Practice Quiz 1

- The real Quiz 1 will be held on Thursday, October 15, in lecture.
- There will be a quiz review on Friday, October 9, during recitation.
- The quiz will be closed book. You may use one double sided Letter \((8\frac{1}{2}'' \times 11'')\) or A4 crib sheet. No calculators or programmable devices are permitted.
- Write your solutions in the space provided. Extra scratch paper may be provided if you need more room, although your answer should fit in the given space.
- Do not waste time re-deriving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Generally, a problem’s point value is an indication of how much time to spend on it.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!
Problem 1. Recurrences [15 points] (4 parts)
Solve the following recurrences by giving tight $\Theta$-notation bounds. You do not need to justify your answers, but any justification that you provide will help when assigning partial credit. As usual, assume that for $n \leq 10$, $T(n) = O(1)$.

(a) [2 points]  
$$T(n) = 3T(n/3) + 0.5n \log(n).$$

Solution: Using case 2 of the Master’s Method gives us $T(n) = \Theta(n \log^2 n)$.

(b) [4 points]  
$$T(n) = 9T(\sqrt[3]{n}) + \Theta(\log(n)).$$

Solution: Let $n = 2^m$. Then the recurrence becomes $T(2^m) = 9T(2^{m/3}) + \Theta(m)$. Setting $S(m) = T(2^m)$ gives us $S(m) = 9S(m/3) + \Theta(m)$. Using case 1 of the Master Method gives us $S(m) = \Theta(m^2)$ or $T(n) = \Theta(\log^2 n)$.
(c) [4 points] \( T(n) = T(2n/7) + T(5n/7) + \Theta(n). \)

**Solution:** The Master Theorem doesn’t apply here. Draw recursion tree. At each level, do \( \Theta(n) \) work. Number of levels is \( \log_{7/5} n = \Theta(\lg n) \), so guess \( T(n) = \Theta(n \lg n) \) and use the substitution method to verify guess.

(d) [5 points] Define \( T(n) \) by the recursion \( T(n) = 9(T(\lfloor n/3 \rfloor) - 1) + n^3 + 2n \) for \( n \geq 1 \), with base case \( T(0) = 0 \). Prove \( T(n) \leq 3n^3/2 \).

**Solution:** Inductive hypothesis: \( T(n) \leq 3n^3/2 - n \).
Base case: \( T(0) = 0 \leq 30^3/2 - 0 \).
Inductive step:

\[
T(n) = 9(T(\lfloor n/3 \rfloor) - 1) + n^3 + 2n \\
\leq 9(3\lfloor n/3 \rfloor^3/2 - \lfloor n/3 \rfloor - 1) + n^3 + 2n \\
\leq 9(3(n/3)^3/2 - n/3) + n^3 + 2n \\
= 3n^3/2 - n.
\]
Problem 2. True or False, and Justify [13 points] (6 parts)
Circle T or F for each of the following statements, and briefly explain why. Your justification is worth more points than your true-or-false designation.

(a) T F [2 points] To achieve asymptotically optimal performance, a skip list must use promotion probability $p = 0.5$.

   Solution: False. Any promotion probability between 0 and 1 achieves the same asymptotic performance.

(b) T F [2 points] Universal hashing requires that you know what elements you’ll hash in advance.

   Solution: False. Perfect hashing requires knowing the elements in advance. Universal hashing does not.
(c) T F  [2 points] In a B-tree, the maximum number of children of an internal non-root node is at most twice the minimum of number of children.

Solution: True. For a B-tree with parameter $t$, there are at least $t$ and at most $2t$ children.

(d) T F  [2 points] A rotate operation on balanced tree always increases the depth of at least one node and decreases the depth of at least one node.

Solution: TRUE. Every rotate operation demotes the root of a subtree and promotes a new node to that position. Promotion decreases a node’s depth. See CLRS 13.2 or Lecture 7 for a description/illustration of rotation.
(e) T F [3 points] In a B-tree of minimum parameter $t$, every node contains at least $t - 1$ elements.

**Solution:** FALSE, Normally nodes in a B-tree of parameter $t$ must have between $t - 1$ and $2t - 1$ elements, but the root node is exempted from this rule to accommodate trees with fewer than $t - 1$ elements. Consider a B-tree of parameter 3 that contains one element: the root node contains 1 element, but here $t - 1 = 2$. 
Problem 3. **Short Answer** [13 points] (4 parts)

Give brief, but complete, answers to the following questions.

(a) [2 points] What is the expected difference between the depth of the deepest leaf and the depth of the least deep leaf in a 2-3-4 tree containing \( N \) elements?

**Solution:** Zero. All leaves are at the same level.

(b) [3 points] Show how to find a divisor \( d \) of \( N \) such that \( d \) is not 1 or \( N \), given \( x, y \) such that \( x^2 = y^2 \mod N \) and \( x \neq y \mod N \), \( x \neq -y \mod N \).

**Solution:** Compute \( \gcd(x - y, N) \) or compute \( \gcd(x + y, N) \)
(c) [4 points] Let \( \mathcal{H} \) be a universal hash family mapping \([1 \ldots N]\) to \([1 \ldots M]\). Let \( X_{ijh} \) be the indicator variable for a collision between \( i \) and \( j \) under the hash function \( h \), \( i \neq j \) and \( h \in \mathcal{H} \). What is \( E(X_{ijh}) \), where the expectation is taken over \( i, j, \) and \( h \)?

**Solution:** By the definition of a universal hash family, the probability of a collision is \( 1/M \), regardless of \( i \) and \( j \). So the expected value of the indicator variable is \( 1/M \).

(d) [4 points] Consider a balanced binary tree of \( n \) elements in which each node has an integer value. The weight of a path is the sum of the values of the nodes visited by the path. Give an optimal algorithm that computes the maximum possible weight of a path in the binary tree, starting at the root. What is the running time of your algorithm?

**Solution:** If the tree consists of a single node (ie, we are at a leaf), then the answer is simply the weight of that node. Otherwise, we recurse using

\[
\text{MAXPATH}(\text{root}) = w(\text{root}) + \max \left\{ \begin{array}{c}
\text{MAXPATH}(\text{root} \rightarrow \text{left}), \\
\text{MAXPATH}(\text{root} \rightarrow \text{right})
\end{array} \right\}
\]

This algorithm accesses each node exactly once, so the runtime is \( \Theta(n) \).
Problem 4. Slightly-Longer Short Answer [29 points] (5 parts)

Give brief, but complete, answers to the following questions.

(a) [5 points] A sequence of $n$ operations is performed, so that the $i^{th}$ operation costs $\lg(i)$ if $i$ is an exact power of 2, and 1 otherwise. That is the amortized cost per operation?

Solution: True. Let $c(i)$ be the cost of the $i^{th}$ operation

$$c(i) = \begin{cases} 
\lg i & \text{if } i = 2^k, \text{ } k \text{ integer} \\
1 & \text{otherwise}
\end{cases}$$

For any $n$, the total cost of $n$ operations is

$$\sum_{i=1}^{n} c(i) = n - \lfloor \lg n \rfloor + \sum_{i=1}^{\lfloor \lg n \rfloor} i$$

$$= n + \Theta(\lg^2(n)) = \Theta(n)$$

Therefore, the amortized cost per operation is $\Theta(1)$.

(b) [6 points] Define set $S = \{ A \in Z_N^* | A = x^{(N-1)/2} \pmod{N} \}$ for a prime $N$. Is $S$ a sub-group of $Z_N^*$? If so, what can you say about the size of set $S$?

Solution: $S$ is a sub-group of $Z_N^*$:

• identity: $1 \in S$
• closure: given $A = x^{(N-1)/2} \pmod{N}$ and $B = y^{(N-1)/2} \pmod{N}$, $AB = (xy)^{(N-1)/2} \pmod{N} \in S$
• inverse: given $A = x^{(N-1)/2} \pmod{N}$, since $x \neq 0$ and $N$ is a prime, it must be that $x \in Z_N^*$. Thus, $x$ must have an inverse, so $A^{-1} = (x^{-1})^{(N-1)/2} \pmod{N}$, which is also in $S$.

By Lagrange’s Theorem, we know that $|S|$ divides $|Z_N^*|$, but in this case, we can say something stronger: $A = x^{(N-1)/2} \pmod{N}$, so $A^2 = x^{(N-1)} = 1 \pmod{N}$ (by Fermat’s Little Theorem), which has only two solutions (1 and $N-1$ by Modular Sqrt Theorem), so $|S| = 2$. 

(c) [6 points] Given two sets $A$ and $B$ of $n$ integers, give an efficient deterministic algorithm to find $A \cap B$ and analyze its runtime. Can you do better with randomization? Explain.

**Solution:** For the deterministic algorithm, sort each list. Then, iterate through the elements looking for elements common to both arrays. This takes $O(n \log n)$ time. Using randomization, hash the elements of $A$. Then iterate through $B$, looking up elements in the hash of $A$. The expected running time is $O(n)$.

(d) [6 points]

Consider an array $A$ of $n$ integers. Find all elements occurring at least $n/3$ times.

**Solution:** Replace the $i$th element with a pair $(A[i], i)$ to make them all distinct. Comparison between pairs is done by comparing the first elements and breaking ties by the second elements.

Use the select algorithm to find the elements of ranks $n/3$, $2n/3$ and $n$. If an element occurs at least $n/3$ times, it must be one of those three elements. Check all three to see if any of them occurs at least $n/3$ times. The running time is $O(n)$. 

(e) [6 points]
Consider a sorted array $A$ of size $n$, containing distinct integers. Give an $O(lg\ n)$ algorithm to find an index $i$ such that $A[i] = i$ (or none, if no such index exists). Does your algorithm still work if $A$ contains repeat elements? Explain why or why not.

Solution: Consider $A[n/2]$. If $A[n/2] = n/2$, then we’re done. Otherwise, if $A[n/2] > n/2$, recurse on $A[1\ldots n/2 - 1]$. If $A[n/2] < n/2$, recurse on $A[n/2 + 1\ldots n]$. The runtime is $O(lg\ n)$.

If there are repeat elements, then we can no longer ensure that the answer is on one side of the median, since it may be true that $A[1] = 1$ and $A[n] = n$, for any value of the median between 2 and $n - 1$. 
Problem 5. Searching in multiple lists [8 points]

Consider two disjoint sorted arrays \( A[1 \ldots m] \) and \( B[1 \ldots n] \). Find an \( O(\log k) \) time algorithm for computing the \( k \)-th smallest element in the union of the two arrays.

Solution: Consider \( A[k/2] \) and \( B[k/2] \). Without loss of generality, assume \( A[k/2] < B[k/2] \). Then \( A[k/2] \) is greater than at most \( k \) elements. Furthermore the elements \( A[1 \ldots k/2 - 1] \) are all less than the \( k \)-th element, so we can eliminate them. Similarly, \( B[k/2] \) is greater than at least \( k \) elements, so the elements \( B[k/2 + 1 \ldots n] \) are all larger than the \( k \)-th element. We can therefore eliminate them too. We are therefore left with two subarrays, and we now want to find the \( k/2 \)-th element (since we eliminated \( k/2 \) elements that were guaranteed to be less than the \( k \)-th element). This divide-and-conquer algorithm follows the recursion \( T(k) = T(k/2) + 1 \), which is \( O(\log k) \).
Problem 6. The Eccentric Landlord [8 points] (2 parts)
Your construction firm is hired to build an apartment building for an eccentric landlord. He wants his building to be a square of size \( M \times M \), containing \( M^2 \) identical square apartments.

The landlord will add one tenant a day. When he can’t fit a new tenant, he will tear down two sides of the building and have new walls built, expanding it an \((M + 1) \times (M + 1)\) building.

It costs your firm $1 to build one apartment’s exterior wall; your other costs (demolishing exterior walls, building interior walls, etc) are negligible. Your costs will be:

Day 1: $4 (build four walls)
Day 2: $6 (expand to 2x2)
Day 3: $0 (tenant moves into empty unit)
Day 4: $0 (tenant moves into empty unit)
Day 5: $8 (expand to 3x3)
Day 6: $0 (tenant moves into empty unit)
...

The costs incurred on day 2 are shown below.

(a) [4 points] What will your asymptotic aggregate cost be for this project? Give your answer as a function of the number of days elapsed.

Solution: We build between 1 and 2 new walls for each tenant, and tenants arrive at a rate of 1/day, so the cost per day is \( O(1) \). The aggregate cost is therefore \( O(n) \).
(b) [4 points] You convince the landlord to expand his building in bigger steps: Whenever he can’t fit a new tenant, he will double the building side length (instead of increasing it by one unit). Repeat your analysis from part (a) for this new condition.

**Solution:** We must spend $O(\sqrt{n})$ dollars to fit $O(n)$ tenants, since the cost is dominated by the most current addition.
Problem 7. Chemical testing [15 points]
A chemistry lab is given \( n \) samples, with the goal of determining which of the samples contain traces of a foreign substance. It is assumed that only few (say, at most \( t \)) samples test positive. The tests are very sensitive, and can detect even the slightest trace of the substance in a sample. However, each test is very expensive. Because of that, the lab decided to test "sample pools" instead. Each pool contains a mixture of some of the samples (each sample can participate in several pools). A test of a pool returns positive if any of the samples contributing to the pool contains a trace of the substance.

Design a testing method that correctly determines the positive samples using only \( O(t \log n) \) tests. The method can be adaptive, i.e., the choice of the next test can depend on the outcomes of the previous tests.

Solution: There exist several related algorithms that solve this problem. The simplest one proceeds as follows: we divide the samples into \( 2t \) groups of size \( \frac{n}{2t} \) each. We pool and test each group. Since at most \( t \) groups are positive, we can label at least \( \frac{n}{2} \) samples as negative. Then we recurse on the remaining \( \frac{n}{2} \) samples. It is easy to see that the number of recursion levels is \( O(\log n) \). Since \( 2t \) tests are performed at each level, the total number of tests is at most \( O(t \log n) \).

A different algorithm divide the samples into two groups of size \( \frac{n}{2} \). Both groups are tested, and the algorithm recurses on group(s) that test positive. As before, the recursion tree has depth \( \log n \), since we divide the group size by 2 at each level. Moreover, the recursion tree contains at most \( t \) leaves. Therefore the total number of tree nodes (and therefore tests) is \( O(t \log n) \).