

EC 504 – Fall 2019 – Homework 2

Due Tuesday, Oct 8, 2019 in the beginning of class. Coding problems submitted in the directory /projectnb/alg504/yourname/HW2 on your SCC account by Tuesday, Oct. 8, 11:59PM.

Reading Assignment: CLRS Chapters 6, 7, 9, 10, 12 and Appendix B .5

1. (20 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 $T(N) = O(4N)$.
 - (b) The height of a binary tree is the maximum number of edges in any root to a 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 $1, \dots, n$ into a binary min-heap in that order will take $\Theta(n)$ 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.
 - (f) (CLRS 6.106) Is the array with values $\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 \rangle$ a max-heap?
2. (20 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) rat by reading CLRS Chapter 6 and do the written
 - (a) Exercises: exercises 6.1-3, 6.1-4, 6.1-6, 6.2-1 and 6.3-1

This Chapter 6 give a background to the fist 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>.

Coding Exercises – More details in class

There is template program on the GitHub but you all the basic framework can be adapted from the coding exercise in HW1. One goal of these coding exercises it to have a set of C tools and Gnuplot tools to reuse.

4. (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 $i = 1, 2, \dots, n$ (May choose to have an longer array with extra space and a save a value `heapSize` to tell how many are in the heap. This is a useful index in any case!

- (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

Implement your algorithm as a C/C++ function. Put final code with makefile `makeHeap` in your top level CCS account directory HW2.)

`/projectnb/alg504/username/HW2`

Extra credit to be pass in with the written exercises.

- (1) Plot timing for range of size $n = 8, 16, 32, \dots, 2^{20}$
- (2) Histogram the performance over random permutation of `UnsortedList100.txt`
- (3) Combine these two part to define the average for $n = 8, 16, 32, \dots, 2^{20}$ and fit $T(n) = a + b n \log[n] + c n^2$

Implement your algorithm as a C/C++ function. Put final code with makefile `makeHeap` on your top level CCS account in directory HW2 (e.g. folder!)

5. (20 pts) Suppose you are given two sorted arrays A, B of integer values, in increasing order, sizes n and m respectively, and $m + n$ is an odd value (so we can define the median value uniquely). Develop an algorithm for finding the median of the combined sorted arrays, and analyze the complexity of your algorithm. It is straight forward to develop an algorithm that is $O(\min(n, m))$. Explain in words. You should attempt to construct a code that is worst case complexity is $O(\min(n, m))$ to get good performance. (The written remarks should be pass in with the written exercises.)

- (1) Submit Timings for the input files provided.

Implement your algorithm as a C/C++ function. Put final code with makefile `makeAB` in your top level CCS account directory HW2 (e.g. folder!)

`/projectnb/alg504/username/HW2`