**COMP 3270 Homework 1**

100 points. Due by **11:59pm (midnight) on Wednesday, June 8th, 2022**

Instructions:

1. This is an individual assignment. You should do your own work. Any evidence of copying will result in a zero grade and additional penalties/actions.
2. Submissions not handed on the due date and time **will not** be accepted unless prior permission has been granted or there is a valid and verifiable excuse.
3. Think carefully; formulate your answers, and then write them out concisely using English, logic, mathematics and pseudocode (no programming language syntax).
4. Type your final answers in this Word document and submit online through Canvas.
5. Don’t turn in handwritten answers with scribbling, cross-outs, erasures, etc. If an answer is unreadable, it will earn zero points. **Neatly and cleanly handwritten submissions are also acceptable**.

1. (3 points) Bill has an algorithm, find2D, to find an element x in an n×n array A. The algorithm find2D iterates over the rows of A and calls the algorithm arrayFind (see below) on each one, until x is found or it has searched all rows of A. What is the worst-case running time of find2D in terms of n? Is this a linear-time algorithm? Why or why not?

Text, letter

Description automatically generated

* The worst-case running time for this algorithm would be if the element x was the last element in the nxn array, which would be O(n). The algorithm with a running time of O(n) is compared to graphs of functions. O(n^2) would be a parabola like x^2, but a parabola is not linear. O(n) is a linear-time algorithm like y=x is a linear function on a graph.

2. (4 points) *Computational problem solving: Developing strategies*: An array A contains n−1 unique integers in the range [0,n−1]; that is, there is one number from this range that is not in A. Describe a strategy (not an algorithm) for finding that number. You are allowed to use only a constant number of additional spaces beside the array A itself.

* One strategy you can use to find the missing element in an array is with the Sum of Consecutive Integers. The formula n(n+1)/2 will calculate the sum of all the elements in the array and subtract it from the sum of the first n natural numbers, which will give the value of the missing element.

3. (3 points) *Computational problem solving: Developing strategies:* Given a string, S, of *n* digits in the range from 0 to 0, describe an efficient strategy for converting S into the integer it represents.

* I believe that finding the values of the digits in a string is best done by a recursive function. This function will take the first digit of the string and multiplies it with the appropriate power of 10. I would then add the recursive result for the substring starting at the second index.

4. (3 points) *Computational problem solving: Estimating problem solving time:* Suppose there are three algorithms to solve a problem- a O(n) algorithm (A1), a O(nlogn) algorithm (A2) and a O(n2) algorithm (A3) where log is to the base 2. Using the techniques and assumptions in slide set L2-Buffet(SelectionProblem).ppt, determine how long in seconds it will take for each algorithm to solve a problem of size 200 million. You must show your work to get credit, i.e., a correct answer without showing how it was arrived at will receive zero credit.

* From the example from the L2 Powerpoint:

I guess I have a really fast machine that runs at 4 GHz, 4 \* 10^9 clock cycles per second. This machine costs about 200 clock cycles to execute one step. Combined, the computer will execute 2 \* 10^7 steps per second.

* At 200 million:

For an O(n) algorithm, it will take (2 \* 10^7)(200 \* 10^6) = 10 seconds to execute.

For an O(nlog(n)) algorithm, it will take (2 \* 10^8 x log2( 2 \* 10^8)) / (2\*10^7) = 275.75 seconds.

For an O(n^2) algorithm, it will take (2 \* 10^8)^2 / (2 \* 10^7) = 2 \* 10^9 seconds, which is approximately 64 years.

5. (6 points) *Computational problem solving: Problem specification*

Suppose you are asked to develop a mobile application to provide **turn by turn** directions on a smartphone to an AU parking lot in which there are at least five empty parking spots nearest to a campus building that a user selects. Assume that you can use the Google Map API for two functions (only) ─ display campus map on the phone so user can select a campus building, and produce turn-by-turn directions from a source location to a destination location ─ where any location in the map is specified as a pair (latitude, longitude). Also assume that there is an application called AUparking that you can query to determine the # of vacant spots in any parking lot specified as a pair (latitude, longitude). Specify the problem to a level of detail that would allow you to develop solution strategies and corresponding algorithms: State the problem specification in terms of (1) inputs, (2) data representation and (3) desired outputs; no need to discuss solution strategies.

* (1) inputs:

For the inputs on this Location-Finding Application, you will need to incorporate the abilities of GPS satellites.

The app will need to have sensors and a pinpoint compass that uses True North, so a magnetometer.

The location of the user must be constantly accessible so no misinformation can be shown.

* (2) data representation

The cardinal directions will be shown at times, along with the longitude and latitude.

The magnetometer needs SI units, so Tesla units(T) is the best choice.

The Tesla units will point north and be utilized as initialization of the rotation of the device.

* (3) desired outputs

Along with the data from the sensors and Google.

The device will need to triangulate with the satellites, so the location is precise.

The output must include the instructions from the system to the user of the app based on the movement.

The instructions will show the directions to the empty spots show on the UI. With text and a marked path that reevaluates as the user gets closer.

6. (5 points) *Computational problem solving: Developing strategies*

Explain a correct and efficient **strategy** to check what the maximum difference is between any pair of numbers in an array containing n numbers. Your description should be such that the strategy is clear, but at the same time, the description should be at the level of a strategy, not an algorithm. Then state the total number of number pairs any algorithm using the strategy “compute the difference between every number pair in the array and select that pair with the largest difference” will have to consider as a function of n.

* To find the maximum difference between any pair of numbers, I will not need to check for the total numbers of pairs. It must sort the array using a built-in function or using an efficient sorting technique instead, like a quicksort. Sorting the number in the first index[0] and the last index[length-1] is the pair needed. The minimum of the array is the first number and the maximum of the array is the last number. The time-complexity for this strategy is O(n log(n)) where n is the length of the array.

7. (9 points) *Computational problem solving: Understanding an algorithm and its strategy*

**Algorithm** Mystery(A[1..n])

**Input**: An n-element array. Indexed from 1 to n

Text

Description automatically generated

1. Explain what the following algorithm outputs and simulate its operation on a valid input instance (e.g., an array of n elements - you can choose n to be 10)

* This algorithm finds the consecutive sequences to ultimately produce the maximum value.

1. What is the approximate time complexity (running time) of the above algorithm (you can use Big-Oh notation)

* This algorithm contains three **for** loops so I believe that he approximate time complexity is O(n^3).

1. How does the following algorithm improve the time complexity of the algorithm (what is its strategy)? What is its time complexity?

* I believe this algorithm is not improved from the first, it is slower. The algorithm above is computing all the prefix sums(O(n)) and then computing all the subarray sums(O(n^2)).

Text

Description automatically generated

8. (9 points) *Computational problem solving: Calculating approximate complexity:*

Using the approach described in class (L5-Complexity.pptx), calculate the approximate complexity of Mystery algorithm above by filling in the table below.

|  |  |
| --- | --- |
| Step | Big-Oh complexity |
| 1 | O(1) |
| 2 | O(n) |
| 3 | O(n) |
| 4 | O(n^2) |
| 5 | O(n^3) |
| 6 | O(n^3) |
| 7 | O(n^3-1) |
| 8 | O(n^3-1) |
| 9 | O(1) |
| Complexity of the algorithm | O(n^3) |

9. (9 points) Calculate the detailed complexity T(n) of Mystery. Fill in the table below, then determine the expression for T(n) and simplify it to produce a polynomial in n.

|  |  |  |
| --- | --- | --- |
| Step | Cost of each execution | Total # of times executed |
| 1 | 1 | 1 |
| 2 | 1 | n+1 |
| 3 | 1 | (n^2+3n)/2 |
| 4 | 6 |  |
| 5 | 3 |  |
| 6 | 2 |  |
| 7 | 1 |  |
| 8 | 1 |  |
| 9 | 1 |  |

T(n) =

10. (3 points) *Computational problem solving: Proving correctness/incorrectness:*

Is the algorithm below correct or incorrect? Prove it! It is supposed to count the number of all identical integers that appear consecutively in a file of integers. E.g., if f contains 1 2 3 3 3 4 3 5 6 6 7 8 8 8 8 then the correct answer is 9

Count(f: input file)

count, i, j : integer //local variables

count=0

while end-of-file(f)=false

i=read-next-integer(f)

if end-of-file(f)=false then

j=read-next-integer(f)

if i=j then count=count+1

return count

- The algorithm above is incorrect.

Proof:

* **Initialization:** Before the loop begins, the count=0 because no integers were read from the file.
* **Maintenance:** Inside the loop, it reads two integers and compares them i = j, if so increase the count by 1.The algorithm should keep track of the previous integer and take the current integer to compare the two. At any time, the algorithm will not give the correct count.
* **Termination:** The loop ends once the end of the file is reached.

11. (10 points) *Computational problem solving: Proving correctness:* Complete the proof by contradiction this algorithm to compute the Fibonacci numbers is correct.

function fib(n)

1. if n=0 then return(1)

2. if n=1 then return(1)

3. last=1

4. current=1

5. for i=2 to n do

6. temp=last+current

7. last=current

8. current=temp

9. return(current)

1. Assume the algorithm is incorrect.
2. Fibonacci numbers are defined as F0=1, F1=1, Fi=Fi-1+Fi-2 for i>1.
3. So the assumption in (1) implies that there is at least one input parameter n=k, k≥0, for which the algorithm will produce an incorrect answer.
4. If k=0, then if statement in the step 1 will be executed and return 0th Fibonacci number that is ‘1’.

If k=1, if statement in the step 2 will be executed and return 0th Fibonacci number that is ‘1’.

So in both cases the algorithm returns the correct answer.

1. This implies that there has to be at least one integer k>1, so that when n=k the algorithm does not return the correct answer Fk=Fk-1+Fk-2.
2. When n=k and k>1, the first two steps will be skipped, and steps 3-9 will be executed.
3. If k=2, the for loop in steps 5-8 will be executed exactly once. By step 6, temp = last + current = 1 + 1 = F0 + F1. Then step 7 updates last to be equal to current = F1. Step 7 updates current to be equal to temp which is F0 + F1. So the value returned in step 9 is current = F0 + F1 = F2. This is the correct answer. So the k for which the algorithm fails must be greater than 2.
4. If k=3, the for loop in steps 5-8 will be executed exactly twice. In the first loop, by step 6, temp=last+current, then step 7 updates to be equal to current. Step 8 updates current to be equal to temp.
5. But if k= 4, for loop in step 5-8 will be executed exactly three times.
6. The above argument can be repeated to show that the algorithm returns correct answer.
7. That is, for all k > 1 the algorithm returns the correct k-th Fibonacci number.
8. So there is no k for which the algorithm will return a value not equal to Fk-1+Fk-2. This contradicts (3).
9. Therefore, the algorithm must be correct.

12. (a) (6 points) *Computational problem solving: Algorithm design:* Describe a recursive algorithm to reverse a string that uses the strategy of swapping the first and last characters and recursively reversing the rest of the string. Assume the string is passed to the algorithm as an array A of characters, A[p…q], where the array has starting index p and ending index q, and the length of the string is n=q–p+1. The algorithm should have only one base case, when it gets an empty string. Assume you have a swap(A[i],A[j]) function available that will swap the characters in cells i and j. Write the algorithm using pseudocode without any programming language specific syntax. Your algorithm should be correct as per the technical definition of correctness.

Recursive – reverse (A[p…q] {

If (p<=q) {

Swap (A [p], A[q])**; // method to swap**

**// Elements of position p.q is A**

**// Recursive call**

Recursive – reverse (A[p+1…q-1]

}

}

(b) (8 points) Draw your algorithm’s recursion tree on input string “i<33270!”- remember to show inputs and outputs of each recursive execution including the execution of any base cases.

Algorthm String

|  |  |
| --- | --- |
| 1. R-r (A [0 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

13. (10 points) *Computational problem solving: Proving correctness:*

Function g (n: nonnegative integer)

if n ≤ 1 then return(n)

else return(5\*g(n-1) – 6\*g(n-2))

Prove by induction that algorithm g is correct, if it is intended to compute the function 3n-2n for all n ≥ 0.

Base Case Proof:

Inductive Hypothesis:

Inductive Step:

14. (12 points) *Computational problem solving: Proving correctness:* The algorithm of Q.11 can also be proven correct using the Loop Invariant method. The proof will first show that it will correctly compute F0 & F1 by virtue of lines 1 and 2, and then show that it will correctly compute Fn, n>1, using the LI technique on the for loop. For this latter part of the correctness proof, complete the Loop Invariant below by filing in the blanks. Then complete the three parts of the rest of the proof.

Loop Invariant:

Before any execution of the for loop of line 5 in which the loop variable i=k, 2≤k≤n, the variable last will contain \_\_\_\_\_\_\_ and the variable current will contain \_\_\_\_\_\_\_\_\_.

Initialization:

Maintenance:

Termination: