COMP 352 Fall Semester

Assignment 1

Rui Zhao

40018813

Writing questions:

Question 1

a) The pseudo code of the given question:

Algorithm *myMethod (A,n)*

Input array A of n integers

Output array A whose elements have been modified according to the requirements

endOfFirstPart ← 0

startOfSecondPart ← 0

if n % 2=0 **then**

  endOfFirstPart ← n/2-1

  startOfSecondPart ← n/2

  if (endOfFirstPart+1) % 2=1 **then**

    endOfFirstPart ← n/2

    startOfSecondPart ← n/2+1

else

  endOfFirstPart ← (n-1)/2-1

  startOfSecondPart ← (n+1)/2

i ← 0

while i < endOfFirstPart and n≠1 **do**

swap (A[i], A[i+1])

i ← i+2

end while

while startOfSecondPart < (n-1) and n≠1 **do**

A[startOfSecondPart+1] ← A[startOfSecondPart] + A[startOfSecondPart+1]

startOfSecondPart ← startOfSecondPart+2

end while

return A

b) The time complexity of the algorithm, in terms of Big-O, is **O(n)**.

c) The space complexity of the algorithm, in terms of Big-O, is **O(1)** (constant function), because in the algorithm I created 3 new variables, and the array that is passed into the function has already been allocated in the memory (no new memory is required).

Question 2

a) $2n^5\log^n$ is $O(n^7\log^n)$

the statement is true. However, it is not a good one, more reasonable O() should be:

$2n^5 <= 2n^5$ for n>=0

$\log^n <= \log^n$ for n>=1

therefore, $2n^5\log^n <= 2n^5\log^n$ for n>=1, **which means $2n^5\log^n$ is $O(n^5\log^n)$.**

b) $10^8n^2+5n^4+7000n^3+n$ is $\Theta(n^6)$

the statement is false.

First, we have to find O()

$10^8n^2 \quad <= 10^8n^4$ for n>=0

$5n^4 \quad <= 5n^4$ for n>=0

$7000n^3 <= 7000n^4$ for n>=0

$n \quad <= n^4$ for n>=0

therefore, $10^8n^2+5n^4+7000n^3+n <= (10^8+7006)n^4$, the function is $O(n^4)$

Next, we need to find $\Omega()$

$10^8n^2+5n^4+7000n^3+n >= n^4$, the function is $\Omega(n^4)$

Thus, the function is $\Theta(n^4)$

c) n^n is O(n!)

this statement is false.

$n^n = n*n*n*...*n$

$n! = n*(n-1)*(n-2)*(n-3)*...1$

since both of them have n multiplication operations, from above we can see that $n^n >= n!$

for any positive integer, therefore n^n cannot be O(n!) by the definition of Big-O. **Instead,**

it is $O(n^n)$.

d) $0.01n^3+0.0000001n^7$ is $\Theta(n^3)$

the statement is false.

First, we have to find O()

$0.01n^3 \quad <= 0.01n^7$ for n>=0

$0.0000001n^7 <= 0.0000001n^7$ for n>=0

therefore, $0.01n^3+0.0000001n^7 <= 0.0100001n^7$, the function is $O(n^7)$

Next, we need to find $\Omega()$

$0.01n^3+0.0000001n^7 >= 0.0000001n^7$, the function is $\Omega(n^7)$

Thus, the function is $\Theta(n^7)$

e) $n^2+0.0000001n^5$ is $\Omega(n^3)$

the statement is true. However, it is not a good one, more reasonable Ω () should be:

$n^2+0.0000001n^5 >= 0.0000001n^5$ therefore, **$n^2+0.0000001n^5$ is $\Omega(n^5)$**

f) n! is $\Omega(2^n)$

the statement is true, but not a good one. Because $n! >= 2^n$ (for n>3). But when we choose the value of Big-Omega, we should choose the biggest one in the hierarchy order, which means **n! should be $\Omega(n!)$**

Question 3

a) Algorithm MyAlgorithm(A, n)

Input: Array of integer containing n elements

Output: Possibly modified Array A	(# of operations)
done ← true	1
j ← 0	1
while j ≤ n - 2 do	n
if A[j] > A[j + 1] then	n
swap(A[j], A[j + 1])	n
done ← false	n
j ← j + 1	n
end while	
j ← n - 1	1

```
while j ≥ 1 do                              n
   if A[j] < A[j - 1] then                  n
      swap(A[j - 1], A[j])                  n
      done ← false                          n
   j ← j - 1                                n
end while
if ¬ done                                   1
MyAlgorithm(A, n)                           n^2
else
return A                                    1
```

we estimated the algorithm has $n^2+10n+5$ operations, it is **$O(n^2)$ and $\Omega(n^2)$** because $n^2+10n+5 >=10n^2$

b) when A= (4, 10, 5, 1, 3) n=5 the resulting A will be A= (1, 3, 4, 5, 10)

c) MyAlgorithm will sort the elements of the input array with increasing order. For a given array with n elements, after using the algorithm, A[0]<=A[1]<=A[2]<=…<=A[n].

d) MyAlgorithm can be improved easily by eliminating the second while loop. Because it is just a repetition of the first while loop, except it iterates the array from the last element.

e) MyAlgorithm is a linear recursive function, because it makes one recursive call each time it invoked. And it is tail-recursive because it makes the recursive call as its last step, and the method who calls itself does not wait for anything return from the recursively called function.