Questions and Exercises to work out and turn in:

Grading Guidelines (See Appendix At The Bottom Of The document):

A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link[[1]](#footnote-1)** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Exercise (100 points) Analyze A Silly Algorithm

Consider the problem to compute the sum of the elements of an upper triangular :

A white paper with black text

Description automatically generated

Below is an awkward algorithm to solve the above problem.

**ComputeSumUpperTriangular(U,n)**

**inputs:**  is an nxn matrix. is the dimension of the matrix

**output:** a real number equal to

1: sum = 0

2: for i = n to 1

3: psum = 0

4: for j = i to n

5: psum = psum + U[i,j]

6: sum = sum +psum

7: return sum

All the questions in this exercise are related to the **ComputeSumUpperTriangular(L,n)** algorithm. The objective of this **exercise is to explore whether the asymptotic time complexity will change if we count different “actions”.**

1. **(40 points) Comparison** Action (Lines 2 and 4).

In this case, we count the total number of *comparisons* performed by the “for loops statements” (Lines 2 and 4 only). Answer the following questions to determine the total number of comparisons performed by the algorithm.

* 1. (2 points) How many comparisons in total are performed by the “*for loop*” statement in Line 2 during the execution of the algorithm? (See on the appendix how Student 3 gets full credit)

answer here .....

In the pseudocode within the “for” loop on Line 2, the comparison is made for each iteration starting from i = n to i = 1. This means that it will compare i to 1 on each iteration to determine if the loop should continue. Since it's counting down, there will be “n” comparisons as it goes from n to 1.This means there are n comparisons performed by the "for loop" statement in Line 2 during the execution of the algorithm.

* 1. (20 points) Let us call ti the number of comparisons performed by the inner “for loop” in Line 4 for a given value of i. Fill in this table (**Justify** how you find ti **using exactly the same steps and sentence pattern shown below for i = n**):

|  |  |
| --- | --- |
| i | ti |
| **n** | when i = **n**, the inner for loop is "for j = n to **n**", the number of comparisons is .... one because it will check if j is less than or equal to n, and j will start at n. |
| n-1 | when i = n-1, the inner for loop is "for j = n-1 to n the number of comparisons is .... 2 because it will check if j is less than or equal to n for “j = n-1 and j = n”. |
| n-2 | *(follow the same patter as above)* when i = n-2, the inner for loop is “......... “, the number……When i = n-2, the inner for loop is "for j = n-2 to n", the number of comparisons is 3 |
| k | when i = k, the inner for loop is “....”, the number ….When i is equal to k, the inner for loop is "for j = k to n", the number of comparisons is “n - k + 1”. |
| 1 | when i = 1, the inner for loop is .... "for j = 1 to n", the number of comparisons is n |

* 1. (8 points) Based on b, express the **total** number of comparisons performed by the inner “for loop” in Line 4 only during the execution of the algorithm. (See how Student 3 gets full credit)

answer here ...

The total number of comparisons performed by the inner "for loop" can be calculated by summing up all of the t sub i. This will form a sequence from 1 to n. The total number of comparisons is the sum of the series 1 + 2 + ... + n. The formula ends up being n(n + 1)/2 which we were shown in class as well.

* 1. (8 points) Express the function fc(n) that represents the overall total number of comparisons performed by the “for loops” statements in Lines 2 and 4 during the execution of the algorithm. (See how Student 3 gets full credit)

answer here ...

In order to calculate the total number of comparisons from the function “fc(n)” we need to calculate the sum of both the inner and outer loops. This ends up producing a formula

“fc(n) = n + n(n + 1)/2”. This will calculate both the inner and outer loop comparisons.

* 1. (2 point) The function fc(n) **grows** like \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. (See how Student 3 gets full credit)

The function fc(n) grows like like n2 mostly considering everything but the constants. This is due to when looking at it from the general asymptotic growth, we will only need to zone in on the term with the highest power, as this shows the exponential growth of the function.

**From now on throughout the semester, you must answer like Student 3 in order to get full credit.**

1. **(10 points) Addition** Action (only on **Line 6**)

In this case, we count the total number of *additions* performed by Line 6 during the execution of the algorithm. Answer the following questions to determine the number of additions performed by Line 6 during the execution of the algorithm*.*

* 1. (7 points) Let us call ai the number of *additions* performed by Line 6 for a given value of i. Fill in this table ((**Justify** how you find ai **using exactly the same steps and patterns shown below for i = n**)):

|  |  |
| --- | --- |
| i | ai |
| n | When i = n, Line 6 performs ...... addition(s) for one iteration. When i = n, Line 6 will perform 1 addition for one iteration because it sums psum (which is the sum of elements on the nth row) to sum exactly once. |
| n-1 | When i = n-1, Line 6 performs ...... addition(s) for one iteration. When i = n-1, Line 6 also performs 1 addition for one iteration, as it adds the sum of elements on the (n-1)th row to sum. |
| n-2 | When i = n-2, Line 6 performs ...... addition(s) ... When i = n-2, Line 6 performs again 1 addition for one iteration, adding the sum of elements on the (n-2)th row to “sum”. |
| k | When i = k, Line 6 performs ...... addition(s) ... Line 6 again performs 1 addition for one iteration, adding the sum of elements on the kth row to sum. |
| 1 | When i = 1, Line 6 performs ...... addition(s) ... When i = 1, Line 6 will again perform 1 addition for one iteration, adding the sum of elements on the first row to sum. |

Line 6 adds psum to sum for each iteration of the outer loop. Since the outer loop runs n times, “ai” is always 1 regardless of the value of i.

* 1. (2 points) Express the function fa(n) that represents the overall total number of additions performed by Line 6 during the execution of the algorithm*.*

The function fa(n) = n simply equals n, as it represents the total number of additions completed by Line 6 of the pseudocode. This adds the partial sum to the total sum exactly one time during each of the “n” iterations of the outer loop.

* 1. (1 point) The function fa(n) **grows** like \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. The function fa(n) grows in a simple linear way since it is equal to n. This means that the functions grows like that of “n”.

1. **(30 points) Addition** Action (only on **Line 5**)

In this case, we count the total number of *additions* performed on Line 5 by the execution of the algorithm. Answer the following questions to determine the total number of *additions (Line 5)* during the execution of the algorithm

* 1. (20 points) Let us call mi the number of *additions* performed by Line 5 in the inner loop for a given value of i. Fill in this table (**Justify** how you find mi **using the exact same steps and sentence pattern shown for i = n**):

The number of additions mi by Line 5 for a given i is determined by the number of elements in each row of the upper triangular matrix, which is n - i + 1.

|  |  |
| --- | --- |
| i | mi |
| **n** | when i = n, the inner for loop is "for j = n to n", the number of additions is .... When i = n, the inner for loop has "for j = n to n", so the number of additions is 1 because there's only one element in the last row of the upper triangular matrix. |
| **n-1** | when i = n-1, the inner for loop is "for j = ... to ...", the number of additions is ... When i = n-1, the inner for loop has "for j = n-1 to n", the number of additions is 2 since there are two elements in the second-to-last row of the matrix. |
| n-2 | when i = n-2, the inner for loop is ... When i = n-2, the inner for loop has "for j = n-2 to n", the number of additions is 3, accounting for the three elements in that row of the matrix. |
| k | when i = k, the inner for loop is ... When i = k, the inner for loop has "for j = k to n", the number of additions is n - k + 1, corresponding to the number of elements in the kth row of the upper triangular section. |
| 1 | when i = 1, the inner for loop is .... When i = 1, the inner for loop has "for j = 1 to n", the number of additions is n, which is the total number of elements in the first row. |

* 1. (8 points) Express the function fm(n) that represents the overall total number of *additions* performed by Line 5 during the execution of the algorithm*.*

*The function fm(n) is the sum of the series 1 + 2 + ... + n, which again is calculated as n(n + 1)/2 like covered in the module videos. This will show the total number of additions performed by Line 5 throughout the entire execution of the algorithm.*

* 1. (2 points) The function fm(n) grows like \_\_\_\_\_\_\_\_\_\_\_\_\_\_. The function fm(n) shows quadratic growth pattern based on the highest power being 2. This quadratic nature is due to the increasing number of additions with each subsequent row of the upper triangular matrix as n increases.

1. **(14 points) Compare** the functions fc(n), fa(n), and fm(n) and discuss their “asymptotic growth”. Which function fc(n), fa(n), and fm(n) should you use to determine the time complexity of this algorithm.

**First when comparing the functions we should first look to see which has the highest exponential as this will show the relative growth. Since both fc(n) and fm(n) grow quadratically and dominate over the linear growth of fa(n), we can use either fc(n) or fm(n) for time complexity. Since there are additional additions in the fm(n) then this will make for a more accurate reflection of the time complexity.**

1. **(6 points) Space complexity**: Answer these questions assuming that we do not count the space taken by the inputs. Is this algorithm in-place? What is its space complexity? How does its space complexity “grow”?

In the given silly algorithm, all operations are performed within the input matrix space, and only a fixed number of variables are used: sum, psum, and loop indices i and j. This implies that the space requirement does not grow with the size of the matrix n. Hence, this algorithm is in-place, as it does not require extra space that scales with the input size.The space complexity of this algorithm is 1 because the amount of additional space needed does not depend on the input size n. The variables sum and psum are merely accumulators, and the loop indices do not require additional space that grows with n. Therefore, the space complexity's growth is constant, not impacted by larger input sizes.

**What you need to turn in:**

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.

**Reminder: How this assignment will be graded (See Appendix At The Bottom Of The document):**

* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

* **Student 1** answers: "9:48am". Student 1 will get 25 points.
* **Student 2**answers : "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
* **Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (**document is attached**).



Therefore, the time t = 108 miles/60mph \* 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".

1. See the appendix for what an obvious and clear link is. [↑](#footnote-ref-1)