Notes on Grading

- You can write “I don’t know” for any question and receive 25% credit. You can take this option for any numbered problem, but not for part of a problem. For example, you can answer 3.1 and write “I don’t know” for 3.2, but you can’t write part of the solution for 3.2 and then write “I don’t know” for the rest.

- You get a 10% bonus for typing your homework. You are encouraged to use \LaTeX. You must type your entire homework to receive the bonus. The 10% bonus does not apply to problems answered with “I don’t know.”

External-memory algorithms

- (5 points) In class, we analyzed the amortized I/O complexity of inserting items into a $B^*$-tree. What is the worst-case I/O complexity of inserting an item into a $B^*$-tree?

- (5 points) How long would it take to sort $N$ items by inserting them into a $B^*$-tree and then iterating over them in order?

- (5 points) What about if you use an LSM-tree?

- (5 points) How do these compare with the I/O cost of sorting using $M/B$-way merge sort?

- (10 points) Analyze the I/O complexity of the following matrix multiplication algorithm:
Algorithm 1 Recursive matrix multiply algorithm. TL, TR, BL, and BR indicate the top-left, top-right, bottom-left, and bottom-right quadrants of a matrix.

function MM-Scan(A,B)
    if $|A| = 1 \times 1$ then
        return $A[0,0] \times B[0,0]$
    else
        $X_{TL} \leftarrow$ MM-Scan($A_{TL}, B_{TL}$)
        $X_{TR} \leftarrow$ MM-Scan($A_{TL}, B_{TR}$)
        $X_{BL} \leftarrow$ MM-Scan($A_{BL}, B_{TL}$)
        $X_{BR} \leftarrow$ MM-Scan($A_{BL}, B_{TR}$)
        $Y_{TL} \leftarrow$ MM-Scan($A_{TR}, B_{BL}$)
        $Y_{TR} \leftarrow$ MM-Scan($A_{TR}, B_{BR}$)
        $Y_{BL} \leftarrow$ MM-Scan($A_{BR}, B_{BL}$)
        $Y_{BR} \leftarrow$ MM-Scan($A_{BR}, B_{BR}$)
        $C \leftarrow X + Y$
        return $C$
    end if
end function

Solution Note in all problems, you may assume that memory has size $M = 1$.

- $O\left(\frac{\log_B N/M}{\epsilon B^{1-\epsilon}}\right)$. Note the amortized cost per insert is $O\left(\frac{\log_B N/M}{\epsilon B^{1-\epsilon}}\right)$. 2 points if you accidentally gave the amortized time. 2 point for B-tree time (i.e. $O(\log_B N)$). We took off at least one point for important mistakes, such as writing $\log N$ instead of $\log_B N$.

- Inserting $N$ items requires $O(N \frac{\log_B N/M}{\epsilon B^{1-\epsilon}}) \ I/O$s. Scanning over all the items in the tree requires a point query (costing $O(\frac{\log_B N/M}{\epsilon}) \ I/O$s) followed by $N/B \ I/O$s to read all the elements in the tree, for a total cost of $O(N \frac{\log_B N/M}{\epsilon B^{1-\epsilon}} + \frac{\log_B N/M}{\epsilon} + N/B) = O(N \frac{\log_B N/M}{\epsilon B^{1-\epsilon}})$. You can get partial credit (1 point each) for indicating the times for the inserts and for the point query and scan. We took of points similar to above for important mistakes such as writing $\log N$ instead of $\log_B N$.

- Inserting $N$ items into an LSM tree requires $O(N \frac{\log_M N/M}{B}) \ I/O$s. Scanning the items requires point queries to the beginning of each tree, for a cost of $O(\log N \log_B N/M) \ I/O$s, and then $N/B \ I/O$s to read all the items. The total cost is therefore $O(N \frac{\log_M N/M}{B}) \ I/O$s. Similar grading policy to the previous question.

- Sorting costs $O(N/B \log_M N/M)$. Using an LSM tree is faster than using a $B^\epsilon$-tree. Sorting is faster than an LSM tree when $M > B^2$. 1 point for the sorting cost. 1 point for realizing that sorting is faster than a $B^\epsilon$-tree. 2 points for saying
something meaningful about sorting vs LSM-trees. Full credit for seeing that sorting is faster than LSM when $M > B^2$.

- Let $N$ be the number of elements in the matrix and $T(N)$ be the worst-case I/O complexity. Then

$$T(N) = \begin{cases} 8T(N/4) + O(N/B) & \text{if } N \neq O(M) \\ O(N/B) & \text{if } N = O(M) \end{cases}$$

The base case makes it a little difficult to use the Master Theorem, so we do it by hand. $T(N) = 8^i T(N/4^i) + O(2^i N/B)$. Solving for $i$ gives $i = \log_4 N/M$. Plugging in gives $T(N) = O((N/M)^{3/2}M/B + \sqrt{N/M}N/B) = O((N^3/B)^{3/2}/B^2M)$. (Note I omitted several steps using log identities.)

If you accidentally solved the problem using $N$ as the height/width of the matrices, that’s ok – you can still solve it using the same basic approach. You should get $O(\frac{N^3}{B^2M})$.

4 points for the recurrence. 2 point for solving for $i$. 4 points for solving for the overall solution. 5 point bonus for solving for general $M$. 

3