CMPUT 204 — Problem Set 1 Partial Solution

(Partially provided by *Xiaohu Lu*)

Topics covered in Part I are mathematical background (Appendix A, pages 1058–1069; Appendix C, pages 1094–1126), mathematical induction and loop invariant, running time (function) comparison, and insertion sort; in Part II are algorithm analysis using insertion sort as an example and algorithm design using merge sort as an example, growth of functions, and asymptotic notations.

It is highly recommended that you read pages **1–61**, **1058–1069**, and **1094–1126** very carefully and do **all** the exercises. The following are some of them that you are **REQUIRED** to practice on.

Quiz questions are mostly based on this list, with some minor modifications necessary. Consult your instructor and TAs if you have any problem with this list.

Part II

1. Give, in big-O notation, the worst case number of times the operation $+$ is performed. You may assume that all variables are integers. Justify your answers.

   (a) Procedure $f_1(n)$, n is an integer

   ```
   for i ← 1 to n − 1 do
       for j ← i + 1 to n do
           for k ← 1 to j do
               x ← a + b + c
   ```

   Hint:

   For each of the k-for loop, 2 additions; and there are j iterations.

   For each of the j-for loop, $2 \times j$ additions (why?); and there are $(n - i)$ iterations.

   For each of the i-for loop, what is the number of additions? how many iterations.

   The worst case number of additions performed is $\frac{n(n-1)(n+1)}{3}$ and thus in $\Theta(n^3)$.

   (b) Procedure $f_2(n)$, n is an integer

   ```
   for i ← 1 to n do
       if odd(i) then
           for j ← 1 to n do
               x ← x + 1
           for j ← 1 to n do
               y ← y + 1
   ```

   Hint:

   For every odd i, n additions are performed.

   How many odd numbers are there in between 1 and n.

   $\Theta(n^2)$.

2. For each of the following statements give an adequate proof to show when it is true or an adequate counterexample to show when it is false. You may assume that all functions are positive non-decreasing.

   (a) If $f(n) + g(n) \in O(n^2)$ then $f(n) \in \Theta(n^2)$.
   Hint: False.
   Counterexample: $f(n) = n$, $g(n) = n^2$.

   (b) If $\frac{f(n)}{g(n)} \in \Theta(\lg n)$ then $g(n) \in o(f(n))$.
   Hint: True.
   Prove by defintions.

   (c) If $g_1(n) \in O(f_1(n))$ and $g_2(n) \in \Theta(f_2(n))$ then $g_2(n)^{g_1(n)} \in O(f_2(n)^{f_1(n)})$.
   Hint: False.

   (d) For any positive constant c, $f(cn) \in \Theta(f(n))$. (Hint: consider some of the fast-growing functions).
   Hint: False.

   (e) If $f(n) \in O(n^2)$ and $g(n) \in O(n^2)$ then $\frac{f(n)}{g(n)} \in O(1)$.
   Hint: False

   (f) If $f(n) \in \Theta(n^3)$ and $g(n) \in \Theta(n^2)$ then $\frac{f(n)}{g(n)} \in \Theta(n)$.
   Hint: True.

(g) $n^\alpha \in \Theta(n \log n)$ for some $\alpha > 1$.
Hint: False.

(h) $n^k \in \Omega(n)$ for $k > 1$.
Hint: True.

(i) If $f(n) > g(n)$ then $f(n)+g(n) \in \Theta(f(n))$.
Hint: True.

(j) If $f(n) \in \Theta(n)$ and $g(n) \in \Theta(n)$ then $2^{f(n)} \in O(2^{g(n)})$.
Hint: False.

(k) $\log n \in o(n^\alpha)$ for any $\alpha > 0$.
Hint: True.

(l) $\log(n^{0.1}) \in o(\log n)$.
Hint: False.

(m) If $f(n) \in \Theta(g(n))$ then $f(n) - g(n) \in \Theta(g(n))$.
Hint: False.

(n) $\sum_{i=2}^{n-1} \log_2 i \in \Theta(n \log n)$.
Hint: True.

(o) If $f(n) \in O(n^2)$ and $g(n) \in O(n^2)$ then $\frac{f(n)}{g(n)} \in O(1)$.
Hint: False.

(p) If $f(n) \in \Theta(n^3)$ and $g(n) \in \Theta(n^2)$ then $\frac{f(n)}{g(n)} \in \Theta(n)$.
Hint: True.

(q) $n^\alpha \in \Theta(n \log n)$ for some $\alpha > 1$.
Hint: False.

(r) $n^k \in \Omega(n)$ for $k > 1$.
Hint: True.

(s) If $f(n) > g(n)$ then $f(n)+g(n) \in \Theta(f(n))$.
Hint: True.

3. P27, Ex 2.2-2.

Hint:

(a) Pseudo code:

```
SELECTION-SORT(A)
  for j ← 1 to length[A] - 1
    do key ← A[j]
      index ← j
      i ← j + 1
      while i ≤ length[A]
        if A[i] < key
          key ← A[i]
          index ← i
        i ← i + 1
      A[index] ← A[j]
      A[j] ← key
```

(b) Loop invariant:
At the start of each iteration of the `for` loop, the subarray $\mathbf{A}[1..j-1]$ contains the $j-1$ smallest elements of the entire array in sorted order.

(c) After $n-1$ iteration, the subarray$\mathbf{A}[1..n-1]$ contains $n-1$ smallest elements of $\mathbf{A}$ in sorted order, $\mathbf{A}[n]$ is the largest element, therefore, the entire array is sorted.

(d) Best case running time: $\Theta(n^2)$
Worst case running time: $\Theta(n^2)$

4. P27, Ex 2.2-3. Hint:

(a) Average case: $\frac{n}{2}$

(b) Worse case: n

(c) Average case: $\Theta(n)$
Worst case: $\Theta(n)$

5. P36, Ex 2.3-1.

6. P36, Ex 2.3-3.
Hint: Identify what is the base case/step.

7. P36, Ex 2.3-4.
Hint:
The running time $T(n)$ for sorting n elements is the sum of the time $T(n-1)$ to sort the first $n-1$ elements and the time of inserting the n^{th} element.

8. P36, Ex 2.3-6.
Hint:
The dominant operation is no longer the KC, but "copy". What is the WC number of "copies"? ($\Theta(n^2)$)

9. P38, Prob 2-2.
Hint:

(a) Elements of the sorted array must be from the original array.

(b) At the start of each iteration of the `for` loop, the subarray $\mathbf{A}[j+1...n]$ contains elements from $\mathbf{A}$ which are all greater then $\mathbf{A}[j]$.

(c) At the start of each iteration of the `for` loop, the subarray $\mathbf{A}[1...i-1]$ contains the $i-1$ smallest elements of $\mathbf{A}$ in sorted order.

(d) Worse-case running time: $\Theta(n^2)$.

Slightly worse than `Insertion sort` — count the exact numbers of KC in both algorithms.

10. P50, Ex 3.1-1.

11. P50, Ex 3.1-5.

12. P58, Prob 3-3.

    Using functions n, $\lg^* n$, $\lg(n!)$, 2^n, $n!$, n^2, $\lg^2 n$,$\left(\frac{3}{2}\right)^n$, $2^{\lg n}$, $n \lg n$.

    Hint:

    In increasing order:

    $\lg^* n, \lg^2 n, (n, 2^{\lg n}), \lg(n!), n\lg n, n^2, \left(\frac{3}{2}\right)^n, 2^n, n!$.