CS 240: Data Structures and Data Management	Winter 2024

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 should not be used as a reference for its content.

1. **True or False**

   a) Final exam for this course is on April 16th, 4:00pm at PAC.

   b) The root of compressed trie always tests first bit.

   c) Linear probing requires two hash functions.

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

   e) In Cuckoo Hashing, two tables can be different size

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

   g) Suppose we have a hash table of size 6 and our keys are selected uniformly at random from $A = \{1, 2, 3, \cdots, 600\}$. Then, in hashing with chaining, $h_1(k) = k \mod 6$ is better than $h_2(k) = 2k \mod 6$.

   h) 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. **PartialSum**

   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.

4. **Hashing**

   Suppose a set of n keys, S, that are mapped to a hash table of size n using chaining. Also, consider the uniform hashing assumption. If we let c_n be the expected number of slots, prove that $\lim_{n\to\infty} \frac{c_n}{n} = \frac{1}{e}$. You can use the fact that $\lim_{n\to\infty}(1 - \frac{1}{n})^n = \frac{1}{e}$. Try to used indicator variables $I_0, \cdots, I_{n-1}$ that take values of 0 or 1, depending on whether the corresponding entry in the table is empty or not. Then, find the expected value of each I_i

5. **Max in AVL Tree**

   Suppose we want to implement operations $deleteMax()$ and $findMax()$ for AVL tree with additional of $O(1)$ space allowed. Describe an algorithm/implementation of $deleteMax()$ and $findMax()$ which has runtime of $O(\log n)$ and $O(1)$ respectively. In addition, describe how $insert()$ and $delete()$ should be modified, while maintaining $O(\log n)$ runtime.

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.

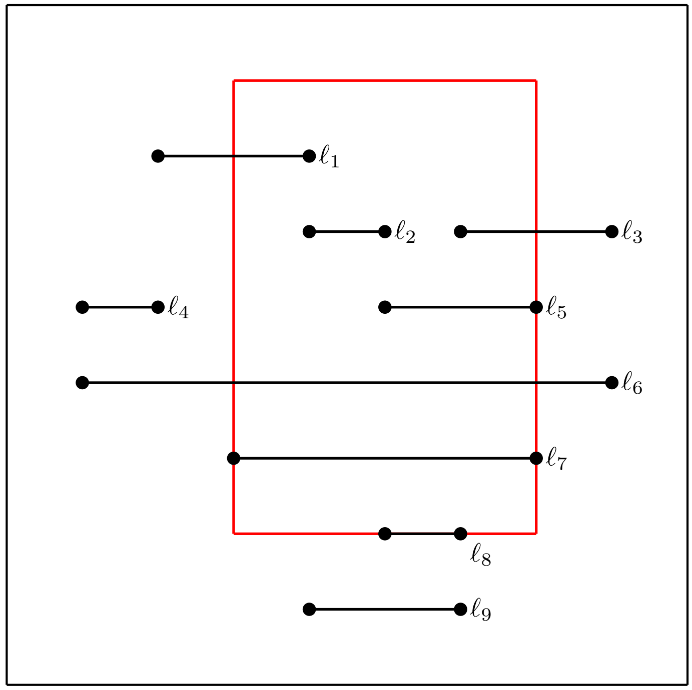
Design a data structure for A with space complexity $O(n)$ to answer queries of the form $MaxDiff(i, j)$ in $O(\log n)$ time. There are no constraints on the runtime for preprocessing the array into the data structure.

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. **Suffix Tree**
Taebin has discovered a secret message in the form of suffix tree, S, indicating a location of hidden treasure. Describe an algorithm that recovers original text T from suffix tree S with $O(n)$ time while using $O(n)$ auxiliary space.

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. **KMP Pattern Match**
Consider the problem of searching for the pattern $P =$ `OMNOMNOM` in the text $T =$ `OMNOONOMNEMOMNOMNOM` with the alphabet $\Sigma = \{O, M, E, N\}$. Construct the failure array and the KMP automaton for P. Then use KMP to search for P in T, showing each character comparison. Indicate characters that are known to match due to the failure array using brackets.

12. **Lempel-Ziv-Welch**
A secret message was reliably intercepted as follows (spaced for convenience):

00011001 00100001 01000000 00101000 00100001 00011000 00100001 01000000 00001011 00001011

Intelligence from other sources indicated that this message was generated by LZW encoding with 8-bit codewords, where the initial dictionary has size 64, though the initial dictionary itself is unknown.

Agent Matthew does not know the original text or the initial dictionary, but he immediately realized that their intelligence was false. Explain how Jason was able to determine this.

13. **Huffman Compression**

a) The following message was compressed using Huffman encoding and transmitted together with its dictionary:

0010000111010101110001011010010

Char	(space)	: (colon)	d	ℓ	p	s	u	w
Code	100	1011	1010	010	001	000	11	011

Decompress the string using the dictionary and write the final message.

b) Agent Taebin doesn't know the information in the message beforehand, but upon seeing the decoded string, he immediately realizes that the message has been tampered with. Explain how Karen determined this.

14. **Simple Burrows-Wheeler Transforms**
Consider Burrows-Wheeler Transforms:

a) Encode the following string using BWT: `MISSISSIPPI`

b) Decode the following string using the inverse BWT: `AIMOEOOPN$TOA`

15. **Burrows-Wheeler Transform**
For the following questions, assume $k > 0$ and $\ell > 0$.

a) Consider the string S of the form $\texttt{(AH)}^k\texttt{A\$}$ (e.g., `AHAHA$` for $k = 2$). What is the Burrows-Wheeler Transform for this string?

b) Consider the string S of the form $\texttt{H}^k\texttt{A}^\ell\texttt{\$}$ (e.g., `HHHAAAA$` for $k = 3$, $\ell = 4$). What is the Burrows-Wheeler Transform for this string?

c) Consider the string S of the form $\texttt{A}^k\texttt{H}^\ell\texttt{\$}$ (e.g., `AAAAAHH$` for $k = 5$, $\ell = 2$). What is the Burrows-Wheeler Transform for this string?