# EC 504 Midterm Pratice– Spring 2023

**Instructions: Put your name and BU ID on the first page. This exam is closed book and no notes – except for 2 pages of personal notes to be passed in the end. No use of laptops or internet. You have one hour and fifty minutes. Do the easy ones first.**

1. (20 pts) Answer True or False to each of the questions below. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions. No explanation is needed, but any explanation is likely to earn partial credit, and no explanation will not earn any credit if the answer is wrong.

   (a) $N^2 e^{-1/\ln(N)} \in \Theta(N e^{\ln(N)})$.

   (b) If $f_i(n) \in O(n^2)$, then $\sum_{i=1}^{n} f_i(n) \in O(n^3)$

   (c) The following tree is an AVL binary search tree – explain.

```
          11
        /    \
       9       23
     / \      /  \
    8   12  16   50
   / \     /       \
  1   10  10        80
```

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

   (e) The set $\Theta(f(n))$ is identical to the set $O(f(n)) \cap \Omega(f(n))$ (Note: $\cap$ means intersection of two sets.)

   (f) It is possible to formulate Quick Sort to be worst case $O(N \log N)$ including the cost of picking a suitable pivot.

   (g) Consider the array [2, 20, 3, 21, 90, 4,23, 22]. This array has elements in order of a min-heap.

   (h) A min-Heap (or priority queue) is a null tree or a root node with a minimum value plus a left and a right min-Heap subtrees.

   (i) The maximum subsequence sum algorithm finds $i$ and $j$ that maximize $\sum_{k=i}^{j} a[k]$ for an array $a[N]$ of positive and negative integers. The fastest one is a recursive algorithm with complexity $\Theta(N \log N)$.

   (j) $N^N \in O(\ e^{\ln(N!)}\ )$

2. (10 pts) Consider the binary tree illustrated below.

```
           *
       /      \
      +         -
     / \       / \
     *  8     -  10
   /  \      / \
  +    2    8   *
 /  \          / \
6    7        2   3
```

(a) List the node values in the order in which they would be found in an in-order traversal.

(b) List nodes in pre-order.

(c) List the nodes in post=order.

(d) Which order is uses on stack to do the arithmetic? . (HINT the stack order is like depth first search on this tree!) Evaluate the arithmetic using stack

3. (20 pts) Consider the following min-heap.

```
                1
           /          \
          10            2
       /     \        /    \
     11       12     17      3
    /  \     /  \   /  \    / \
  15    18 201   13 155  20 4    5
 / \    /
17 16 21
```

(a) Show the min-heap which results after inserting the element 7 in the heap. (Indicate the sequence of steps with arrows.) Then in this new heap show the steps required to delete element 1 and then delete 11. Draw the final min-heap.

(b) Re-arrange the original min-heap (repeated below) into a max-heap by a "bottom up" $O(N)$ algorithm. Describe the steps level by level.

```
                1
           /          \
          10            2
       /     \        /    \
     11       12     17      3
    /  \     /  \   /  \    / \
  15    18 201   13 155  20 4    5
 / \    /
17 16 21
```

(c) Using this example for a min-heap with the size $N = 18$ and $\mathbf{a[0]} = \mathbf{N}$. Describe in a few word an an $\Theta(N \log N)$ algorithm to sort `in place` the array elements $a[1], a[2], \cdots, a[N]$ in **descending order** (i.e. $a[1] > a[2] > \cdots > a[N]$ )

(d) Solve the recursion relation for the Heap, $T(H) = 2T(H-1) + c_0 H$ to show that "bottom up" scales like $T(H) = \Omega(N)$ (HINT: Use $N \simeq 2^H$)

4. (20 pts) This problem is to construct step by step BST and AVL trees given the following list of $N = 12$ elements.

$$1 \quad 2 \quad 12 \quad 7 \quad 3 \quad 6 \quad 30 \quad 36 \quad 37 \quad 2 \quad 40 \quad 51$$

(a) Insert them sequentially into a BST (Binary Search Tree).

(b) 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).

(c) Give the total height $T_H(N)$ and the total depth $T_D(N)$ for both the resulting BST and AVL.

(d) Compare the sum $T_H(N)+T_D(N)$ for the BST and AVL. For each compare these values with $HN$ where $H$ is the height of each tree.

5. (10 POINTS) Consider the following recursion relation.

$$T(N) = 3T(N/2) + N^k$$

(a) Substitute $T(N) = c_0 N^\gamma$ into the homogeneous equation (i.e. dropping $N^k$) and determine $\gamma$ and for what values of $k$ does the exact solution satisfy $T(N) = \Theta(N^\gamma)$.

(b) Now assume that $k = \gamma$ and find the exact solution in the form: $T(N) = c_0 N^\gamma + c_1 N^\gamma \log_2(N)$. (HINT: First show that $c_1 = 1$. Then Determine $c_0$ in terms of $T(1)$ by setting $N = 1$. )

6. (20 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 [1] give below:

| alphabet: | $V$ | $A$ | $M$ | $P$ | $I$ | $R$ | $E$ | $S$ |
|---|---|---|---|---|---|---|---|---|
| weights: | 4 | 40 | 16 | 7 | 30 | 18 | 75 | 8 |

(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 symbol 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?

[1] vampire: Origin mid 18th cent: from French, from Hungarian **vampir**, perhaps form Turkish **uber "witch"**. In European folklore, a monster with long pointed canine teeth!

7. (10 POINTS EXTRA CREDIT) Given an array of N-distinct positive integers $a[N]$ and a second array of N-distinct positive integers $b[N]$ to construct the sum:

$$\sum_{i=0}^{N-1} a[i] * b[i] \tag{1}$$

(a) Assume that $a[k]$ is sorted in ascending order. Give an $O(N \log N)$ algorithm that maximizing the sum $S$ by permuting the values of $b[k]$.

(b) Given the result of part (a) give an algorithm that minimizes $S$ in $O(N)$.

(c) If you are allowed to permute independently both $a[k]$ and $b[k]$ how many different solutions will maximize $S$?