|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Kingdom of Saudi Arabia**  **Ministry of Education**  **University of Jeddah**  **College of Science and Computer Engineering** |  | **المملكة العربية السعودية**  **وزارة التعليم**  **جامعة جدّة**  **كلية علوم و هندسة الحاسب** | |  |  |

**CCCS 314 – Design and Analysis of Algorithms**

**LAB 2**

**Topics:**

1. **Algorithm analysis**
2. **Asymptotic notations and basic efficiency classes**
3. **Mathematical analysis of non-recursive algorithms**
4. **Mathematical analysis of recursive algorithms**

**Total Marks: 2**

**Student Name: Bassam Al-gamdi**

**Student ID: 2141362**

**Marks:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exercises | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
| Allocated | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.4 | 2 |
| Obtained |  |  |  |  |  |  |  |  |  |  |
| **CLO, PLO** | 1.2, K2 | 1.2, K2 | 1.1, K1 | 1.1, K1 | 1.1, K1 | 1.1, K1 | 1.1, K1 | 1.2, K2 | 1.2, K2 |  |

**CLO** **Marks:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | CLO1.1, K1 | CLO1.2, K2 | Total |
| Allocated | 1 | 1 | 2 |
| Obtained |  |  |  |

**(Algorithm Analysis)**

1. For each of the following algorithms, determine the basic operation:
2. Computing the sum of *n* numbers.

Answer: The basic operation for computing the sum of n numbers is addition. Each number in the list is added to the running total.

1. Computing *n*!

Answer: The basic operation for computing n! (n factorial) is multiplication. The product of all the integers from 1 to n is calculated.

1. Finding the largest element in a list of *n* numbers.

Answer: The basic operation for finding the largest element in a list of n numbers is comparison. Each element in the list is compared to the current maximum value, and if it is larger, it becomes the new maximum value.

1. For each of the following functions, indicate how much the function’s value will change if its argument is increased fourfold.
2. 

the value of the function will increase by a constant factor if the argument is increased fourfold. the value of the function will increase by a constant factor of log(4) = 2 (since log(4n) = log(4) + log(n)).

1. 

the value of the function will increase by a factor of 2 if the argument is increased fourfold. the value of the function will increase from √n to √(4n) = 2√n.

1. 

the value of the function will increase by a factor of 16 if the argument is increased fourfold. the value of the function will increase from n² to (4n)² = 16n².

**(Asymptotic Notations and Basic Efficiency Classes)**

1. Indicate whether the first function of each of the following has a smaller, same, or larger order of growth (to within a constant multiple) than the second function.
2.  and 

The function n(n+1) has the same order of growth as 200n², since both functions are quadratic and differ only by a constant factor.

1.  and 

The function nlogn has a larger order of growth than log n², since nlogn grows faster than logarithmic functions.

For each of the following functions, indicate the class  the function belongs to.

the dominant term is 10n^2, and therefore the function belongs to the square root of n squared, or O(√n²), which simplifies to O(n).

The dominant term is (n+2) ^2 log(n), and therefore the function belongs to

O (n^2 log n).

1. Order the following functions according to their order of growth (from the lowest to the highest)

, , , ,

1. 5 log((n+100) ^10) = O (log n)
2. 0.0001n^4 + 3n^3 + 1 = O(n^4)
3. (n-1)! = O(n^n)
4. 2^(2n) = O(2^n)
5. 3^n = O(3^n)

**(Mathematical Analysis of Non-recursive Algorithms)**

1. Compute .

S = (n/2) (a + l),

where S is the sum, n is the number of terms, a is the first term, and l is the last term.

In this case, a = 1, l = 999, and the common difference is 2.

Therefore, we can find the number of terms n as follows:

l = a + (n-1) d

999 = 1 + (n-1)2

n = 500

Now we can substitute these values into the formula:

S = (500/2) (1 + 999)

S = 250(1000)

S = 250000

Therefore, the sum of the sequence 1+3+5+7+...+999 is 250000.

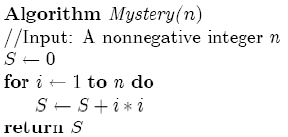
1. Find the order of growth of the following sums:

Simplifying this expression, we can see that the highest order term is n^5, and therefore the order of growth of the sum is O(n^5).



Simplifying this expression, we can see that the highest order term is n^3, and therefore the order of growth of the sum is O(n^3).

1. Consider the following algorithm.



1. What does this algorithm computes?

This algorithm computes the sum of the squares of the first n non-negative integers.

1. What is the basic operation?

The basic operation is the multiplication of two integers (i \* i).

1. How many times is the basic operation executed?

The basic operation is executed n times.

1. What is the efficiency class of this algorithm?

The efficiency class of this algorithm is O(n).

1. Suggest an improvement, or a better algorithm and indicate its efficiency class.

The improved algorithm is as follows:

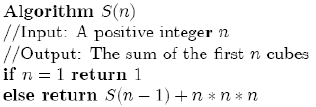
BetterMystery(n)

//Input: A no negative integer n

return n\*(n+1) (2n+1)/6

**(Mathematical Analysis of Recursive Algorithms)**

1. Consider the following recursive algorithm for computing the sum of the first n cube: .



1. Set up and solve a recurrence relation for the number of times the algorithm’s basic operation is executed.

Let T(n) be the number of times the basic operation (i.e., n^3 multiplication) is executed by the algorithm S(n). Then we have:

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

The base case is T (1) = 1.

We can solve this recurrence relation by repeatedly substituting T(n-1) in the previous equation to obtain:

T(n) = T(n-2) + 1 + 1 = T(n-3) + 1 + 1 + 1 = ... = T (1) + (n-1) = n

Therefore, the number of times the basic operation is executed is O(n).

1. How does this algorithm compare with the straightforward non-recursive algorithm for computing this function?

The straightforward non-recursive algorithm for computing S(n) is to simply compute the sum of the first n cubes using the formula S(n) = n(n+1)/2) ^2. This formula is derived by using the formula for the sum of the first n integers and then squaring it. This algorithm has a time complexity of O (1) since it only involves a few arithmetic operations.

Comparing this with the recursive algorithm, we see that the recursive algorithm has a time complexity of O(n), which is less efficient than the non-recursive algorithm. However, the recursive algorithm may be useful in situations where the formula for S(n) is not known or easily computable.