CSCI203 - Assignment 1

Question 1

$1^n < lg(n+10)^2 < n^{\frac{4}{8}} < 2^{lglgn} + \sqrt[3]{n} < nlgn + 2n^2 < \sum_{k=1}^{n} k < 4^{lgn} < n^3 < 16^n < (n^3+3)!$

Question 2

static int myMethod(*A*[], *n*) {	
$count \leftarrow 0$	1
for $i \leftarrow 0$ *to* $n-1$ {	n + 1
for $j \leftarrow i$ *to* $n-1$ {	(n(n+1)/2)
$count \leftarrow count + A[j]$	(n(n+1)/2) - 1
$k \leftarrow 1$	(n(n+1)/2) - 1
while ($k < n^2$) {	$n^2(n^2+1)$
if (j%2 == 0) {	n^4
$count = count + 1$	a
}	
k + +	n^4
}	
}	
}	
return count	1
}	

Runtime Complexity:

T(n) = 1 + (n + 1) + $\frac{n(n+1)}{2}$ + $2\left(\frac{n(n+1)}{2} - 1\right) + n^2(n^2+1) + 2n^4 + a + 1$

T(n) = $1 + (n+1) + \left(\frac{n^2}{2} + \frac{n}{2}\right) + (n^2 + n - 2) + (n^4 + n^2) + 2n^4 + a + 1$

T(n) = $3n^4 + \frac{5n^2}{2} + \frac{5n}{2} + 1 + a$

Hence, the runtime complexity of the algorithm “myMethod()” is $\Theta(n^4)$

Question 3

(i) T(n) = T(n-1) + O(n)

(ii) T(n) = T(n-1) + O(n)

T(n) = (T(n−2)+O(n))+O(n)

T(n) = ((T(n−3)+O(n))+O(n))+O(n)

T(n) = T(1) + $\sum_{k=1}^{n-1} O(n)$ = 1 + $\sum_{k=1}^{n-1} O(n)$ = O(n^2)

Hence the upper bound complexity of the recursive Insertion sort implemented in part (i) is O(n^2)

Question 4

(a) a=2, b=2,c=2,p=2

$\frac{a}{b^c} = \frac{2}{4} = \frac{1}{2} < 1$, p≥0

Therefore T(n) = Θ$(n^c log_b^p n)$ = Θ$(n^2 log_2^2 n)$

(b) a=4,b=2,c=2,p=1/2

$\frac{a}{b^c} = \frac{4}{4} = 1$, p>-1

Therefore T(n) = Θ$(n^{log_b a} log_b^{p+1} n)$ = Θ$(n^{log_2 4} log_2^{\frac{3}{2}} n)$ = Θ$(n^2 log_2^{\frac{3}{2}} n)$

(c) a=4,b=4,c=1,p=2

$\frac{a}{b^c} = \frac{4}{4} = 1$, p>-1

Therefore T(n) = Θ$(n^{log_b a} log_b^{p+1} n)$ = Θ$(n^{log_4 4} log_4^3 n)$ = Θ$(n log_4^3 n)$

(d) a=4,b=2,c=3,p=0

$\frac{a}{b^c} = \frac{4}{8} = \frac{1}{2} < 1$, p≥0

Therefore T(n) = Θ$(n^c log_b^p n)$ = Θ$(n^3 log_2^0 n)$ = Θ(n^3)

(e) The recurrence relation does not conform to the format required to be solved using master theorem.

Hence, the recurrence relation cannot be solved using master theorem.

Thus, the running time complexity is solved using expansion and substitution.

T(n) = T(n-1) + 4(n-1) + c
= T(n-2) + 4(n-2) + 4(n-1) + c
= T(n-3) + 4(n-3) + 4(n-2) + 4(n-1) + c
= T(n-3) + … + 4(n-3) + 4(n-2) + 4(n-1) + c
T(n) = T(n-k) + … + 4(n-3) + 4(n-2) + 4(n-1) + c

Substitute k = n:
T(n) = T(0) + … + 4(n-3) + 4(n-2) + 4(n-1) + c
= 1 + … + 4(n-3) + 4(n-2) + 4(n-1) + c

Therfore T(n) = $2n^2 + c$

Question 5

(a) define a quick_sort function taking in three arguments, array A, the lower (0) and upper bounds (n-1) of the array

- check if the lower bound is smaller than the upper bound
  - Using k as the pivot element in the array, arrange the elements smaller than the pivot are on the left and elements greater than the pivot are on the right
  - Perform a recursive call on quick_sort on the left and right of the pivot

(b) θ(n log(n)) because that is the runtime complexity for a quick sort