CS 240: Data Structures and Data Management **Spring 2022**

Final Help Session - Practice Problems

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

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

e) 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.

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

g) 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).

h) A modified version of the Boyer-Moore algorithm that compares characters from the start of the pattern and moving forward (instead of starting from the end of the pattern and moving backward) when checking a pattern shift will always successfully find the first occurrence of a pattern P in a text T (if it appears in T).

i) 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.

j) A modified version of a kd-tree, where every internal node is split by the x-coordinate, will maintain $O(\log n)$ height and $O(\sqrt{n} + s)$ time.

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

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

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) Learning beauty tips from the ~~style icons~~ ISAs

v) All of the above

3 Running Time

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 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.

5 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 Esfajamshidpetrick 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 Arvidark determined this.

6 Karp-Rabin

For Karp-Rabin pattern matching, consider the following hash function for strings over the alphabet $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$:

$$\begin{aligned} h(P) =& (\#\text{ of occurrences of } \texttt{A}) + 2 \times (\#\text{ of occurrences of } \texttt{C}) \\ &+ 3 \times (\#\text{ of occurrences of } \texttt{G}) + 4 \times (\#\text{ of occurrences of } \texttt{T}) \end{aligned}$$

Given the pattern $P = \texttt{TAGCAT}$ and sequence $T = \texttt{TGCCGATGTAGCTAGCAT}$, use the table below to show all the character comparisons performed during Karp-Rabin pattern matching. Start a new pattern shift (in which character comparison occurs) in a new row. You may not need all the available space.

T	G	C	C	G	A	T	G	T	A	G	C	T	A	G	C	A	T

Table 1: Table for Karp-Rabin problem.

7 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.

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

9 Encoding Schemes

Suppose you need to design an encoding algorithm for source texts of length n over the alphabet $\{0, 1, 2, 3\}$, given that all source texts begin with an even digit and alternate between even and odd digits. There are no constraints on the alphabet of the coded text, but the encoding must be lossless.

a) Design an encoding scheme (i.e. both an encoding algorithm and a decoding algorithm) that achieves the lowest best-case compression ratio.

b) Design an encoding scheme that achieves the lowest worst-case compression ratio.

c) Prove that the worst-case compression ratio in part (b) is optimal, i.e. there is no encoding scheme that achieves a lower worst-case compression ratio.

10 Suffix Trees

Zahra 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:

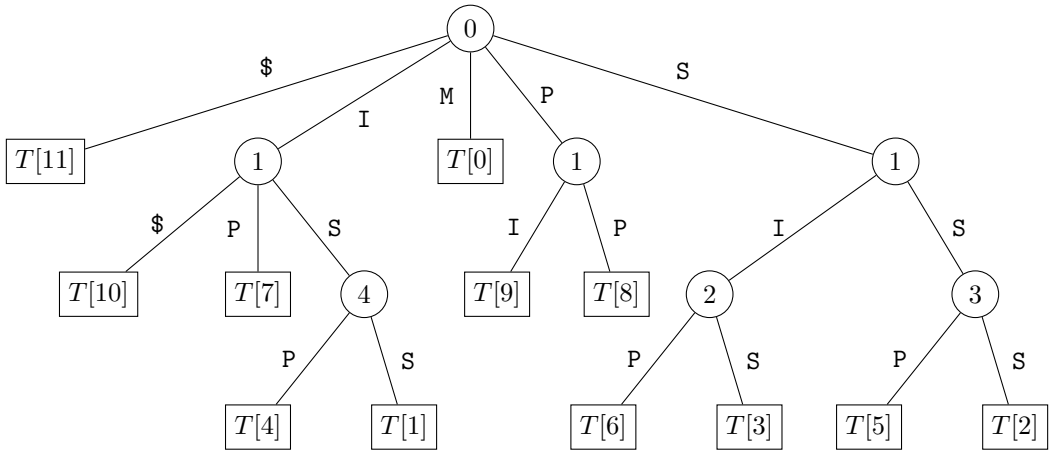

Figure 1: Mysterious Suffix Tree

11 Lexicographically Least Permutation

The lexicographically least permutation of a string is the permutation for which the characters are in non-decreasing order. For example, the lexicographically least permutation of `COUNTSORT$` is `$CNOORSTTU`.

For the following scenarios, S is a string of length n whose characters are from an ordered alphabet Σ, but the order of the characters is not known. Characters can be compared for equality, but not relative order.

a) Let $h_1 \colon \Sigma \to \mathbb{N}$ be a hash function such that $h_1(c_1) < h_1(c_2)$ if and only if $c_1 < c_2$. Given S and h_1, design an algorithm that generates the lexicographically least permutation of S in $O(n \log n)$ time.

b) Let $h_2 \colon \Sigma \to \mathbb{N}$ be a hash function such that $h_2(c)$ maps to the sorted position of c in Σ, i.e., the smallest character in Σ maps to 0, the second smallest to 1, and so on. Given S and h_2, design an algorithm that generates the lexicographically least permutation of S in $O(n + |\Sigma|)$ time.

c) Suppose S contains n distinct characters. Let C be the Burrows-Wheeler Transform of $S{\cdot}\$$, i.e., S with `$` concatenated at the end, where `$` is an end-character which is not in Σ but is smaller than everything in Σ. Given S, C, and `$`, design an algorithm that generates the lexicographically least permutation of S in $O(n^2)$ time.

   For example, given $S = \heartsuit\clubsuit\spadesuit\diamondsuit$ and $C = \diamondsuit\heartsuit\spadesuit\$\clubsuit$, the lexicographically least permutation of S is $\clubsuit\diamondsuit\heartsuit\spadesuit$.

12 String Decoding

The following bit-string C was generated by 3 steps of encoding: BWT, MTF, RLE.

$$C = 00100100011011111000100001111110001111101001001011010110011 1$$

a) The final step of encoding was applying RLE to encode C' to C. Use run-length decoding to recover C'.

b) The middle step of encoding was applying MTF to encode S' to C', using the following initial dictionary:

Character	$	_	...	A	L	M	N	O	P	...	Z
Code (DEC)	0	1	...	6	17	18	19	20	21	...	31

The codewords here are shown in decimal, but are each represented using 5 bits of binary. For example, the decimal codeword 7 would be represented as 00111. This ordering of the characters is also the sorted ordering.

Decode the C' from part (a) using MTF to get $S' =$ `AAPP_OOOOL$MM`, and show the final dictionary.

c) The first step of encoding was applying BWT to encode S to S'. Apply the inverse BWT on S' to recover the original text S.

13 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.

14 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.

15 Lempel-Ziv-Welch

a) Suppose a text of length n is encoded with LZW into a sequence of w codewords, Assume that the number of bits to represent each codeword is large enough such that the dictionary never becomes full (i.e., a new substring is always added to the dictionary after every step). Give a $\Theta()$ bound in terms of n to denote the smallest value of w.

b) 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 Duan does not know the original text or the initial dictionary, but he immediately realized that their intelligence was false. Explain how Yundi was able to determine this.

Final Review Questions

Bilal Khan
CS240

July 25, 2022

Question 1.

(a) **False**, by definition.

(b) **False**, by definition.

(c) **True**

(d) **True**

(e) **True**

(f) **False**, it should just add an element to the end of the linkedlist ???

(g) **False**, e.g. $T = feedapaper$ and $P = paper$ would shift the pattern too far and miss the occurence of P.

(h) **False**, e.g. $T = abcd$ and $P = bcd$ would see a mismatch at idx 0 and shift the pattern forwards three spots and miss the occurence of P.

(i) **True**, technically should be leq this since its not *explicitly* an upper bount but unless there's some off-by-one thing im missing ??

(j) **False**

Question 2.

(a) ii) unless im missing smthg

(b) ii) neq 0 and relatively prime with 7 (?)

(c) iv) ? it maps keys to the most spots in the hashmap

(d) ii) 5. $0-128, 0-128-> 64-128, 0-64-> 64-96, 0-32-> 80-96, 16-32-> 88-96, 24-32-> 88-92, 24-28$

(e) ii)

(f) iv) suffix array will take $log(n)$ time to search and find no match in best case where pattern differs from all suffixes in first spot. boyer-moore will take $\sqrt(n)$ time in best case where first char checked in pattern is not in text and pattern moved $\sqrt(n)$ spots.

(g) iii) :')

Question 3.

(a) not sure how much they want you to formalize this? False, not a tight bound around n^9. worst case is lower bounded by n^8.

(b) False, not strictly lower bounded, can use limit rule iirc to show that limit of ratios isn't ∞.

(c) True. unless there's something with non-integer powers in runtime analysis that messes with this?

(d) False, idk why this is a question?

Question 4.

(a) inserting key $p-1$ and then inserting key p^2-1. $p-1 \bmod p = p-1$ and so we insert it into slot $p-1$. $p^2-1 \bmod p = p-1$ and so we have a collision. using double hashing to try and resolve the collision: our new slot becomes: $((p^2-1 \bmod p)+1 \cdot (\lfloor p^2-1/p \rfloor + 1)) \bmod p = ((p-1)+1 \cdot (p-1+1)) \bmod p = (p-1+p-1+1) \bmod p = 2p-1 \bmod p = p-1$ which is still a collision.

(b) TODO

Question 5.

(a) (001)(000)(011)(1010)(1011)(100)(010)(11)(010)(010) = "pswd: lull"

(b) the encoding of u uses fewer bits than the encoding of any other character but appears only once and less often than any other character. e.g. l is repeated three times in the message and has a 3-bit encoding instead of being shorter than the encoding of u. Huffman automatically allocates shorter bit encodings to more frequently used characters.

Question 6. $T = TGCCGATGTAGCTAGCAT$
$P = TAGCAT$
$h(P) = 2 + 2 * 1 + 3 * 1 + 4 * 2 = 15$

T	G	C	C	G	A	T	G	T	A	G	C	T	A	G	C	A	T
T	G	C	C	G	A 15												
	G	C	C	G	A	T 15											
		C	C	G	A	T	G 15										
			C	G	A	T	G	T 17									
				G	A	T	G	T	A 16								
					A	T	G	T	A	G 16							
						T	G	T	A	G	C 17						
							G	T	A	G	C	T 17					
								T	A	G	C	T	A 15				
									A	G	C	T	A	G 14			
										G	C	T	A	G	C 15		
											C	T	A	G	C	A 13	
												T	A	G	C	A	T 15

Question 7. Consecutive strings will have a common prefix and path in the compressed trie until you reach a vertex at which the values of corresponding indices in the binary string differ. At that point, two binary strings are consecutive iff they are the strings stored in the left-most path down from the right child of that vertex and the right-most path down from the left child of that vertex. We give the following algorithm:

Follow a path down from the root of the trie for the common prefix of b_1 and b_2 until you reach a vertex where one (or both) string(s) end(s) or both strings differ at that index. For any string b_1, b_2 that hasn't ended yet, continue to follow the corresponding left/right-most path down from the left/right child of that vertex until you reach the leaf node that matches the binary string we're looking for, and return successfully, or until you reach a vertex where the value of the edge of the left/right-most path differs from the string we're looking for and return unsuccessfully.

Question 8.

(a) $\$(AH)^kA \rightarrow A\$(H)^k(A)^k$

(b) $\$H^kA^l \rightarrow A^lH^k\$$

(c) $\$A^kH^l \rightarrow H\$(A)^{k-1}(H)^{l-1}A$

Question 9.

(a) best case we have long runs of identical characters. use a modified version of rle that uses a 0 or 1 at the beginning of the run to represent which even/odd digit runs are switching to.

(b) LZW probably. relies less on long runs/distribution of the characters than other algorithms?

(c) idk, unless they want us to use a custom compression algorithm.

Question 10.

(a) visit leaf nodes in an in-order traversal. suffix start idx at that leaf node + idx stored in any parent node of that = idx in the string that corresponds to the value of the edge from that parent node. not O(n) runtime, but yeah there's a similar one that is O(N).

(b) $MISSISSIPPI\$$

Question 11.

(a) merge sort, but use the hash fn? assuming hash fn takes constant time?

(b) go over each char in string (O(N)), and for each char add 1 to an array of counts for each char of size $|\Sigma|$. then, go over the count of each char and output that many chars of that char (O($|\Sigma|$)).

(c) decode BWT string C to get original string O(N) time. then, since we dont know anything about the ordering of the characters in the alphabet (i *think* this is what the question is stating) we cant just sort the string. but we know that the first characters in the cyclically shifted matrix are in lexographic order. we also know the last characters in the cyclically shifted matrix are the characters in C. Go over each char in C, and for each char, go through S (keeping in mind to append \$ to S b/c EOS symbol) and find the character after that char in S wrapping around as necessary (if a char is at the end of a permutation, then the char at the beginning of the permutation is the one that comes right after it in S since thats how we chop off prefixes/suffixes for cyclic shifting) and append that char to the output string.

Question 12.

(a) $C = 00100100011011111000100001111110001111101001001011010110011 1$

$C = (0)(010)(010)(00110)(1)(1)(1)(1)(1)(0001000)(011)(1)(1)(1)(1)(0001111)(1)(010)$
$(010)(010)(1)(1)(010)(1)(1)(00111)$

$C = (10)(10)(110)(1)(1)(1)(1)(1)(1000)(11)(1)(1)(1)(1)(1111)(1)(10)(10)(10)(1)(1)(10)(1)(1)(111)$

$C = (2)(2)(6)(1)(1)(1)(1)(1)(8)(3)(1)(1)(1)(1)(15)(1)(2)(2)(2)(1)(1)(2)(1)(1)(7)$

$C = 00110000001010100000000111010100000000000000010011001011010000000$

$C = (00110)(00000)(10101)(00000)(00011)(10101)(00000)(00000)(00000)(10011)(00101)$
$(10100)(00000)$

$C = (6)(0)(21)(0)(3)(21)(0)(0)(0)(19)(5)(20)(0)$

(b)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
m	\$	l	o	_	p	a					b	c	d	e	f	g	h	i	j	k	n	q	r	s	t	u	v	w	x	y	z

$AAPP\_OOOOL\$MM$

(c)

0	1	2	3	4	5	6	7	8	9	10	11	12
A, 0	A, 1	P, 2	P, 3	_, 4	O, 5	O, 6	O, 7	O, 8	L, 9	\$, 10	M, 11	M, 12

0	1	2	3	4	5	6	7	8	9	10	11	12
\$, 10	_, 4	A, 0	A, 1	L, 9	M, 11	M, 12	O, 5	O, 6	O, 7	O, 8	P, 2	P, 3

$T = OOMPA\_LOOMPA\$$

Question 13. basically like range search but each non-root node keeps track of the min/max of its children. seach through like in range search, checking boundary nodes etc, and then return the max - min of the largest max and smallest min in range. needs two traversals down left/right subtrees and so is in log n.

Question 14.

(a) $C = 1000101100110011$

$C = [1](0001011)(00110)(011)$

$C = [1](1011)(110)(11)$

$C = [1](11)(6)(3)$

$C = (11111111111)(000000)(111)$

$C = (1111)(1111)(1110)(0000)(0111)$

$C = (15)(15)(14)(0)(7)$

$T = POMME$

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

(b) The string of n "A"s. $A \to 0 \to 000$ in ASCII the RLE of which is $(0)[\lfloor \log(n) \rfloor \#zeroes]$ [binary representation of n].

(c) the string of $PPOONNMMLL \cdots AA$ repeated an arbitrary number of times. P initially gets mapped to $15 = 1111$ and on the second try gets mapped to $0 = 0000$ and so on for all the letters. the RLE then swaps between zeros and ones b/c move to front a maximal n number of times. each char is either 1111 or 0000 and so on. each run is of length 4 and so the rle of each run is 00100 and we have n of these and an additional 1 at the beginning of the encoded text to indicate that we start from a run with a leading one.

Question 15.

(a) assuming this q means whats the best case space usage? best case is A^n which will take $\Theta(\sqrt{n})$ space.

(b) $C = (25)(33)(64)(40)(33)(24)(33)(64)(11)(11)$

The sequence of two code-numbers 33 and 64 appears as the second and third code numbers in the encoded text, the concatenation of the strings they represent should be added to the dictionary as code-number 65. Later on in the sequence, the sequence of 33 and 64 appears again, so the dictionary should have used the stored code-number 65 that represents the longer string.