Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to implement a pseudocode algorithm to collect data
* to check/validate a theoretical result.

What you need to do:

Complete the tasks described below.

Objective of this assignment:

* To verify empirically **Theorem 12.4** (Chapter 12, Textbook) that states:

“*The expected height h(n) of a randomly built binary search tree on n distinct keys is O(lg n).*”

What you need to do:

**Objective**:

The objective of this programming assignment is to verify empirically Theorem 12.4 that states that “*the expected height of a randomly built binary search tree on n distinct keys is O(lg n)*.”

In order to conduct this experiment, you must implement in your preferred language (available on Tux Machines) the Tree-Insert(T,z) operation. Below, I explain in pseudocode the program you must write to collect data and verify Theorem 12.4. You do not need to enforce that the keys are distinct.

**Program to implement**

collectData()

for n = 2,000 to 30,000 by 2,000 // 2,000, 4,000, 6,000, …. 30,000

sum\_heightn = 0

for j = 1 to m do //Take m = 5 measurements mj

for i = 1 to n

pick randomly a number p in the range [0-15,000]

create a node z

set z.key = p

Tree-Insert(T,z)

Measure the height hj of the tree

Discard Tree (free memory)

sum\_heightn += hj

collect Height(n)= sum\_heightn/m // Average height for n

Write in a file F the value n and Height(n)in a csv format

1. (3 points) Implement the *Tree-Insert(T,z)* operation on a binary search tree.

Insert here the implementation of Tree-Insert method/procedure. You can use your preferred language as long as it is already installed on Tux machines.

1. public class BinarySearchTree {
2. class Node {
4. int key;
5. Node left, right, parent;
6. public Node(int item) {
7. key = item;
8. left = right = parent = null;
9. }
10. }
11. Node root;
12. // Constructor for BST.
13. public BinarySearchTree() {
14. root = null;
15. }
16. // Static method to insert a new node in the created Binary Search Tree.
17. public static void treeInsert(BinarySearchTree T, Node z) {
19. Node y = null;
20. Node x = T.root;
21. while (x != null) {
22. y = x;
23. if (z.key < x.key) {
24. x = x.left;
25. } else {
26. x = x.right;
27. }
28. }
30. z.parent = y;
31. if (y == null) {
32. T.root = z; // Tree was empty.
33. } else if (z.key < y.key) {
34. y.left = z;
35. } else {
36. y.right = z;
37. }
38. }

}

(15 points) **Repeatedly** insert randomly picked numbers in a binary search tree and **collect** the height of the obtained tree. is the height of the binary search tree containing n elements. For simplicity, no need to enforce that the keys are distinct. In other words, you do not need to check whether a number is already in the tree.

a) Implement the program *collectData()* described below using pseudocode. You can use your preferred language as long as it is already installed on Tux machines. Turn in the source code of *collectData*() with this assignment file. Your program will be compiled, executed and tested on Tux machines.

b) Log onto a Tux machine, enter the command date, compile your *collectData*() program,

c) (5 points) Take a screenshot that clearly shows the date, your username, the tux machine name, the compilation directive and the program launch. The screenshot should be as readable as this template screenshot below. Credit for this assignment is conditioned on a readable screenshot that contains ALL required information (date, username, tux machine, compilation directives, program launch). In other words, if this screenshot is too small or missing, no credit will be awarded for this assignment. Insert the screenshot after the screenshot template below.

Screenshot for after compiling and running my BinarySearchTree.java file.

A screenshot of a computer

Description automatically generated

d) (10 points) Submit the csv (comma-separated-values) file F (See Pseudocode of *collectData*() )

I have attached the csv file that I named data\_for\_tree\_heights.csv along with this document for the assignment.

(60 points = (20 points per function/plot))

a) (20 points) Plot here on the graph the plot function versus ***n***.

Insert the plot here

b) (20 points) Plot here on the graph the plot function versus ***n***.

Insert the plot here

b) (20 points) Plot here on the graph the plot function versus ***n***.

Insert the plot here

(22 points)

a) (4 points) Suppose that , what should be the expected expression and shape of when n tends to infinity?

Answer here ... (hint: if , then when n tends to infinity with K a constant)

If we use this hint given h(n) is approximately K\*log(n), we can substitute this into the expression :

As n tends to infinity should approach the constant value

This would imply that plot 1 provided versus n should become a horizontal line at y = . This means that the ratio used in plot 1 stabilizes to a constant values as n becomes very large, which in turn also reflects the asymptotic behavior of h(n) being proportional to log(n).

b) (4 points) Suppose that , what should be the expected expression and shape of when n tends to infinity?

Answer here ...

Again given h(n) is approximately equal to K\*log(n) we can substitute this into the expression used for plot 2 . As “n” tends to infinity, it is easily noticeable from the graph that log(n) grows much slower than 2\*square root of n. As n tends to infinity, should approach 0, which is observable in plot 2. This reflects that the growth rate of 2\*square root of n exceeds that of the original log(n)

c) (4 points) Suppose that , what should be the expected expression and shape of  when n tends to infinity?

Answer here ...

Given h(n) is approximately equal to K\*log(n) and then substituting into the expression :

So as “n” tends to infinity, the term log(n) grows again much slower than just n. This means as n grows the plot 3 will tend towards 0. This is observed in the given plot 3 as well. The shape is a decreasing curve that approaches 0.

d) (10 points) Conclude about the validity of Theorem 12.4. Recall that this theorem states “*The expected height h(n) of a randomly built binary search tree on n distinct keys is O(lg n).*”

The plot of showed a horizontal line that ends up stabilizing as the constant value K increases. This is consistent with h(n) being proportional to log(n).

The plot of showed a decreasing curve that was approaching 0. This indicates that grows much faster than h(n) which is the expected output.

The plot of again showed a decreasing curve that approached 0. This again upholds the theorem 12.4. This shows that n grows much faster than h(n). Hence the ratio approaching 0 supports the idea that h(n) grows significantly slower than n.

The plots of the given 3 ratios all support the assertion that Theorem 12.4 states. Specifically the behavior of the plots being consistent with the expected height h(n) of the randomly built binary search tree being O(logn).

**Report**

* **Insert** your answers where indicated. This file with your inserted answers will **contain**, explain, and discuss the plots.
* In addition, the above report (this file) must contain the following information:
  + whether your data collection program works or not (this must be just ONE sentence)
  + the directions to compile and execute your program
* Good writing is expected.
* Recall that answers must be well written, documented, justified, and presented to get full credit.

**What you need to turn in:**

* Electronic copy of your source data collection program (separately attached to this assignment)
* Electronic copy of the csv file F
* Electronic copy of this file (including your inserted answers) (separately attached). Submit the file as a Microsoft Word or PDF file.

**Grading**

* See points distribution with questions/tasks.