CS 240: Data Structures and Data Management **Winter 2021**

Final Practice Problems

Reminder: Final Exam is between April 19, 9 am and April 20, 9 am.

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/False

For each statement below, write true or false.

a) Open addressing hashing that uses linear probing will require two hash functions.

b) Suffix trees for pattern matching require preprocessing the pattern.

c) Run-length encoding may result in text expansion on some strings.

d) When doing range search on a quadtree, if there is no point within the range specified, the worst case runtime complexity is in $\Theta(h)$.

e) The best-case expected runtime for Karp-Rabin pattern matching is in $O(m)$.

f) Every string consisting of English characters and one end-character can be decoded with the inverse BWT.

g) When using KMP to search for the pattern $\texttt{a}^{m-1}\texttt{b}$ in the text $\texttt{a}^{n}$, the positions of the pattern shifts are the same as the brute-force algorithm.

h) If $\alpha = 1$ for hashing with chaining, inserting a new key will always fail.

i) Every AVL Tree is also a 2-4 Tree.

j) Move-to-front transformation utilizes an adaptive dictionary.

2 Multiple Choice

Pick the best answer for each question.

a) Which of the following functions $f(i)$ would cause interpolation search to have the least worst-case runtime on an array A with $A[i] = f(i)$?

i) $f(i) = \log(i)$
ii) $f(i) = i$
iii) $f(i) = i^2$
iv) $f(i) = 2^i$

b) Given $h_0(k) = k \bmod 7$ in a hash table of size 7, which of the following hash functions would be most suitable for h_1 in double hashing?

i) $h_2(k) = k^2 \bmod 7$

ii) $h_2(k) = (k \bmod 6) + 1$

iii) $h_2(k) = 2 \cdot (k \bmod 4)$

iv) $h_2(k) = \left\lfloor \frac{1}{2} \cdot (k \bmod 13) \right\rfloor$

c) Given $h_0(k) = k \bmod 7$ with two hash tables, each of size 7, which of the following hash functions would be most suitable for h_1 in cuckoo hashing?

i) $h_2(k) = k^2 \bmod 7$

ii) $h_2(k) = (k \bmod 6) + 1$

iii) $h_2(k) = 2 \cdot (k \bmod 4)$

iv) $h_2(k) = \left\lfloor \frac{1}{2} \cdot (k \bmod 13) \right\rfloor$

d) If the root of a quadtree represents the region $[0, 128) \times [0, 128)$ while the deepest (lowest) internal node represents the region $[88, 92) \times [24, 28)$, what is the height of the quadtree?

i) 4

ii) 5

iii) 6

iv) 7

e) Which one of the following statements about compressed tries is false?

i) Every internal node stores an index indicating the bit position to be tested on a search.

ii) The root of the compressed trie always tests the first bit.

iii) A compressed trie that stores n keys always contains less than n internal nodes.

iv) The height of a compressed trie never exceeds the length of the longest string it stores.

f) Given that P is not in T, and $m = \sqrt{n}$, which of the following pattern matching algorithms would have the lowest best-case runtime for searching (excluding preprocessing) for P in T?

i) Rabin-Karp

ii) Knuth-Morris-Pratt

iii) Boyer-Moore Bad Character

iv) Suffix Array

g) CS240 is a course about

i) Data structures and algorithms

ii) Unreasonable time management

iii) Reconsidering academic/life choices

iv) All of the above

3 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.

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

4 Huffman Compression

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

00100001110101011100010110100 10

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 Kang 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 Frank determined this.

5 Boyer-Moore

Boyer-Moore can be modified in many ways. For each of the modifications listed below, state whether or not the modification is valid, i.e., the modified Boyer-Moore will always successfully find the first occurrence of P in T, if P appears in T, or return FAIL if P is not in T. If the answer is "Yes", provide a brief explanation of why it is still valid. If the answer is "No", demonstrate a counterexample, i.e., trace the algorithm on a specific P and T of your choice where the result is incorrect.

Note: We are using Boyer-Moore with only the Bad Character heuristic here.

a) Using a first-occurrence function (denoting the index of the first occurrence of the argument character) instead of a last-occurrence function.

b) When checking a pattern shift, compare characters from the start of the pattern and move forward, instead of scanning backwards from the end of the pattern.

c) Use the last-occurrence function for $P[0 \ldots m-2]$, i.e., P with its last character removed, instead of the last-occurrence function for P.

6 KD-Trees

Consider the following set of 3-dimensional points:

$$(80, 3, 44), (52, 70, 8), (70, 96, 12), (94, 20, 15), (65, 98, 54),$$
$$(41, 26, 58), (28, 84, 91), (63, 32, 99), (36, 87, 72), (39, 90, 40).$$

a) Draw the kd-tree corresponding to these points.

b) Draw the subset of the tree that is visited during a range query in the rectangular box $[60, 70] \times [90, 100] \times [50, 90]$.

7 Suffix Arrays

Mark is trying to acquire some secret text T, consisting of English letters only, but all he was able to obtain was a randomly scrambled text T' (which is a permutation of T), and the suffix array A^S of the original T. Design an algorithm to allow Mark utilize the T' and A^S to recover the original text T in $O(n)$ time.

8 Move-to-Front + Run-Length Encoding

Consider an encoding algorithm that utilizes the following fixed dictionary, where the alphabet consists of letters from A to P:

Char	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The steps of the encoding algorithm are:

- Encode each character with the dictionary above using 4-bit codewords, while also applying Move-To-Front.
- Encoding the resulting string with Run-Length-Encoding.

a) Decode the string `1000101100110011`, which was encoded using the algorithm described.

b) For each $n > 1$, give an example of a valid string whose encoding has the minimum number of bits over all strings of length n.

c) For each $n > 1$, give an example of a valid string whose encoding has the maximum number of bits over all strings of length n.

9 Range Trees

Consider a range tree storing n points, where n is of the form $2^h - 1$, such that the primary tree and all associated trees are perfect, i.e., all levels are full. Given an exact expression for the total number of nodes in the primary trees and all associated trees combined, in terms of n.

10 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.

11 Lempel-Ziv-Welch

Sajed proposes a modification to LZW to expand the dictionary faster: at every step, the encoder adds two new dictionary entries instead of one, when possible; one entry corresponds to the current string being encoded + the next character (like the usual LZW), while the other entry corresponds to the current string being encoded + the next two characters.

For example, if the text is `APPLE`, then after encoding `A` (as 65), the encoder adds two entries to the dictionary: `AP` at 128, and `APP` at 129. Note that after encoding `L` (as 76), the encoder only adds one new entry, `LE` at 134, and that no entries are added after encoding `E` (as 69).

a) Encode the following string with the modified LZW: `BAN_ANANAS_AND_BANANAS`.

b) Modify the LZW decoding algorithm to decode strings that were coded with this modified LZW.

c) Decode the following string that was encoded with the modified LZW:

$$82 - 79 - 84 - 79 - 95 - 77 - 69 - 138 - 78 - 133 - 145 - 128 - 143$$

12 Burrows-Wheeler Transform

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

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

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

13 B-Trees

Consider the following B-Tree, of order 5:

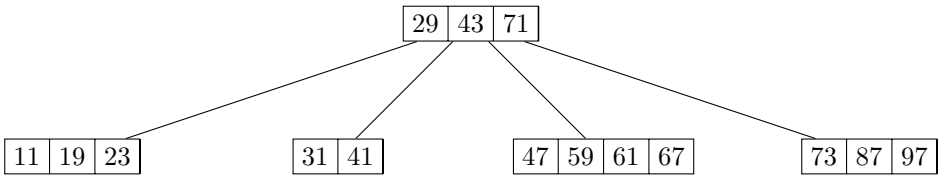

Figure 1: B-Tree of order 5

a) Insert the following keys into the B-Tree, in the order given: 13, 53, 17. Show the tree after each insertion.

b) Delete the following keys from the original B-Tree, in the order given: 19, 43, 31, 29. When deciding between successor/predecessor, choose the successor. When deciding between left or right sibling for transfer/merge, select the right sibling. Show the tree after each deletion.

14 Maximal Difference

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:

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

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.