Midterm Review

Recurrences, B-Trees, and More

Section Feedback

Please put any notes on the index card!

Ideas:

- Balance of practice problems vs. TA mini-presentations
- Technical stuff (handwriting, organization, clarity of speaking)
- Any other ways you notice section could be improved!

Survey Results

Recurrence Relations, B-Trees, AVL Trees, Big-Oh...

Solving Recurrences: Theory

1. **Look for tricks!** (try to understand what the code is doing)
2. **Write the recurrence** (read the code, fill in the template)

$$T(n) = \begin{cases} \underline{\text{Base Work}} & \text{if} \quad n\ \underline{\text{Small Enough}} \\ \underline{\text{Recursive Work}} + \underline{\text{Non-Recursive Work}} & \text{otherwise} \end{cases}$$

3. **Solve the recurrence** (tree or expansion to find a closed form)

$$T(n) = \sum_{i=0}^{\text{levels in tree}} (\underline{\text{Work Per Node on Level } i}) \cdot (\underline{\text{Number of Nodes on Level } i})$$

4. **Simplify** (probably using the summations, which will be given on the exam)

Solving Recurrences: Shortcuts & Sanity Checks

$T(n) = O(1) + T(n/2)$	logarithmic
$T(n) = O(1) + 2T(n/2)$	linear
$T(n) = O(1) + T(n\text{-}1)$	linear
$T(n) = O(n) + T(n\text{-}1)$	quadratic
$T(n) = O(1) + 2T(n\text{-}1)$	exponential
$T(n) = O(n) + T(n/2)$	linear
$T(n) = O(n) + 2T(n/2)$	O(n `log` n)

Solving Recurrences: Example 1 (18wi)

Give a base case and a recurrence for the runtime of the following function. Use variables appropriately for constants (e.g. c1, c2, etc.) in your recurrence (you do not need to attempt to count the exact number of operations). **<u>YOU DO NOT NEED TO SOLVE</u>** this recurrence.

```
int mystery(int n) {
    int x = 0;
    if (n <= 1) {
        for (int i = 7 * n; i > 0; i -= n) {
            x += 8;
        }
        return x;
    } else {
        x = 5 * mystery(n - 1);
        for (int i = 0; i < n * n; i += n) {
            x += 7;
        }
        return x + mystery(n - 1) * mystery(n - 3);
    }
}
```

$$T(n) = \begin{cases} \underline{\text{Base Work}} & \text{if} \quad n \ \underline{\text{Small Enough}} \\ \underline{\text{Recursive Work}} + \underline{\text{Non-Recursive Work}} & \text{otherwise} \end{cases}$$

Solving Recurrences: Example 2 (17au)

Suppose that the running time of an algorithm satisfies the recurrence relationship:

$$T(1) = 9.$$

and

$$T(N) = 2 * T(N/2) + 7N \qquad \text{for integers } N > 1$$

Find the closed form for T(N). **You may assume that N is a power of 2**. Your answer should *not* be in Big-Oh notation – show the relevant *exact* constants in your answer (e.g. don't use "c1, c2" in your answer). You should not have any summation symbols in your answer. The list of summations on the last page of the exam may be useful. ***Show your work***.

$$T(n) = \begin{cases} \underline{\text{Base Work}} & \text{if } \quad n \ \underline{\text{Small Enough}} \\ \underline{\text{Recursive Work}} + \underline{\text{Non-Recursive Work}} & \text{otherwise} \end{cases}$$

$$T(n) = \sum_{i=0}^{\text{levels in tree}} (\underline{\text{Work Per Node on Level } i}) \cdot (\underline{\text{Number of Nodes on Level } i})$$

B-Trees: Theory

Problem:
For data structures that don't fit in memory, every node access means another (slow) page access.

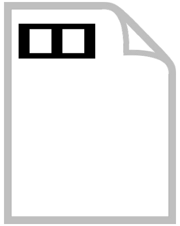
Solution:
A shallow tree with big nodes that take up *almost* all of a page, but **not** two pages.

Tree Properties:
Internal Nodes:
- Have at least $\lceil M / 2 \rceil$ pointers (children)
- Only contain keys, which are the smallest key of the right subtree.

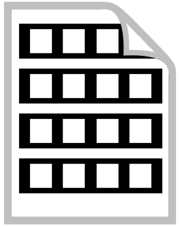
Leaf Nodes:
- Have at least $\lceil L / 2 \rceil$ key-value pairs
- All at the same depth

Root Node: is a leaf *or* has between 2 and M children

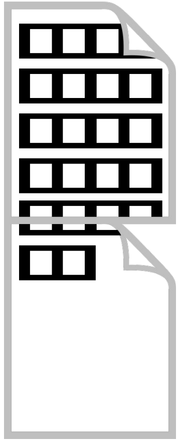
M Too Small: tree has many small nodes, requiring many page accesses to traverse the tree 😞



M Just Right: tree has few large nodes, requiring few page accesses to traverse the tree ❤



M Too Big: tree has few large nodes, but each one requires multiple page accesses, so traversing the tree requires many page accesses 😞



B-Trees: Example 1 (M & L) (13wi)

a) (4 pts) Given a B-tree (as defined in lecture and in Weiss) with M = 3 and L = 20, if you have inserted 100 items into the tree, what is **the minimum and maximum height of the tree**? Give an exact number, not a formula for your answer.

b) (4 pts) Given the following parameters for a B-tree with M= 37 and L = 5:

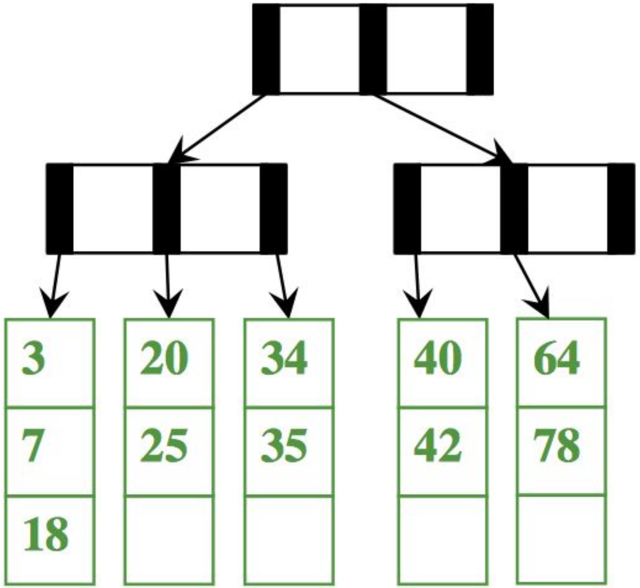
Key Size = 6 bytes

Pointer Size = 8 bytes

Data Size = 100 bytes per record (***includes*** the key)

Assuming that M and L were chosen appropriately, **what is the likely size of a disk block** on the machine where this implementation will be deployed? Give a numeric answer and a short justification based on one or more equations using the parameter values above.

B-Trees: Example 2 (Insert & Delete) (13wi)



a) (2pts) In the B-Tree shown on the left, please write in the appropriate values for the interior nodes.
b) (4 pts) Starting with the B-tree shown on the left, insert **9**. Draw and circle the resulting tree (*including values for interior nodes*) below. Use the method for insertion described in lecture.
c) (4 pts) Starting with the ***original*** B-tree shown above on the left (***before*** **inserting 9**), delete **78**. Draw and circle the resulting tree (*including values for interior nodes)* below. Use the method for deletion described in lecture.

Good Luck!