CS 240: Data Structures and Data Management **Spring 2023**

Final Practice Problem

Note: This is a sample of problems designed to help prepare for the final exam. These problems do not encompass the entire coverage of the exam (and have a heavy emphasis on post-midterm material even though the exam is cumulative), nor do they reflect the length of the exam, and should not be used as a reference for its content.

1. **True or False**

   (a) Final exam for CS 240 is August 10th, 9am at PAC.

   (b) A SkipList with n keys and height h must have a total of $\Theta(nh)$ nodes

   (c) In a SkipList with n keys, search operation is guaranteed to have runtime of $O(\log n)$ in the worst-case.

   (d) If we perform odd number of searches in linked list with transpose heuristic, the resulting list will always be different from initial list.

   (e) The root of compressed trie always tests the first bit

   (f) Linear probing requires two hash functions.

   (g) If is α is 1 for hashing with chaining, then next `insert` is guaranteed to fail

   (h) In Cuckoo hashing, two tables can be different size

   (i) Given $h_0(k) = k \mod 7$ with double hashing, each of size 7, $f(k) = (k \mod 6) + 1$ is more suitable to be used as $h_1(k)$ than $g(k) = k^2 \mod 7$

   (j) If the root of a quadtree represents the region $[0, 128) \times [0, 128)$ while deepest internal node represents the region $[88, 92) \times [24, 28)$, the height of quadtree is 6.

   (k) A range tree storing n points - where n is of the form $2^{h+1} - 1$ - such that the primary tree and all associated trees are perfect (i.e. all levels are full) has $(n+1)\log(n)$ nodes in the primary trees and all associated trees combined

   (l) A modified version of the Boyer-Moore algorithm that uses a first-occurrence array (denoting the index of the first occurrence of the argument character) instead of a last-occurrence array will always successfully find the first occurrence of a pattern P in a text T (if it appears in T).

2. **Revenge of First Principle**

   (a) Prove from first principles that $7n^4 - 5n^2 + 6 \in \Theta(n^4)$

   (b) Prove from first principles that $14n + 22 \in o(nlogn)$

3. **Run-time Analysis**

   When Prashanth goes to sleep, his brain generates a number n and then runs an algorithm called $Dream(n, hour)$. This algorithm has two inputs: the number n and the hour at which Prashanth goes to sleep, which is either 9:30 PM, 10:00 PM, or 10:30 PM. The runtime of $Dream(n, hour)$ is n^i computations where i is the number of hours that Prashanth sleeps.

   If Prashanth sleeps at 9:30PM, he will get between 8 to 9 hours of sleep. If he goes to sleep at 10:00 PM, he will get exactly 7.5 hours of sleep. If he sleeps at 10:30 PM, he will get between 5 to 7 hours of sleep.

   Prove or disprove the following statements:

   (a) In the worst case, the running time of $Dream$ is $\Theta(n^9)$.

(b) In the best case, the running time of *Dream* is $\omega(n^5)$

(c) If Prashanth sleeps at 10:00 PM, the running time of *Dream* is $\Omega(n^{7.5})$

(d) The running time of *Dream* is $O(i^n)$.

(e) After learning about some data structures in CS 240 that could help speed up his algorithm, Prashanth speeds up his *Dream* algorithm so that it runs in $\Theta(i)$ time where i is still the number of hours that he sleeps. The running time of *Dream* is $o(1)$.

4. **Partial Sum Part 2**
Consider the problem where we have a sequence of n elements: $S = a_1, a_2, ..., a_n$, and 3 operations:

- $Add(S, b) \rightarrow a_1, a_2, ..., a_n, b$
- $Update(S, i, \Delta) \rightarrow a_1, ..., a_{i-1}, \Delta, a_{i+1}, ..., a_n$
- $PartialSum(S, k) \rightarrow \sum_{i=1}^{k} a_i$

Design a data structure that can perform each of these operations in $O(\log n)$ expected time.
NOTE: we did solve this problem in tutorial 07 using AVL-tree. So, do NOT use AVL-tree this time.

5. **Find Largest x in kd-tree**
Given a 2D kd-tree with n points, design an algorithm that finds a point with largest x-coordinate. For simplicity, you may assume n is a power of 4.

6. **Consecutive Trie Strings**
Given an uncompressed trie T that stores a list of binary strings, design an algorithm $Consecutive(b_1, b_2)$ that takes two binary strings in T as input, and outputs true if the strings are consecutive in pre-order traversal of the trie, and outputs false otherwise. Assume that branches are ordered as \$, 0, 1. The runtime should be bounded by $O(|b_1| + |b_2|)$.

For example, suppose T stores $\{000, 01, 0110, 101, 11\}$.

$Consecutive(0110, 101)$ outputs true.
$Consecutive(01, 000)$ outputs true.
$Consecutive(11, 000)$ outputs false.

7. **MaxDiff**
Consider an array A of n integers. We want to implement a range query called $MaxDiff(i, j)$ which will find the maximal difference between two elements from $A[i]$ to $A[j]$ inclusive, for $i < j$. For example, suppose our array A is:

$A = 3\ 0\ 5\ 4\ 5\ 6\ 3\ 4\ 5\ 7\ 9\ 8\ 1\ 0\ 1$

If we run the query $MaxDiff(2, 9)$, then the subarray from indices 2 to 9 is:

$A[2 \dots 9] = 5\ 4\ 5\ 6\ 3\ 4\ 5\ 7$

The largest number is 7 and the smallest number is 3, so the maximal difference is $7 - 3 = 4$. The query $MaxDiff(2, 9)$ should return 4.

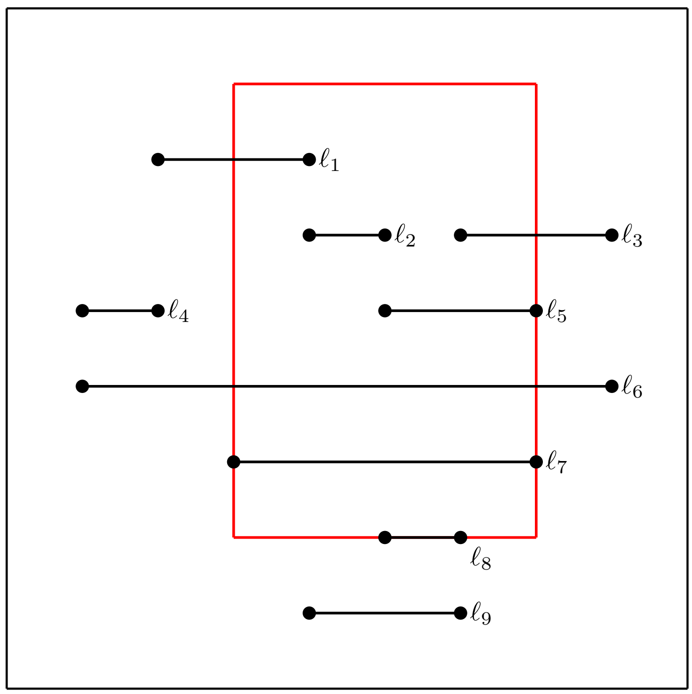
Design a data structure for A with space complexity $O(n)$ to answer queries of the form $MaxDiff(i, j)$ in $O(\log n)$ time. There are no constraints on the runtime for preprocessing the array into the data structure.

8. **Range Tree**
Suppose you have a set of n horizontal line segments in a plane, where line segment ℓ_i has coordinates (x_i, y_i) and (x'_i, y_i). Assume that all coordinates are integers.

For each of the range-search queries below, design a data structure and provide an algorithm to answer the queries in $O(\log^3 n + s)$ time, where s is the number of lines reported. Each range-search query is a rectangle of the form $[a, b] \times [c, d]$.

   (a) The algorithm reports all line segments that are entirely contained inside the query rectangle. For the example below, the algorithm would return ℓ_2, ℓ_5, ℓ_7 and ℓ_8.

   (b) The algorithm reports all line segments that intersect the query rectangle. For the example below, the algorithm reports all line segments **except** ℓ_4 and ℓ_9.



9. **Hashing**
Let $p \geq 3$ be prime, and consider the universe of keys $U = \{0, 1, \ldots, p^2 - 1\}$.

   (a) With a hash table of size p, and using double hashing with $h_0(k) = k \bmod p$ and $h_1(k) = \lfloor k/p \rfloor + 1$, give a sequence of **two** keys to be inserted that results in failure.

   (b) With two hash tables of sizes p and $(p - 1)$, and using cuckoo hashing with $h_0(k) = k \bmod p$ and $h_1(k) = k \bmod (p - 1)$, give a sequence of **four** keys to be inserted that results in failure.

10. **Chaotic Karp-Rabin**
Consider following hash function to be used in Karp-Rabin algorithm.

$$h(P) = \text{Number of vowels in } P$$

   (a) Describe how fast-update on fingerprint must be done.

   (b) Find an example pattern and text that leads to a case that maximizes execution of `strcmp`.

11. **Basics of Range Tree** Consider the following points being stored in a 2D range tree: $(2, 12)$, $(17, 77)$, $(23, 92)$, $(40, 47)$, $(55, 91)$, $(67, 27)$, $(89, 79)$, $(99, 53)$, $(10, 23)$, $(35, 7)$, $(61, 40)$, $(95, 56)$, $(22, 42)$, $(88, 15)$, $(42, 2)$.
    (a) Draw the primary tree for this range tree.
    (b) Draw the corresponding associate trees for the points $(88, 15)$, $(61, 40)$ and $(67, 27)$.
    (c) Perform a range-search with the query rectangle $[35, 90] \times [5, 30]$, indicating all boundary nodes and topmost inside nodes.

12. **Modified AVL Tree**
    We consider a modified version of AVL trees where the height difference between the right and left subtrees of any node is in the range $[-2, 2]$ instead of $[-1, 1]$. These are called AVL-2 trees. Prove that the height of an AVL-2 tree on n nodes is in $O(\log n)$.

13. **Tracing SkipList**
    Insert the numbers 12, 11, 13, 10, 20 into an empty skip-list using the coin flips HHTHTHTTHHHT. Then delete the keys 13 and 20.

14. **Boyer More**
    Let the text be "BATARATATATCAT", and the pattern be "TATCAT", show the full Boyer Moore Algorithm applied to this string search.