1. (10 Points) Consider the following pseudo-code and analyze the best and the worst-case running time of this by providing the form of the function the running time takes in each of the cases (Analyze statement-by-statement and use the length of the array 𝐴 as 𝑛):  
Algorithm 1 AdjacentDuplicates(A)  
for 𝑖 = 0 𝑡𝑜 𝐴.𝑙𝑒𝑛𝑔𝑡ℎ −2 do n-1  
if 𝐴[𝑖] == 𝐴[𝑖 +1] then 1,but runs (n-2) times  
return 1 1,but may runs (n-2) times   
end if  
end for  
return 0 1

In the best case scenario, the first element satisfies the equation,so the running time could be O(1).

In the worst case scenario, none of elements of arrayA satisfy the condition, so the algorithm must iterate through the entire array, so the running time should be O(n).

2. (10 Points) Consider the following pseudo-code and analyze the best and the worst-case running time of this by providing the form of the function the running time takes in each of the cases (Analyze statement-by-statement and use the length of the array 𝐴 as 𝑛):

Algorithm 2 Linear-Search(A,v)

i= NIL 1

for j = 1 to A.length do 1, but runs n times

if A[j] = v then 1, but may runs n times

i=j 1

return i 1

end if

end for

return I 1

In the best case scenario, the first element equals the value v ,so the running time could be O(1).

In the worst case scenario, none of elements of arrayA satisfy the condition, so the algorithm must iterate through the entire array, so the running time should be O(n).

3. (5 points) Find the upper bound for 𝑓s (𝑛) = 𝑛4 +10𝑛2 +5. Prove your answer by giving values for the constants 𝑐 and 𝑛0. Choose the smallest integral value possible for 𝑐.

𝑓 (𝑛) = 𝑛4 +10𝑛2 + 5 < cn4

1 + 10/n2 + 5/n4 < c

So c >= 16, n0 = 1

Suppose c = 2,then

𝑛4 + 10𝑛2 + 5 < 2n4

10𝑛2 + 5 < n4

10 + 5/n2 < n2

So,c =2,n0 = 4

4. (10 points) Find an asymptotically tight bound for 𝑓 (𝑛) = 3𝑛3 −2𝑛. Prove your answer by giving values for the constants 𝑐1, 𝑐2, and 𝑛0. Choose the tightest integral values possible for 𝑐1 and 𝑐2.

𝑓 (𝑛) = 3𝑛3 −2𝑛 ∈ θ (𝑛3)

The upper bound:

3𝑛3 −2𝑛 ≤ 3𝑛3 ( n ≥ 1)

The lower bound :

3𝑛3 −2𝑛 ≥ 𝑛3

2 𝑛3 ≥ 2𝑛 ( n ≥ 1)

So,c1 = 1,c2 = 3, n0 =1, 3𝑛3 −2𝑛 ∈ θ (𝑛3)

5. (5 points) Is 3𝑛 −4 ∈Ω 𝑛2 ? Prove your response.

If 3𝑛 −4 ∈ Ω (𝑛2), then

cn2 < 3n – 4 < 3n (∀ n> n0)

cn2 - 3n < 0

n(cn -3) < 0

cn - 3 < 0

n < 3/c

Since n can grow to infinity, it is impossible to find a

positive constant c such that n is bounded above by the

constant 3/c. Therefore, the c we need to find does not

exist. Therefore 3𝑛 −4 ∉ Ω(n²).

6. (20 points) Express the complexity of the following functions with the most appropriate notation

int function1(int n) {

int count = 0;

for (int i=n/2;i<=n; i++){ n/2

for (int j= 1;j<=n;j\*=2) { log2n

count++;

}

}

return count;

}

So the running time will be (n/2) log2n, the time complexity is θ(nl0g(n))

--------------------------------------------------------------------------------------------

int function2(int n) {

int count = 0;

for (int i=1; i\*i\*i<= n; i++){

count++;

}

return count;

}

So the running time will be , the time complexity is θ()

--------------------------------------------------------------------------------------------

int function3(int n) {

int count =0;

for (int i= 1; i <= n; i++) { n

for (int j= 1; j <= n; j++) { n2

for(int k=1;k<= n; k++) { n3

count++;

}

}

}

return.count;

}

So the running time will be n3, the time complexity is θ(n3)

--------------------------------------------------------------------------------------------

int function4(int n) {

int count = 0;

for(int i= 1;i<= n; i++) { n

for (int j = 1; j <= n; j++){

count++;

break; 1

}

}

return count;

}

The inner loop will execute only once because of break, so the running time will be n, the time complexity is θ(n)

--------------------------------------------------------------------------------------------

int function5(int n) {

int count = 0;

for (int i= 1;i <= n; i++) { n

count++;

}

for (int j= 1; j<= n; j++) { n

count++;

}

return count;

}

The running time will be 2n, the time complexity is θ(n)

--------------------------------------------------------------------------------------------

7. (10 points) Find an asymptotically tight bound for 𝑓 (𝑛) = 𝑛2/2− 7𝑛. Prove your answer by giving values for the constants 𝑐1, 𝑐2, and 𝑛0. Choose the tightest integral values possible for 𝑐1 and 𝑐2 [Hints: Solve it for the smallest possible integer 𝑛0].

0 ≤ c1(g(n)) ≤ f(n) ≤ c2(g(n))

0 ≤ c1n2≤ 𝑛2/2− 7𝑛 ≤ c2n2

c1 ≤ 1/2 – 7/n ≤ c2

Suppose c2=1/2, c1=1/6, n0 = 21, 𝑛2/2− 7𝑛∈ θ (𝑛2)

8. (5\*4 = 20 Points) Solve the following recurrence relations (using the backward substitution method with a demonstration of all the steps):

a) 𝑥(𝑛) = 3𝑥(𝑛 − 1) for 𝑛 > 1, 𝑥(1) = 4

x(n-1) = 3x(n-1-1) x(n) = 3(3x(n-1-1)) = 9x(n-2)

x(n-2) = 3x(n-2-1) x(n) =9 (3x(n-2-1)) = 27x(n-3)

…

x(n) = 3kx(n-k) n-k =1 x(n) =3n-1x(1) = 4\*3 n-1

--------------------------------------------------------------------------------------------

b) 𝑥(𝑛) = 𝑥(𝑛 − 1) + 𝑛 for 𝑛 > 0, 𝑥(0) = 0

x(n-1) = x(n-1-1) + (n-1) = x(n-2) + (n-1) x(n) = x(n-2) + n + (n-1)

x(n-2) = x(n-2-1) + (n-2) = x(n-3) + (n-2) x(n) = x(n-3) + n + (n-1)+ (n-2)

…

x(n) = x(n-i) + n + (n-1) +… + (n-(i-1))

n = i , x(n) = 0 + 1 +2 + … + n = (1+n)n/2 = O(n2)

--------------------------------------------------------------------------------------------

c) 𝑥(𝑛) = 𝑥(𝑛/2) + 𝑛 for 𝑛 > 1, 𝑥(1) = 1

x(n/2) = x(n/4) + (n/2) x(n) = x(n/4) + (n/2) + n

x(n/4) = x(n/8) + (n/4) x(n) = x(n/8) + (n/4) + (n/2) + n

…

x(n) = x(n/2k) + (n/2k-1) +( n/2k-2) + … + (n/2k-logn)

n = 2k,k = logn,x(n) = x(n/2logn) +n(1/2 + …+ 1/2k-1) = 1 + n = O(n)

--------------------------------------------------------------------------------------------

d) 𝑥(𝑛) = 𝑥(𝑛/3) + 1 for 𝑛 > 1, 𝑥(1) = 1

x(n/3) = x(n/9) + 1 x(n) = x(n/9) + 2

x(n/9) = x(n/27) + 1 x(n) = x(n/27) + 3

…

x(n) = x(n/3k) + k

n = 3k k =1ogn, x(n) = 1 + logn = O(logn)

9. (10 Points) For each function below, compute the recurrence relation for its running time and then use the Master Theorem to find its complexity by specifying the different terms of the term explicitly:

int f(int arr[], int n) {

if (n == 0) {

return 0;

}

intsum =0;

for (int j=0;j< n; ++j) {

sum += arr[j]; f(n) = n d=1

}

return f(arr,n / 2) + sum + f(arr, n / 2); a = 2, b = 2

}

T(n) = 2T(n/2) + n, d = 1, bd = 2 = a, so T(n) ∈ θ (𝑛dlogbn) = θ (𝑛log2n) = θ(nlogn)

--------------------------------------------------------------------------------------------

void g(int n, int arrA[], int arrB[]){

if(n == 0){

return;

}

for(int i =0;i <n; ++i) {

(for(int j=0;j<n; ++j){

arrB[j] += arrA[i]; f(n) = n2 d=2

}

}

g (n / 2，arrA,arrB); a = 1, b = 2

}

T(n) = T(n/2) + n2, d = 2, bd = 4 ＞ a, so T(n) ∈ θ (𝑛d) = θ (𝑛2)

10. (2\*5 = 10 points) Use the Master Theorem to find the complexity of each of the following recurrence relations (showall the steps and the values of different terms to apply the theorem):

a) 𝑇 (𝑛) = 4𝑇(n/2) + n2

a = 4, b = 2, d = 2, a = bd, so T(n) ∈ θ (𝑛dlogbn) = θ (𝑛2log2n) = θ (𝑛2logn)

b) 𝑇 (𝑛) = 3𝑇(n/3) +

a = 3, b = 3, d = 1/2, a＞bd, so T(n) ∈ θ (𝑛logba) = θ (𝑛)

11. (2\*10 = 20 points) Use the Recursion tree method to find appropriate notation for the complexity of each of the following recurrence relations (draw the recursion tree and show all the steps to your solution):

a) 𝑇 (𝑛) = 7𝑇(n/5)+ 𝑐𝑛3

cn3

c(n/5)3  c(n/5)3 c(n/5)3 c(n/5)3 c(n/5)3 c(n/5)3 c(n/5)3

7 𝑐(n/5)3 = cn3(7/53)

…………… 72 𝑐(𝑛/52)3 = cn3(7/53)2

73 𝑐(𝑛/53)3 = cn3(7/53)3

T(1) T(1) T(1) T(1) … … … T(1) 7k 𝑐(𝑛/5k)3 = cn3(7/53)k

T(n) = cn3(7/53)k = cn3 (7/53)k ∈ O (𝑛3)

b) 𝑇 (𝑛) = 5𝑇 (𝑛 − 6) + 𝑐

c

c c c c c 5c

ccccc ccccc ccccc ccccc ccccc 52c

………………..

T(0) T(0) T(0) T(0) … … … T(0) 5kc

T(n) = 5c + 52c +…+ 5kc ∈ θ (5n/6)

c