EC 504 – Fall 2022 – Homework 3

Due Tuesday, Oct 25 on your SCC account by 11:59PM in the directory `/projectnb/alg504/username/HW3` **.**

Reading Assignment on GitHub – In CRLS add to past assignments Sorting_Data_Structure (Chapters 6, 7,8,9,10) , new material on Trees.pdf (12, 13) and TreeMath_B.pdf (Appendix B .5) and Huffman Coding 16.3

NOTE: MidTerm Exam in class on Nov 1, 2022 – only written questions on algorithm scaling, sorting and trees.

Written Exercise

1. (10 pts) Determine whether the following statements are true or false, and explain briefly why.

   (a) If doubling the size ($N \rightarrow 2N$) causes the execute time $T(N)$ of an algorithm to increase by a factor of 4, then $T(N) \in O(4N)$.

   (b) The height of a binary tree is the maximum number of edges from the root to any leaf path. The maximum number of nodes in a binary tree of height h is $2^{h+1} - 1$.

   (c) In a binary search tree with no repeated keys, deleting the node with key x, followed by deleting the node with key y, will result in the same search tree as deleting the node with key y, then deleting the node with key x.

   (d) Inserting numbers 1, . . . , n into a binary min-heap in that order will take $O(n)$ time.

   (e) The second smallest element in a binary min-heap with all elements with distinct values will always be a child of the root.

2. (15 pts) This exercise is to learn binary search tree operations

   (a) Draw the sequence of binary search trees which results from inserting the following values in left-to-right order, assuming no balancing. 15, 10, 31, 25, 34, 56, 78, 12, 14, 13

   (b) Starting from the tree at the end of the previous part, draw the sequence that results from deleting the following nodes in left-to-right order: 15, 31, 12, 14.

   (c) After deleting them draw the sequence of reinserting left-to-right in reverse order: 14, 12, 31, 15. in order into the tree and comment on the result?

3. (20 pts) Reading CRLS Chapter 6 and do the written

   (a) Exercises: 6.1-3, 6.1-4, 6.1-6, and 6.3-3

   (Note chapter 6 gives background to the coding exercise to use an array for a Max Heap.
   Also there is of course a nice Wikipedia article to look at https://en.wikipedia.org/wiki/Heapsort.)

(b) Exercises: 12.3-2, 12.3-3, 12.3-4, B.5-4

(Note that the degree of in an undirected graph (or tree) is the number links incident on the node. A leaf is a node with degree 1.)

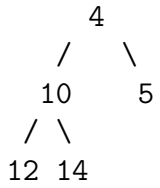
4. (15 pts) Determine whether the following statements are true or false, and explain briefly why.

   (a) A heapsort algorithm for a given list first forms a max-heap with the elements in that list, then extracts the elements of the heap one by one from the top. This algorithm will sort a list of n elements in time of O(log(n)).

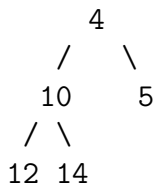
   (b) A connected undirected acyclic graph with N nodes and N-1 edges is a tree.

   (c) A binary heap of n elements is a full binary tree for all possible values of n.

   (d) The following tree is a valid min-heap.



   (e) The following tree can appear in a binomial min-heap. .



   (f) Given a array of positive integers $a[i]$, maximizing $\sum i * a[i]$ over permutations of the array requires sorting $a[i]$ in assigning order.

5. (15 pts) Given the following list of elements,

$$35, 22, 11, 98, 55, 66, 77, 80, 85, 90, 95$$

   (a) Draw the sequence of binary min-heaps which results from inserting the following values in the order in which they appear into an empty heap.

   (b) Compare this answer with inserting them into the BST tree.

   (c) Compare this answer with inserting them into the AVL tree.

6. (15 pts) You are interested in compression. Given a file with characters, you want to find the binary code which satisfies the prefix property (no conflicts) and which minimizes the number of bits required. As an example, consider an alphabet with 8 symbols, with relative weights (frequency) of appearance in an average text file give below:

alphabet:	A	L	G	O	R	I	T	H
weights:	68	20	5	30	18	15	19	12

(a) Determine the Huffman code by constructing a tree with **minimum external path length**: $\sum_{i=1}^{8} w_i d_i$. (Arrange tree with smaller weights to the left.)

(b) Identity the code for each letter and list the number of bits for each letter and compute the average number of bits per symbols in this code. Is it less than 3? (You can leave the answer as a fraction since getting the decimal value is difficult without a calculator.)

(c) Give an example of weights for these 8 symbols that would saturate 3 bits per letter. What would the Huffman tree look like? Is a Huffman code tree always a full tree?

7. (15 pts) Given the following list of $N = 10$ elements.

$$28 \quad 14 \quad 7 \quad 4 \quad 6 \quad 30 \quad 36 \quad 33 \quad 10 \quad 40$$

(a) Insert them sequentially into a BST (Binary Search Tree). Compute the total height $T_H(N)$ and the total depth $T_D(N)$, where H is the height of the root. (Note the height of a node is the longest path length to a leaf whereas its depth is its unique distance from the root.)

(b) Find H, $T_H(N)$, $T_D(N)$ and check the sum rule:

$$T_H(N) + T_D(N) \geq HN$$

(c) Insert them sequentially into an empty AVL tree, restoring the AVL property after each insertion. Show the AVL tree which results after each insertion and name the type of rotation (RR or LL zig-zig or versus RL or LR zig-zag).

(d) Has the final AVL tree decreased the total height $T_H(N)$ and the total depth $T_D(N)$? What are the new values? What is the new value of $T_H(N) + T_D(N)$?

Coding Exercise

8. (20pts) Implement a Max Heap for n elements as and array `int HeapArray[n+1];` of $n+1$ setting elements by placing the integers setting `HeapArray[0] = n` and copying the elements putting the elements in sequence into `HeapArray[i], for i = 1,2,..., n`

Provide the in `HW3_codes/heap.cpp` to enable:

```
(1) Insert random sequence to Heap array
(2) Bottom up Heapify for Max Heap
(3) Delete any key and restore Max Heap
(4) Insert new key and restore Max Heap
(4) Sort in place and print out array
```

Put final code with `Makefile` in `/projectnb/alg504/username/HW3`

Plotting and Analysis

9. (20pts) Analyze the behavior of this code and present plots to support this analysis.

```
(1) Plot timings for range of sizes n = 8, 16, 32,....2^20
so that you can fit performance to  T(n) = a + b n Log[n] + c n*n
to  T(n) = a + b n Log[n] + c n*n. You can use any fitting
program but it is really easy using gnuplot.

(2) (Extra Credit)  To improve the fit you can repeat several tunes
(say 10 times) each size n = 8, 16, 32,....2^20 by calling a randomizing
function in the sort.h. Then as described in HW2 you can make a table
of average time for each n and error bars.

(3)(Extra Credit) Now repeat Step (1) above  for n = 8, 16, 32,....2^20
finding best fit to  T(n) = a + b n Log[n] + c n*n  the averages
within the error bars.
```

Place these figures in `/projectnb/alg504/username/HW3` as well.