Homework 4

CS6033 Design and Analysis of Algorithms I

Fall 2024

(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 10/23 by 1pm (Extended)
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 111 points

Note: This assignment has 3 pages.

1. (15 points)

Show the results of deleting B, O, U, in order, from the B-tree of Fig. 1 below, using the **one-pass top-down deletion algorithm** of B-trees as discussed in class and described in the Textbook pages 513-516. (Note that here the keys are letters. We assume that there is a total order among the letter keys that is the alphabetical order of the letters, i.e., A $<$ B $<$ C $< \cdots <$ Z.)

For each deletion, show the tree after each structural change and the final tree, and also state the case number being applied.

(**Note: 5 points for each deletion.**)

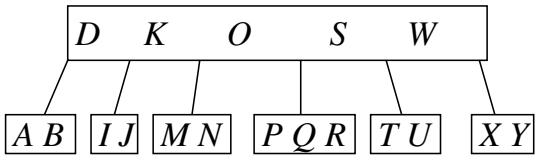

Figure 1: B-tree for Question 1, where $t = 3$. Note that a node (other than the root) cannot have fewer than 2 keys.

2. (16 points)

Recall the following "Baby" Master Theorem that we proved in class:

"Baby" Master Theorem: Let $a > 0, b > 1$ and $d \geq 0$ be constants. Then the solutions of a recurrence of the form

$$T(n) = a\,T(n/b) + \Theta(n^d),$$

where $T(n)$ is a constant for all small enough n, is

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a/b^d < 1, \\ \Theta(n^d \log n) & \text{if } a/b^d = 1, \\ \Theta(n^{\log_b a}) & \text{if } a/b^d > 1. \end{cases}$$

Apply this theorem to solve the recurrences in **(a) - (d)** below, where $T(n)$ is constant for $n \leq 2$. For each recurrence, justify how you apply the theorem (what are the values of a, b, d and

which case applies), and express the solution in the $\Theta()$ notation. **(Note: 4 points for each of (a)-(d).)**

(a) $T(n) = 4\,T(n/2) + n^2$.

(b) $T(n) = 2\,T(7n/10) + n^3$.

(c) $T(n) = 6\,T(n/4) + n\sqrt{n}$.

(d) $T(n) = 9\,T(n/2) + n^3$.

3. (15 points)
Suppose that $T(n)$ is a constant for $n \leq 2$. Solve the following recurrences for $T(n)$ by **repeated unfolding** and express the solutions in the $\Theta()$ notation.

(a) $T(n) = T(n-3) + n^3$. **(7 points)**

(b) $T(n) = 3\,T(n/3) + n\log^2 n$. **(8 points)**

4. (15 points)
Suppose that $T(n)$ is a constant for $n \leq 2$. Solve the following recurrence for $T(n)$ using the **recursion-tree method** and express it with the $\Theta()$ notation:

$$T(n) = 2\sqrt{n}\,T(\sqrt{n}) + cn$$

where $c > 0$ is a constant.
(**Hint:** See how many levels the recursion tree has.)

5. (10 points)
Consider the deterministic quick selection algorithm (the algorithm SELECT in the Textbook, to find the k-th smallest item from a set of n unsorted items in $O(n)$ worst-case time for any given integer $k \in [1, n]$). Suppose we modify the algorithm so that the items are divided into groups of 9 (rather than groups of 5). Does the algorithm still run in worst-case linear time? Give a new recurrence for the worst-case running time $T(n)$, and solve $T(n)$ and express it in terms of an asymptotic bound, i.e., either show that $T(n) = O(n)$, or show that $T(n) = \omega(n)$.

6. (20 points)
Professor Goodhead has developed a hardware priority queue device for his computer. This device can store up to p records, each containing a key and a small amount of satellite data (such as a pointer). With this device, INSERT and EXTRACT-MIN operations on the priority queue can be performed in $O(1)$ worst-case time each, as long as the records are stored in the device. Your task is to develop a sorting algorithm using the device. Suppose there are n records to be sorted, originally residing in main memory. (The hardware device can be viewed as a special-purpose cache.)

(a) Suppose $n \leq p$. Give a sorting algorithm that runs in $O(n)$ worst-case time using the hardware

priority queue. **(5 points)**

(b) Now suppose $n > p$. Design and analyze a **divide-and-conquer** sorting algorithm using the hardware priority queue that runs in $O(n \log_p n)$ worst-case time. Note that p is a given parameter and is **not** considered as a constant.
(**Hint:** Modify merge sort, and also use the hardware priority queue.) **(15 points)**

7. (20 points)
Let $x_1, x_2, \cdots, x_n$ be an unsorted sequence of real numbers (each of which can be positive, negative, or 0), where $n \geq 1$. Design and analyze a **divide-and-conquer** algorithm to find the **subsequence of consecutive elements** such that the sum of the numbers in it is minimum over all consecutive subsequences. Here the "subsequence" must have length at least 1; i.e., it must be non-empty. Your algorithm should run in $O(n \log n)$ worst-case time.