McMaster University  
SFWR ENG 2MD3 Winter 2020 Assignment 5  
Due: Sunday March 15, 2020 at 23:55

**Manuel Lemos, 400177763**

Question 1 (10 marks (3+3+4))

Use the definition of the fact that f(x) is O(g(x)) to show that:

1. is O(𝑥2)

Use K=7 and , it can found that

As a result, 7𝑥2 is 𝑂(𝑥2).

1. is

Use K = summation of element coefficients = 21 and ,

It can found that

As a result, is

1. is

Use K = summation of element coefficients = 1 and ,

It can found that

Alternatively, we can consider that, upon simplification, the fastest growing term is x.

As a result, is

Question 2 (10 marks (3+3+4))

Give as good a big-oh estimate of the following as possible



Upon simplification, the fastest growing term is

As a result, is

Dominant term from the left and right factor is and

Combining these dominant factors results in

As a result, is

Dominant term from the left and right factor is and

Combining these dominant factors results in

As a result, is

Question 3 (9 marks (5+2+2))

Provide answers for:

1. Give the names of five separate algorithms whose analysis reveals that their running times are in the following respective five classes: 0(log n), 0{n), 0(n log n), 0(n2) and 0(2").

0(log n) Binary Search

0{n) Sequential Search

0(n log n) Merge Sort

0(n2) Selection Sort

0(2n) Towers of Hanoi

1. A) When is it inappropriate to apply the conclusions normally derived from O-notation analysis to an algorithm's running-time equation?

When considering small input sizes

B) What can you do to find optimal algorithms to run on small-sized problems?

For small sized problems, it is best to dynamically measure the algorithm and see where the code expends the most time and resources. Given this data, one can then tune the algorithm, and repeat these processes of measuring and tuning until an acceptable level of optimization has been achieved.

Question 4 (6 marks)

Analyze the following abstract program strategy for selection sorting. Give both the running time equation and the O-notation that results from your analysis.

void SelectionSort(SortingArray A, int n)

{

if (n>1) {

(Find the position, p, of the smallest item in A[0:n-1])

(Exchange A[p] and A[n-1] to make A[n-1] the smallest item in A[0:n-1])

(Sort the rest of the array A[0:n-2] by calling SelectionSort(A,n-1))

}

}

The cost of finding p in A[0:n-1] is where = running time per loop, and = loop setup

The cost of exchanging A[p] and A[n-1] is constant and will be named

From here, and will be combined to form the arbitrary constant

The recursion results in:

where is the result of

As a result, the selection sort algorithm is O(n2)