**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?

**The worst-case running time of find2D is O(n2) because you are iterating through every element on each array of length “n” n times. This is not a linear-time algorithm because the worst-case running time is not O(n).**

Text, letter

Description automatically generated

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.

**To find the number that is not in A, I would compute the sum of all the numbers in A (put to variable sumA). Then I would compute the sum of all numbers from 0 to n (put to sumN). I would then subtract sumA - sumN, and the answer is the missing number.**

3. (3 points) *Computational problem solving: Developing strategies:* Given a string, S, of *n* digits in the range from 0 to 9, describe an efficient strategy for converting S into the integer it represents.

**To convert S into the integer it represents, I would first take S and parse it into characters. I would convert the first character to an integer and then multiply that number by 10. Then I would add the next character and multiply the number by 10. The process will continue until the last element, where the number will not be multiplied by 10.**

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.

**Current assumptions - machine running at 4 \* 109 clock cycles per second, 200 clock cycles to execute one computation step. Our algorithm will need to execute 2 \* 108 steps because our problem is of size 200 million. We can conclude that our computer can execute steps each second.**

1. **O(n)**

**n – (2 \* 10^8) steps**

**(2 \* 10^8)/ (2 \* 10^7) = 10 seconds**

1. **O(nlogn)**

**nlogn – (2-10^8) log2(2\*108)**

**((2-10^8) log2(2\*108))/ (2 \* 107) = 276 seconds**

1. **O(n2)**

**n^2 – (2 \* 10^8) ^2**

**((2\*10^8) ^2)/ (2 \* 107) = 2,000,000,000 seconds**

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.

**Our goal is to find a parking space with at least 5 open parking slots closest to our desired building.**

1. **Inputs - we would need to specify a location on a map as our input, which can be done through the Google Map API. We would also need to use AUparking to input all parking lot locations to see which lots fit our criteria and eliminate the lots not available.**
2. **Data representation - all possible locations on a map (including buildings and parking lots) will be represented as a coordinate (latitude, longitude). The directions given by the Google Map API from the source location to the destination location will be generated as a turn-by-turn path.**
3. **Desired output - we want the nearest vacant parking lot and turn-to-turn directions between the parking lot and source location. The vacant parking lot location will be given as a coordinate (latitude, longitude) and the turn-to-turn directions will be produced by the Google Map API.**

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.

**I would set two variables, max and min, to equal the first element in the array. I would then loop through each element in the array. If the number is greater than max, then it is the new max. If the number is less than min, then it is the min. If none of the above, go to the next element. After the loop, I would subtract the max from the min, which will give me the greatest difference between two numbers in the loop.**

**For the strategy “compute the difference between every number pair in the array and**

**select that pair with the largest difference,” the algorithm will need to consider n2 number**

**pairs because one loop will be needed for the first element, and another loop will be**

**needed to loop through the rest of the elements until the first loop reaches the end 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)

**The algorithm outputs the greatest partial sum of all elements in the array as the index number moves farther towards the right. For example, for the array {9, 1, -2, 5, 10}, the first partial sum would be as follows:**

**1. 9 + 1 - 2 + 5 + 10 = 23**

**2. 1 - 2 + 5 + 10 = 14**

**3. - 2 + 5 + 10 = 13**

**4. 5 + 10 = 15**

**5. 10 = 10**

**The algorithm outputs 23 since it is the greatest partial sum.**

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

**The algorithm’s time complexity is O(n3) since there are 2 for loops nested inside one**

**for loop.**

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

**The following algorithm first stores all partial sums in an array using a for loop. Then**

**the next for loop then determines which partial sum is maximum. The strategy is different**

**since the partial sums iteration is taken care of outside of the nested for loops. The time**

**complexity for the algorithm would be O(n2).**

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 (original algorithm – not the improved version) above by filling in the table below. Note that the algorithm has 9 lines of code. Each line is a distinct step.

|  |  |
| --- | --- |
| Step | Big-Oh complexity |
| 1 | **O(1)** |
| 2 | **O(n)** |
| 3 | **O(n)** |
| 4 | **O(1)** |
| 5 | **O(n)** |
| 6 | **O(1)** |
| 7 | **O(1)** |
| 8 | **O(1)** |
| 9 | **O(1)** |
| Complexity of the algorithm | **O(n3)** |

9. (9 points) Calculate the detailed complexity T(n) of Mystery (original algorithm – not the improved version). 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 |  |
| 4 | 1 |  |
| 5 | 1 |  |
| 6 | 7 |  |
| 7 | 3 |  |
| 8 | 2 |  |
| 9 | 1 | **1** |

Text

Description automatically generated with medium confidence

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

**We can prove the algorithm is incorrect with proof by counterexample. Let’s use the file containing 1 2 3 3 3 4 3 5 6 6 7 8 8 8 8. We know the count starts as 0. The first integer read is 1 and the second integer read is 2. 1 is not equal to 2 so count remains 0. The next integer read is 3 and the following integer read is 3. 3 is equal to 3 so count is 1. The next integer read is 3 and the following integer read is 4. 3 is not equal to 4 so count remains 1. The next integer read is 3 and the following integer read is 5. 3 is not equal to 5 so count remains 1. The next integer read is 6 and the following integer read is 6. 6 is equal to 6 so count is 2. The next integer read is 7 and the following integer read is 8. 7 is not equal to 8 so count remains 2. The next integer read is 8 and the following integer read is 8. 8 is equal to 8 so count is 3. The next integer read is 3 and then the algorithm runs into the end of the file. At the end, the count is equal to 3, which is not 9 like it was supposed to be. Therefore, the algorithm is incorrect.**

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 the value of k is either 0 or 1, then steps 1 and 2 are executed respectively and return the value 1 for both 0 and 1, which is true. So in both cases the algorithm returns the correct answer. So in both cases the algorithm returns the correct answer.**
5. 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.
6. **When n=k and k>1, steps 1 and 2 fail, and steps 3-9 will be executed.**
7. 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.
8. **If k=3, the for loop in steps 5-8 will be executed twice. We know that last is 1 and current is 2 from (7). By step 6, temp = last + current = 1 + 2 = F1 + F2= 3. Then step 7 updates last to be equal to current = F2 = 2. Step 7 updates current to be equal to temp, which is F1 + F2 = 3. This is the correct answer. So the k for which the algorithm fails must be greater than 2.**
9. **But if k= 4, the for loop in steps 5-8 will be executed 3 times. We know that last is 2 and current is 3 from (8). By step 6, temp = last + current = 2 + 3 = F3 + F4 = 5. Then step 7 updates last to be equal to current = F4 = 3. Step 7 updates current to be equal to temp, which is F3 + F4 = 5. This is the correct answer. So the k for which the algorithm fails must be greater than 2.**
10. **The above argument can be repeated to show that the for loop always produces the correct answer.**
11. That is, for all k > 1 the algorithm returns the correct k-th Fibonacci number.
12. So there is no k for which the algorithm will return a value not equal to Fk-1+Fk-2. This contradicts (3).
13. 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.

**To design a recursive algorithm that reverses a string, we must first define the base case. The algorithm will return the string once it receives an empty string as a parameter. For all string lengths greater than 0, we can swap p and q, and increment p while decrementing q recursively. Pseudocode:**

1. **reverse (A[p…q])**
2. **if (p - q + 1) <= 1                      // Base case**
3. **return A[p…q]**
4. **if (p <= q)                                 // Makes sure p and q don’t overlap**
5. **swap(A[p],A[j])              // Swaps first and last characters**
6. **reverse (A[p+1…q-1])    // Moves inwards the string to continue reversing**
7. **return A[p…q]**

(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.

A piece of paper with writing on it

Description automatically generated

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:

**When n ≤ 1, then n can be either 0 or 1. When n = 0, then 30-20 = 1 – 1 = 0, which is equal to n. When n = 1, then 31-21 = 1, which is also equal to n. Since these two conditions are satisfied, the base case is true.**

Inductive Hypothesis:

**Assume that for any integer k ≥ 2, the function g(k) = 3k-2k and g(k-1) = 3k-1-2k-1 will be computed.**

Inductive Step:

**g(n) = 5\*g(n-1) – 6\*g(n-2)**

**g(k+1) = 5\*g(k)-6\*g(k-1)**

**g(k+1) = 5\*(3k-2k) – 6\*(3k-1-2k-1)**

**g(k+1) = 5(3k) – 5(2k) – 2(3k) – 3(2k)**

**g(k+1) = 3(3k) – 2(2k)**

**g(k+1) = 3k+1 – 2k+1**

**Therefore, for any integer k ≥ 2, the algorithm g will compute the function 3k- 2k and is correct.**

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 (k-2) number in the sequence and the variable current will contain (k-1) number in the sequence.**

Initialization:

**We know that the loop invariant is true because if 2≤k≤n, at k = 2, last is 1, and current is 1, as shown from the first two if statements.**

Maintenance:

**Assume that the loop invariant holds at the start of iteration k. Then it must be that last contains the (k-2) th number and current contains the (k-1) th number. In iteration k, variable temp is equal to**

**num(k-2)+num(k-1), which is num(k) in the sequence. Last is equal to current, which is num(k-1), and current is equal to temp, num(k). Thus, at iteration k+1, temp is equal to num(k-1)+num(k), which is num(k+1) in the sequence. Last is equal to current, which is num(k), and current is equal to temp, num(k+1). So the loop invariant holds.**

Termination:

**The loop terminates at the nth iteration, so value of current at this point is num(n). The loop invariant says before any execution of the loop where the loop variable is 2≤k≤n, the variable last will contain (n-2) number in the sequence and the variable current will contain (n-1) number in the sequence. Since num(n) is returned, the algorithm is correct.**