**COMP 3270 Homework 1**

100 points. Due by **11:59pm (midnight) on Thursday, September 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**.
6. (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?

O(n)^2

No, because the worst case would end up being O(n)^2 because inside the while loop is O(n) and the while loop is O(n) so O(n)\*O(n) therefore non-linear.

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. I would Search through the range [0,n-1] comparing each number with array A to see if array A contains that number and if so dispose of those values and go to the next number until you find the one in the range that isn’t in A

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. Count the number of characters in the string until the next character is a white space then this would be the integer it represents

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.

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

O(nlogn) = [2\*10^8(log(2\*10^8))]/(2\*10^7) = 275.754 seconds

O(n^2) = [(2\*10^8)^2]/(2\*10^7) = 2\*10^9 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.

1. Inputs: User Location (latitude, longitude), Destination (latitude, longitude), AUparking Parking lot (latitude, longitude)
2. Data Representation: Locations using nodes and shown with latitude and longitude values
3. Outputs: closest location with at least 5 empty parking spots, distance, campus map, AUparking # of open spots, turn by turn directions

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.

Assign the first element of the array as the min and max value, then iterate through the array. When an element greater than the last max is found declare that the new max, and when one that’s less than the declared min is found declare it the min value. Repeat for n array elements then the max and min found at the end will produce the largest difference

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 be n(n-1)/2

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 sum of a subarray with a max sum in the given array

For an array with 4 elements A = {2, 4, 6, 8}

S = 2 + 4 + 6 + 8 = 20

4 + 6 + 8 = 18

6 + 8 = 14

8 = 8

The program returns 20

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

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?

Text

Description automatically generated

The time complexity would be O(n2), and this algorithm improves the time complexity because there are only 2 nested loops and the other is independent.

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(n2) |
| 4 | O(1) |
| 5 | O(n3) |
| 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. 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 | 1 |
| 3 | 1 | n+1 |
| 4 | 1 | n |
| 5 | 1 |  |
| 6 | 6 |  |
| 7 | 3 |  |
| 8 | 2 |  |
| 9 | 2 | 2 |

T(n) = T(n) = 1 + 1 + n + 1 + n + + 6 + 3 + 2 + 2

= 5 + 2n + 12 +

= 5 + 3n + 12 – 12n

= 5 – 9n + 12(n(n+1)/2)

= 5 – 9n + 6n + 6n2

T(n) = 6n2 - 3n + 5

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

If you set f = 1 2 3 3 3 4 3 5 6 6 7 8 8 8 8 then this algorithm would return 6 because it reads the next number to see if it’s the same as the one it’s on and increases count, but it doesn’t count the one is already on. Therefore, it would be incorrect because the correct answer is 9.

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. \_\_\_For k = 0, F0 = 1 and the value returned by the algorithm is also 1. For k = 1, F1 = 1 and the value returned by the algorithm is also 1.\_\_\_ 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 \_\_\_\_\_\_\_\_\_\_The condition checks in 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 done one time. By step 6, temp = last = current 1 + 2 = F1 + F2 Then step 7 updates last to be equal to current = F2. Step 8 updates current to be equal to temp which is F1 + F2. So the value returned in step 9 is current = F1 + F2 = F3. This is the correct answer. So the k for which the algorithm fails must be greater than 3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
9. But if k= 4, \_\_\_\_ the for loop in steps 5-8 will be done one time. By step 6, temp = last + current = 2 + 3 = F2 + F3. Then step 7 updates last to be equal to current = F3. Step 8 updates current to be equal to temp which is F2 + F3. So the value returned in step 9 is current F2 + F3 = F4. This is the correct answer. So the k for which the algorithm fails must be greater than 4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
10. The above argument can be repeated to show that \_\_for k = n, the for loop in steps 5-8 will be done one time. By step 6, temp = last + current = Fn-2 + Fn-1. Then step 8 updates last to be equal to current = Fn-1. Step 7 updates current to be equal to temp which is Fn-2 + Fn. So the value returned in step 9 is current = Fn-2 + Fn-1 = Fn. This is 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.

RecursiveString(A[p..q])

int n < 2

return 0;

if p <= q

swap(A[p], A[q]);

RecursiveString(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.

Recursive-String-Reversal(A[1…8])

A = !<33270i

Recursive-String-Reversal(A[2…7])

A = !<03327<i

Recursive-String-Reversal(A[3…6])

A = !07323<i

Recursive-String-Reversal(A[4…5])

A = !07233<i

Recursive-String-Reversal(A[5…4])

Output: A = !07233<i

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:

If n = 0, 0 is returned

If n = 1, 1 is returned, 31 – 21 = 1

1 = 1

Inductive Hypothesis:

For n = k, g(k) = 3k-2k = 1.

Inductive Step:

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

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

g(k+1) = 3k(5-2) – 2k(5-3)

g(k+1) = 3k \* 3 – 2k \* 2

g(k+1) = 3k+1 – 2k+1

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 \_\_\_ F0\_\_\_\_ and the variable current will contain \_\_\_ F1\_\_\_\_\_\_.

Initialization:

When initialized the value of F0 is stored in last and F1 is stored in current

Maintenance:

The values stored in last and current are then added together into temp. Current is then moved into last then temp is moved into current. Current is now initialized to F2 and last is F1. This pattern of upscaling the Fibonacci sequence will continue until the loop reaches termination.

Termination:

It terminates when it reaches any value greater than n. n is the number in the Fibonacci sequence we are looking for. The correct n value is returned, and the loop knows when to terminate at the correct time.