**COMP 3270 Homework 1**

**Solutions to Selected Problems**

*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-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.

1. Complexity of A1 is O(n)

Input size = 200 million = 2 \* 108

Number of steps executed by the computer in 1 second = 2 \* 107

Number of steps that needs to be executed for an algorithm of complexity O(n) = 2 \* 108

Execution time = (2 \* 108) / (2 \* 107) = 10 sec.

1. Complexity of A2 is O(nlogn)

Input size = 200 million = 2 \* 108

Number of steps executed by the computer in 1 second = 2 \* 107

Number of steps that needs to be executed for an algorithm of complexity O(n) = 2\*108(log(2\*108)

Execution time = (2 \* 108)[log(2 \* 108)] / (2 \* 107) = 275.75 sec ≈ 276 sec

1. Complexity of A3 is O(n2)

Input size = 200 million = 2 \* 108

Number of steps executed by the computer in 1 second = 2 \* 107

Number of steps that needs to be executed for an algorithm of complexity O(n) = (2\*108)2

Execution time = (2 \* 108)2 / (2 \* 107) = 2 \* 109 sec = 200 million sec

*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:

Current location

Destination location

1. Data Representation

All locations are represented as nodes and specified with latitude and longitude values.

Inputs:

* Current location node is specified as: (C[latitude, longitude])
* Destination location node is specified as: (D[latitude, longitude])

Outputs:

* Location of the parking lot node, nearest to the destination and having at least 5 empty parking spots is specified as: (L(latitude, longitude))
* Turn by turn direction to the parking lot from current location.

This can be represented as an nXn adjacency matrix, where n are the nodes from the current location to the destination location. The values of the matrix can be specified using ‘↓’, ‘↑’, ‘→’, ‘←’(representing the ‘south’, ’north’, ‘east’, ‘west’ direction of navigation). The value of element aij in the matrix specifies the direction of navigation from the current node ‘i’ to the node ‘j’.

This can be represented as an sXs adjacency matrix, where s are the segments joining the nodes from the current location to the destination location. The values of the matrix can be specified using ‘↓’, ‘↑’, ‘→’, ‘←’(representing the ‘south’, ’north’, ‘east’, ‘west’ direction of navigation). The value of element aij in the matrix specifies the direction of navigation from the current segment ‘i’ to the segment ‘j’.

1. Desired Outputs

Location of the parking lot nearest to the destination and having at least 5 empty parking spots.

Turn by turn direction to the parking lot from current location.

*Computational problem solving: Developing strategies*

Given a sorted sequence of distinct integers a1, a2, …, an, determine whether there exists an index i such that ai = i. [The trivial strategy of iterating over the elements one by one and checking if there exist an element ai = I is not acceptable -- you are expected to devise a different strategy than this obvious one]. Then state the total number of steps an algorithm that implements the strategy will have to consider as a function of n.

Although the strategy underlying pure binary search is not applicable because the value of searched items is not given. However, the property we seek is adaptable to the binary search strategy. Consider the value of an/2 (assume that n is even. If this value is exactly n/2, then we are done. Otherwise, if it sis less than n/2, then, since all numbers are distinct, the value of an/2-1 is less than n/2-1, and so on.No number in the first half of the sequence can satisfy that property, and we can continue searching the second half. The same argument holds if the answer is “greater than”.

*Computational problem solving: Understanding an algorithm and its strategy by simulating it on a specific input:*

Understand the following algorithm. Simulate it mentally on the following four inputs, and state the outputs produced (value returned) in each case: (a) A: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; (b) A: [─1, ─2, ─3, ─4, ─5, ─6, ─7, ─8, ─9, ─10], ; (c) A: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], (d) A: [─1, 2, ─3, 4, ─5, 6, 7, ─8, 9, ─10].

Algorithm-1 (A:array[1..n] of integer)

sum, max: integer

1 sum = 0

2 max = 0

3 for i = 1 to n

4 sum = 0

5 for j = i to n

6 sum = sum + A[j]

7 if sum > max then

8 max = sum

9 return max

Output when input is array (a) above: 55

Output when input is array (b) above: 0

Output when input is array (c) above: 0

Output when input is array (d) above: 14

What does the algorithm return when the input array contains all negative integers?

It returns zero.

What does the algorithm return when the input array contains all non-negative integers?

It returns the sum of all the integers.

What does the algorithm return when the input array contains negative, zero and positive integers?

It returns the largest of the positive sum of all the integers and the largest positive number.

*Computational problem solving: Calculating approximate complexity:*

Using the approach described in class (L5-Complexity.pptx), calculate the approximate complexity of Algorithm-1 above by filling in the table below.

|  |  |
| --- | --- |
| Step | Big-Oh complexity |
| 1 | O(1) |
| 2 | O(1) |
| 3 | O(n) |
| 4 | O(n) |
| 5 | O(n2) |
| 6 | O(n2) |
| 7 | O(n2) |
| 8 | O(n2) |
| 9 | O(1) |
| Complexity of the algorithm | O(n2) |

Calculate the detailed complexity T(n) of Algorithm-1. 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 | ∑i=1n (n – i + 2) |
| 6 | 6 | ∑i=1n (n – i+1) |
| 7 | 3 | ∑i=1n (n – i+1) |
| 8 | 2 | ∑i=1n (n – i+1) |
| 9 | 1 | 1 |

T(n) = 1 + 1 + n + 1 + n + ∑i=1n (n – i + 2) + 6∑i=1n (n – i+1) + 3∑i=1n (n – i+1) + 2∑i=1n (n – i+1) + 1

You need to carry out the math by evaluating the summation expressions to derive a closed form polynomial expression as a function of n.

*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 is incorrect.

Proof by counterexample:

1. Let f = 1 2 3 3 3 4 3 5 6 6 7 8 8 8 8.
2. The output of the algorithm is 3.
3. The correct output is 9.
4. The algorithm is incorrect.

*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:

For n =1, g(1) = 31-21 = 1.

Inductive Hypothesis:

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

Inductive Step:

Now we need to show that, for n = k+1,

g(k+1) = 3k+1-2k+1

We know that g(k) = 3k-2k and g(k-1) = 3k-1-2k-1, from inductive hypothesis.

We also know that the value returned by the algorithm for n = k+1, is

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

Upon substitution we get,

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

Therefore, proved.

*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-String-Reversal(A[p…q])

1. If p >= q
2. return A
3. else swap(A[p], A[q])
4. Recursive-String-Reversal(A[p+1], A[q-1])

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