**Exam II**

Ramsey Numbers Final Exam

# Problem 1:

State the asymptotic relationship between the functions and , as in , where may be , , . You must justify your answer by showing your work using the

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  | *formula* |
|  | *substitute values* |
|  | *remove smaller order terms* |
|  | *cancel like terms* |
|  | *solve* |

Since , the statement is true.

Since , the statement is true.

Since , the statement is true.

|  |  |
| --- | --- |
|  | *formula* |
|  | *substitute values* |
|  | *cancel like terms* |
|  | *limit rule* |
|  | *limit rule* |
|  | *limit rule* |
|  | *limit rule* |
|  | *solve* |

Since , the statement is true.

Since , the statement is false.

Since , the statement is false.

|  |  |
| --- | --- |
|  | *formula* |
|  | *substitute values* |
|  | *base conversion* |
|  | *log rule* |
|  | *cancel like terms* |
|  | *limit rule* |

Since , the statement is true.

Since , the statement is true.

Since , the statement is true.

|  |  |
| --- | --- |
|  | *formula* |
|  | *substitute values* |
|  | *factor* |
|  | *cancel like terms* |
|  | *exponent rule* |
|  | *exponent rule* |
|  | *subtraction* |
|  | *limit rule* |
|  | *limit rule* |
|  | *limit rule* |

Since , the statement is false.

Since , the statement is true.

Since , the statement is false.

# Problem 2:

Using the master theorem for solving recurrences, state the Big-O value for the following recurrences. If it is inappropriate to use the master method, then state this fact instead.

|  |  |
| --- | --- |
|  | |
| for some constant |  |
|  |  |
| for some constant  for some constant  and all sufficiently large |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| Case 3 |
| is polynomially greater than |
| : The work completed in the sub-roots, which are not leaves, of the recursion tree is the primary determinant of the computational complexity. As is polynomially greater than , our regularity condition is met. |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| Case 2 |
| and are polynomially equivalent |
| : The work is evenly distributed between the leaves and sub-roots of the recursion tree. |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| No Master Theorem Case Applicable |
| We can not compare theandasis *undefined*. As such, the master theorem can not be applied here. |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| Case 3 |
| is polynomially greater than |
| : The work completed in the sub-roots, which are not leaves, of the recursion tree is the primary determinant of the computational complexity. As is polynomially greater than , our regularity condition is met. |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| Case 1 |
| is polynomially greater than |
| : The work completed in the sub-roots, which are not leaves, of the recursion tree is the primary determinant of the computational complexity. As is polynomially greater than , our regularity condition is met. |

# Problem 3:

For the following algorithm, (3A) give a precise line count for every numbered line of code in terms of . Use indicator functions if/when needed. When you've completed deriving the line frequencies, (3B) give the efficiency of the algorithm in terms of Big-O.

## **Code**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | void sort(int A[], int n) { | | | | | | | | |
| *1* |  | |  | int i **=** 0**,** j**,** s**;** | | | | | | |
| *2* |  | |  | **while(**i **<** n**-**1**){** | | | | | | |
| *3* |  | |  | |  | s = i**;** | | | | |
| *4* |  | |  | |  | j = i+1**;** | | | | |
| *5* |  | |  | |  | **while(**j **<** n**){** | | | | |
| *6* |  | |  | |  | |  | **if(**A**[**j**]** **<** A**[**s**]){** | | |
| *7* |  | |  | |  | |  | |  | s = j**;** |
|  |  | |  | |  | |  | } | | |
| *8* |  | |  | |  | |  | j **+=** 1**;** | | |
|  |  | |  | |  | } | | | | |
| *9* |  | |  | |  | A**[**i**]** **=** A**[**s**];** | | | | |
| *10* |  | |  | |  | i **+=** 1**;** | | | | |
|  |  | |  | } | | | | | | |
|  |  | } | | | | | | | | |

## **Frequency Counts and Big O Notation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Line | Code | | | | | Count |  |
|  | void sort(int A[], int n) { | | | | | ----- | ----- |
| *1* |  | int i **=** 0**,** j**,** s**;** | | | | 1 |  |
| *2* |  | **while(**i **<** n**-**1**){** | | | |  |  |
| *3* |  |  | s = i**;** | | |  |  |
| *4* |  |  | j = i+1**;** | | |  |  |
| *5* |  |  | **while(**j **<** n**){** | | |  |  |
| *6* |  |  |  | **if(**A**[**j**]** **<** A**[**s**]){** | |  |  |
| *7* |  |  |  |  | s = j**;** |  |  |
|  |  |  |  | } | | ----- | ----- |
| *8* |  |  |  | j **+=** 1**;** | |  |  |
|  |  |  | } | | | ----- | ----- |
| *9* |  |  | A**[**i**]** **=** A**[**s**];** | | |  |  |
| *10* |  |  | i **+=** 1**;** | | |  |  |
|  |  | } | | | | ----- | ----- |
|  | } | | | | | ----- | ----- |

### Line 1: int i **=** 0**,** j**,** s**;**

#### Count:

* + This line initializes three integer variables i, j, and s and is invoked only once during program execution; therefore, the count is .
  + The time complexity is constant, , because this line has a fixed execution time.

### **Outer While Loop**

#### Line 4: **while(**i **<** n**-**1**)** **{**

##### Count:

###### *Outer While Loop Code Analysis*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **while(**i **<** n**-**1**){** | | | | | |
|  | s = i**;** | | | | |
|  | j = i+1**;** | | | | |
|  | **while(**j **<** n**){** | | | | |
|  | |  | **if(**A**[**j**]** **<** A**[**s**]){** | | |
|  | |  | |  | s = j**;** |
|  | |  | } | | |
|  | |  | j **+=** 1**;** | | |
|  | } | | | | |
|  | A**[**i**]** **=** A**[**s**];** | | | | |
|  | i **+=** 1**;** | | | | |
| } | | | | | |

* + We know  *(line 1),* assume

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration |  | Condition | | Result |
| 1 |  |  | true | loop executes |
| 2 |  |  | true | loop executes |
| 3 |  |  | true | loop executes |
| 4 |  |  | false | loop terminates |

* + When , the while loop condition is executed for 4 iterations, or . This is because the loop condition must perform an additional check for when the condition fails:

However, the inside of the while loop is executed for 3 iterations, , because the loop body is only executed when the condition is true. The exact count value varies depending on the value of .

* + The time complexity is equivalent to the count of , and represents the linear relationship between the input size and the execution time.