CS240E	**Final review session**	Apr. 17 (W23)

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. Also, these problems are not organized by difficulty, but by the order in which the relevant concepts were taught.

1. **True/false.**

   For each statement, write true or false.

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

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

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

   (d) Suffix trees for pattern matching require preprocessing the pattern.

   (e) Inserting a set of keys into an empty compressed trie will always result in the same final trie regardless of the insertion order.

   (f) The runtime complexity of range query for kd-trees depends on the spread factor of points.

   Recall: the spread factor is the ratio of the side length of the minimum bounding box, whose bottom-left corner is at $(0, 0)$, to the minimum distance between the points. We assume the points have non-negative coordinates.

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

   (h) Rehashing may be required in Cuckoo Hashing even if the load factor is at an acceptable value.

   (i) Adaptive (rather than static) dictionaries use move-to-front.

2. **Multiple choice.**

   Pick the one 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 $h_0(k) = k \mod 7$ with two hash tables, each of size 7, which of the following hash functions would be most suitable for h_1 in double hashing?

   (i) $h_1(k) = k^2 \mod 7$
   (ii) $h_1(k) = (k \mod 6) + 1$
   (iii) $h_1(k) = 2 \cdot (k \mod 4)$
   (iv) $h_1(k) = \lfloor \frac{1}{2} \cdot (k \mod 13) \rfloor$

(c) 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

(d) Which of the following statements about compressed tries is false?

   (i) every internal node stores an index indicating the 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

(e) Which of the following search operations on a non-dictionary structure has the most efficient worst-case runtime?

   (i) searching for a specific key in a max-heap
   (ii) searching for a specific point in a kd-tree with points in general position
   (iii) searching for any occurrence of a specific character in a text using a suffix tree, with children pointers stored as arrays
   (iv) searching for a specific character in a decoding trie of characters (like Huffman's trie)

3. **Hashing.**

Let $p \geq 3$ be prime, and consider the universe of keys $U = \{0, 1, \ldots, p^2 - 1\}$. Answer each question for an initially empty hash table of size p.

(a) Using double hashing with $h_1(k) = k \mod p$ and $h_2(k) = \lfloor k/p \rfloor + 1$, give a sequence of two keys to be inserted that results in failure.

(b) Using cuckoo hashing with $h_1(k) = k \mod p$ and $h_2(k) = k \mod (p - 1) + 1$, give a sequence of three keys to be inserted that results in failure.

(c) Using cuckoo hashing with $h_1(k) = k \mod p$ and $h_2(k) = \lfloor k/p \rfloor$, give a sequence of three keys to be inserted that results in failure.

4. **Boyer-Moore.**

   Boyer-Moore can be modified in many ways. For each of the modifications listed below, state whether 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 counter-example, i.e. trace the algorithm on specific P and T of your choice where the result is incorrect.

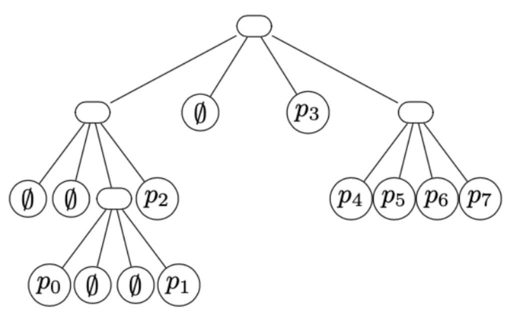
   (a) Using a first-occurrence function (denoting the index of the first occurrence of the argument character) instead of a last-occurrence function.

   (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 $P[0..m-1]$, i.e. P with its last character removed, instead of the last-occurrence function for P.

5. **Quad trees.**

   (a) Create a set of 8 distinct points for which all coordinates are integers in the range $[0, 8)$ and that has the following quad tree:



   (b) Given a quad-tree T, what is the smallest integer k such that there exists a set of distinct points whose quad-tree is T and whose coordinates are integers in the range $[0, 2^k)$?

6. **Range queries.**

   Consider the following set of points in $[0, 16]^2$:

$$p_0\colon (3,5),\ p_1\colon (7,8),\ p_2\colon (6,2),\ p_3\colon (8,0),\ p_4\colon (0,3),$$
$$p_5\colon (4,6),\ p_6\colon (2,9),\ p_7\colon (9,1).$$

(a) Show the corresponding quad-tree.

(b) Show the corresponding kd-tree.

(c) Show one possible range tree. The primary tree should be perfectly balanced.

7. **Pattern matching.**

Consider the pattern $P = 0110101$ and the text T listed in the following table.

0	0	1	1	1	1	0	0	1	1	1

(a) Indicate all the checks that were done by the brute-force method.

(b) Consider the Karp-Rabin fingerprint that simply counts the number of 1s in the bit-string. Is this a rolling hash-function? And using these fingerprints, how many checks were done during Karp-Rabin pattern matching?

(c) Compute the KMP failure-function for P.

(d) Show the KMP automaton for P.

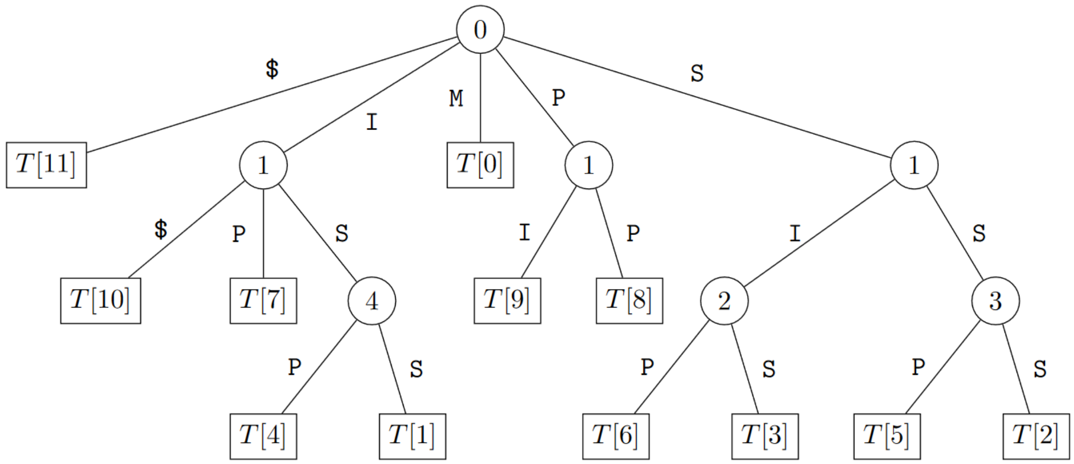
(e) Consider now the pattern $P =$`fiddledidi`. Show the Boyer-Moore last-occurrence array.

8. **Suffix trees.**

Jason discovered a secret message in the form of a suffix tree S, indicating the location of a hidden treasure.

(a) Design an algorithm that recovers the original text T from its corresponding suffix tree S. The algorithm should run in $O(n)$ time while using $O(n)$ auxiliary space.

(b) Determine the original text for the following suffix tree:



9. **Move-to-front and 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.
   - Encode the resulting string with RLE.

   (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.

10. **Consecutive strings in a trie.**

    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 $\{000, 01, 0110, 101, 11\}$. Then:

- *consecutive*$(0110, 101)$ returns true
- *consecutive*$(01, 000)$ returns true
- *consecutive*$(11, 000)$ returns false

11. **Burrows-Wheeler Transform.**

    (a) Encode the following string using BWT: `TORONTO`

    (b) Decode the following string using the inverse BWT: `IPSSM$PISSII`