) = 2T(n-3) + 1$
e. $T(n) = T(\sqrt{n}) + \log n$
f. $T(n) = T(2n/3) + T(n/4) + n$

Hint. For part (f) you may want to use the recursion tree method to bound $T(n)$ asymptotically, or you can guess the form of the solution and use the substitution method to verify it.

Problem 4. (20 points)
1. (13 points) We are given two sorted arrays A and B with *n* (positive or negative) integers each, and wish to determine if there is an element *a* of A and an element *b* of B such that $a+b=100$.
Give an algorithm for this problem that runs in linear time (i.e. O(n) time) in the worst case.

2. (7 points) We are given an unsorted array C of *n* integers and wish to determine if there are two elements *a,b* of C such that $a+b=100$.
Give a O($n \log n$)-time algorithm for this problem.

Problem 5. (30 points) We are given two positive integers *a, b* (in binary) with *n* bits, and want to compute their greatest common divisor.

1. (10 points) Consider the following naïve algorithm:

```
t := min(a, b);
while (t does not divide both a,b)
      { t := t-1 }
return t
```

Is the number of iterations bounded by a polynomial in *n* in the worst case, i.e. is it O(n^c) for some constant *c*? Justify your answer.

2. (20 points) Consider Euclid's algorithm:

```
 $x$ := max($a$, $b$);
 $y$ := min($a$, $b$);
while ($y$ does not divide $x$)
     { $r$ := $x$ mod $y$ ;  $x$ := $y$ ;  $y$ := $r$ }
return $y$
```

Recall that (x mod y) denotes the remainder of the division of x by y; thus, x mod y =0 if and only if y divides x.

a. (8 points) Let $C(a,b)$ denote the set of positive integers that divide both a and b. Show the following loop invariant: (i) $x \geq y$, (ii) $C(a,b) = C(x,y)$.

b. (4 points) Show that Euclid's algorithm returns the greatest common divisor of a, b.

c. (8 points) Let $f(n)$ be the number of operations performed by Euclid's algorithm (you can regard max, min, mod as primitive operations for this problem). Show that $f(n)$=O(n). (*Hint:* Show that (x mod y) ≤ $x/2$, and use this fact to deduce that x,y decrease at least by a factor of 2 after every two iterations.)