**Regis University CC&IS**

**CS324 Algorithms & Analysis**

**Alternate Homework 1 (v1)**

**Assignment**Place your answers directly under each question below -- do not delete the questions.

***Part 1: Algorithm Complexity Analysis***

1. Give the big-Oh characterization, in terms of **n**, of the running time of the pseudocode loops shown below: (*4 pts each*)
2. x 🡨 n + 10;

y 🡨 n + 20;

while x > 0 do

y 🡨 y + x;

x 🡨 x / 2

It is known that when an algorithm divides or multiplies by 2: the resultant analysis can be generalized to the formula

x 🡨 n + 10;

y 🡨 n + 20;

AND

y 🡨 y + x;  
  
Are all constnt time assignments

**Therefore the Big-Oh of the above is:**

1. w 🡨 0

for i 🡨 1 to n3 do

for j 🡨 2 to n do

w 🡨 w \* i

Run times for nested loops are the result of multiplying the inner loops number of runs, vs the outer loop.  
  
In this example the assignment and multiplication steps are constant.  
**Therefore the runtime is:**

1. x 🡨 1

for i 🡨 2n downto 1 do

x 🡨 x \* i

Assignment and multiplication steps are constant.

The loop executes 2n since I is assigned 2n and iterates through every number to 1.

**Therefore the time complexity is**

1. a 🡨 100

for i 🡨 1 to n do

for j 🡨 1 to n/2 do

a 🡨 a + i  
The outer loop executes n times.

The inner loop executes n/2 times

Assignment is constant.

We drop the constant from the runtime value of

**Therefore the time complexity is**

)

1. Complete problem C1.6 on page 45 of your textbook (*6 pts*)

Show ALL math conversion steps.  
Starting with the base assumptions:

and

We can then see that

Continuing we get

Looking at the following we can see the difference between any successive term is always a power of 2

We can conclude from this fact and reminder from earlier that all terms in the sequence T(n) are powers of 2 – 1. That is:

1. Given the following functions, determine the big-O notation for each (3 *pts each*)

Hint: Use the rules given in the textbook.

1. f(n) = 33n2 + 5n3 + 1000n2

Divide by n^2

Divide by 2, then by 5 to obtain

For all

Therefore:

1. f(n) = 6n4 + 5n4 + 20n4
   1. O(n^4)

Because this is not strictly true we cannot say that there is an n for which:

The term 20n^4 still grows the fastest and thus we say:

1. f(n) = 9n3 + 2n + 8n2

Assuming we can prove O(n^3) we take the following:

Dividing by n^2 we get the following:

Subtracting 8 from both sides we obtain:

1. f(n) = 50n (log n) + 22n

C = 50 because 50 is the greatest term.

Subtracting 22 from both sides get us the following which can be then divided by 28

Dividing by 28 yields:

We know this is always true for all:

1. Consider the following algorithm (shown with line numbers) that sorts the elements in an array of positive integers into ascending order.

**1: void selectionSort(int nums[], int N) {**

**2: int temp,**

**3: currentIdx = 0,**

**4: lastIdx = N - 1;**

**5: while (currentIdx < lastIdx) {**

**6: minIdx = currentIdx;**

**7: for (int i = currentIdx + 1; i <= lastIdx; i++) {**

**8: if (nums[i] < nums[minIdx])**

**9: minIdx = i;**

**10: }**

**11: if (minIdx != currentIdx) {**

**12 temp = nums[minIdx];**

**13: nums[minIdx] = nums[currentIdx];**

**14: nums[currentIdx] = temp;**

**15: }**

**16: currentIdx++;**

**17: }**

**18: }**

1. Best case
   1. Describe the best case **data** for this algorithm. *(3 pts)*
      * The best case scenario is when the items are sorted.
        1. The outer loop executes n times.
        2. The inner loop runs the sum of n…0 times
           1. The result is a runtime of (n)\*sum((n))
        3. Ultimately this algorithm runs in:
           1. O((n\*(n-1)/2)
   2. Then, for this best case, state how often is line 9 executed. *(2 pts)*

Line nine is executed zero times because num[i], the item after nums[minIdx], will always be sorted and thus always be less than the previous item.

1. Worst case
   1. Describe the worst case **data** for this algorithm. *(3 pts)*
      * The worst case is when the items are in reverse order.
        1. The outer loop runs n times
        2. The inner the sum of n-1 times.
        3. The runtime is:
   2. For this worst case, state how often is line 9 executed. *(2 pts)*

Worst case scenario line 9 is executed

times

1. Suppose that for the worst case, given input size **n**:

Algorithm 1 does **f(n) = n3 + 1** steps

Algorithm 2 does **f(n) = n2 + 200** steps

For what input sizes is algorithm 1 faster than algorithm 2, in the worst case? (*5 pts)*

This occurs when

The inequality only holds if

**Therefore algorithm 1 is faster than algorithm 2 for input sizes less than 10.**

1. Given the following ordered array: (*5 pts each*)

|  |  |
| --- | --- |
| [0] | 3 |
| [1] | 6 |
| [2] | 8 |
| [3] | 12 |
| [4] | 19 |
| [5] | 23 |
| [6] | 32 |
| [7] | 44 |
| [8] | 48 |
| [9] | 51 |
| d[10] | 62 |
| [11] | 70 |
| [12] | 73 |
| [13] | 74 |
| [14] | 78 |
| [15] | 80 |
| [16] | 85 |
| [17] | 88 |

1. Using the binary search algorithm to search for the target value **32**

For each pass, list the high and low index, and the calculated middle index.

Finally, state the result of the search.

Using the binary search algorithm to fin 32:

PASS 1

HIGH: 17 LOW: 0 MID: 8

PASS 2

HIGH: 7 LOW: 0 MID: 3

PASS 3

HIGH: 7 LOW: 4 MID: 5

PASS z

HIGH: 7 LOW: 6 MID: 6

RESULT: 6

1. Using the binary search algorithm to search for the target value **79**

For each pass, list the high and low index, and the calculated middle index.

Finally, state the result of the search.

Using the binary search algorithm to fin 79:

PASS 1

HIGH: 17 LOW: 0 MID: 8 Target: 79

PASS 2

HIGH: 17 LOW: 9 MID: 13 Target: 79

PASS 3

HIGH: 17 LOW: 14 MID: 15 Target: 79

PASS 4

HIGH: 14 LOW: 14 MID: 14 Target: 79

PASS 5

s

RESULT: NOT FOUND -1

1. Assume you have a 15-element hash table which uses the hash function   
        **H(key) = key mod 15**  
   and quadratic probing to resolve collisions.   
   If the following keys were inserted into the hash table, in the order shown: **29, 31, 22, 44, 45**  
   where will each be placed (i.e. at what index)? *(8 pts)*

|  |  |
| --- | --- |
|  |  |
| *Number* | *Index* |
| *29* | *14* |
| *31* | *1* |
| *22* | *7* |
| *44* | *0* |
| *45* | *4* |

1. Begin with an empty binary search tree. (5 *pts*)

Insert the following keys, in the given order:

58, 72, 26, 39, 31, 79, 88, 63, 42, 15, 81  
Draw the resulting tree.

1. Starting with the binary search tree shown below. (*5 pts each*)

# 

1. List the order the nodes will be visited, using pre-order traversal.

The nodes will be visited in the following order, array indices serve as priority/visit order.

[55, 34, 20, 15, 24, 30, 48, 65, 60, 56, 64, 73, 70, 68, 72]

1. List the order the nodes will be visited, using post-order traversal.

The nodes will be visited in the following order, array indices serve as priority/visit order.

[15, 30, 24, 20, 48, 34, 56, 64, 60, 68, 72, 70, 73, 65, 55]

1. Deletion algorithms either use the **left** subtree or **right** subtree to find the node to replace the deleted node (must use ***same choice for all deletions***).   
   ***State which subtree you will use***.

Then show the resulting tree, after delete nodes 24, 73, 34, 65, 55, in that order.

Deleting 24 results in

A diagram of a diagram

Description automatically generated

Deleting 73 results in

A diagram of a number

Description automatically generated

After deleting 34

A diagram of a number

Description automatically generated

After deleting 65

After deleting 55

1. Complete problem **A1.9** on page 49 of your textbook (*8 pts*)

Present the algorithm in pseudocode.

Do not use any built-in code library methods in your solution.

1. Compare and contrast some of the previously studied data structures:

Linked lists

Binary trees

Discuss advantages and disadvantages of each, in terms of the time it takes to **add** and **delete** elements within the data structure, and the **memory required** for the data structure. (*5 pts*)

**Submission**

This homework assignment is due by midnight of the date listed on the **Course Assignments by Week** page of the Content.

* Before you submit your homework, append your last name to the front of the Word doc filename. For examples: **Smith-CS324-AltHwk1v1.docx**
* Submit your **.docx** file to the **Hwk Assn 1** Submission Folder (located under **Assignments** tab in online course

***WARNING:****Homework submitted more than* ***3 days*** *past the due date will* ***not*** *be accepted,  
and will receive a grade of 0.*