Final Review Questions

Bilal Khan
CS240

July 29, 2022

Also see: https://github.com/y87feng/CS240-fall-2017
S22 review qs

Question 1.

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

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

(c) **False**, $O(nh)$.

(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) **True**, e.g. bad character heuristic won't mess anything up since we take the min(j, L[c]) and we increase j from 0 - len pattern so we won't use the last char heuristic unless the last occurence of a char in a string is left of the mismatch, which is impossible for all indices.

(i) **False**, the given bound is an exact number, not an O or theta bound. range trees have at least 2n nodes b/c internal nodes.

(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? True, worst case is upper bounded by n^9.

(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) True.

(e) False, theta i is in 5-9, not in o(1).

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) 0, $p-1$, p^2-1, p^2-p get mapped all to 0 for h_1 and to either 0 or $p-1$ for h_0. 3 unique hashing locations for 4 keys is a conflict from pigenhole principle.

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

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

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

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) identity or some standard method like huffman maybe?

(c) uhh

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) can you just shift C over ? **probably wrong** 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.

Question 2017 q1.

True/False Problems

For each statement below, write true or false. Justify five of them.

(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 quad tree, if there is no point within the range specified, the worst case runtime complexity is $\Theta(1)$.

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

(e) Deleting any element stored at the root of a 2-3 tree always decreases the height of the tree by 1.

(f) If the bubble-up version of $heapify$ is used in Heapsort, then the worst case runtime of Heapsort will be $\Omega(n^2)$.

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

(h) A valid way to hash strings is to flatten the string into an integer using the ascii values of each letter and some radix base R.

(i) If an AVL tree node has balance 2 and its right child has balance 1, then a double left rotation is required.

(j) Move-to-front compression uses adaptive instead of fixed dictionaries.

(a) False

(b) True

(c) False

(d) False

(e) N/A

(f) False? heap sort is always nlogn?

(g) False

(h) True, valid but not optimal

(i) False, single left rotation iirc?

(j) True.

Question 2017.

Multiple Choice

Pick the best answer for each question.

1. The last occurrence function for the pattern `MELSMEMES` would contain the following values for each character:

   (a) E=8, L=3, M=7, S=9

   (b) E=6, L=7, M=8, S=2

   (c) E=2, L=8, M=6, S=7

   (d) E=7, L=2, M=6, S=8

2. A 2-3 tree of height 1 in which every node contains two keys will have _____ NIL leaves:

   (a) 6

   (b) 8

   (c) 9

   (d) 12

3. Using LZW decoding, the last code 132 decodes to what?

   67 128 129 130 131 132

   (a) CCCCCC

   (b) CCCCCCC

   (c) CCCCCCCC

   (d) CCCCCCCCC

4. What one of these statements about hashing is false?

   (a) Constant time search and delete if Cuckoo Hashing is used

   (b) Hash tables may use more space than the number of elements

   (c) Two keys will never hash to the same index using chaining

   (d) Insert for Cuckoo Hashing can result in a loop

5. Suppose we have an array of n numbers where each number is no larger than n^3, and assume n is a perfect square. Consider running Heapsort, Quicksort, and Radix sort with radix base $R = \sqrt{n}$ on this array. The worst case asymptotic runtimes of each sorting algorithm, from best to worst, is:

   (a) Heapsort, Quicksort, Radix sort

   (b) Radix sort, Heapsort, Quicksort

   (c) Quicksort, Radix sort, Heapsort

   (d) Radix sort, Quicksort, Heapsort

4. A quadtree with bounding box $[0, 8] \times [0, 8]$ over the following points has a height of _____.

$$(6, 2), (0, 1), (3, 4), (7, 5), (1, 0)$$

   a) 2
   b) 3
   c) 4
   d) 5

7. If a length-m pattern does not appear in a length-n text, the following string-matching algorithms still need to read at least $n - m$ characters of the text in the best-case, except for:

   a) DFA
   b) KMP
   c) Boyer-Moore
   d) Rabin-Karp

1. d
2. n/a
3. a
4. c
5. b
6. c
7. c

Knuth-Morris Pratt

(a) Compute the failure array for the pattern $P =$`JWJWJX`.

4

J	W	X	J	J	W	J	J	W	J	W	J	W	J	X	J	W	X	J	J

Table 1: Table for KMP problem.

(b) Show how to search for pattern $P =$`JWJWJX` using the KMP algorithm for the text in the table above. Place each character of P in the column of the compared-to character of T. Put brackets around the character if an actual comparison was not performed. You may not need all the available space.

(a) 001230

(b)

J	W	X	J	J	W	J	J	W	J	W	J	W	J	X	J	W	X	J	J
J	W	J	W	J	X														
		J	W	J	W	J	X												
			J	W	J	W	J	X											
				J	W	J	W	J	X										
						[J]	W	J	W	J	X								
							J	W	J	W	J	X							
									[J]	[W]	[J]	W	J	X					

Hashing

Using double hashing with the hash functions $h_1(n) = n \bmod 7$ and $h_2(n) = (3n \bmod 6) + 1$ and a table of size 7, answer the questions below:

(a) Fill the table with correctly hashed values such that a call to $search(6)$ succeeds at the end of a probe sequence of length four.

(b) Suppose the numbers written in your table above were inserted using linear probing instead with the hash function $h_2(n)$. Show the resulting table.

(a) insert $13 = 6 mod 7$, 0, 1, 2 into the hash table. $(3(6) mod 6) + 1 = 1$ so it will look in successive spots in the table on each probe.

(b) u can show this but i won't be

Huffman Compression

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

00100001110101011100010110100 10

' '	=	100 (blank space)
:	=	1011
d	=	1010
ℓ	=	010
p	=	001
s	=	000
u	=	11
w	=	011

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

(b) Agent Bond doesn't know the password beforehand, but upon seeing the decoded string, she immediately realizes that the message has been tampered with. Explain how Jane determined this.

Run Length Encoding

(a) Use run length encoding to compress the string shown below:

0100100001000000001111111010111010

(b) State the compression ratio achieved.

(c) Use run length decoding to decompress the string shown below:

1111000111000111000110001100 1001

(a) $pswd : lull$

(b) u has a shorter encoding than l.

(a) $T = 01001000010000000011111110111010$

$T = (0)(1)(00)(1)(0000)(1)(00000000)(1111111)(0)(111)(0)(1)(0)$

$C = [0](1)(1)(2)(1)(4)(1)(8)(7)(1)(3)(1)(1)(1)$

$C = 0(1)(1)(010)(1)(00100)(1)(0001000)(00111)(1)(011)(1)(1)(1)$

$C = 011010100100100010000011111011111$

(b) $32 * log(2)/32 * log(2)$

(c) $C = 11110001110001110001100011001001$

$T = [1](1)(1)(1)(0001110)(00111)(0001100)(011)(00100)(1)$

$T = [1](1)(1)(1)(14)(7)(12)(3)(4)(1)$

$T = 10100000000000000111111000000000000011100001$

Lempel-Ziv-Welch Encoding

Encode the following string using LZW compression:

`DARK_DAN_BARKS_DANK`

Char	Ascii Value
A	65
B	66
D	68
K	75
N	78
R	82
S	83
_	95

Add new entries to the encoding dictionary starting at value 128.

$T = DARK\_DAN\_BARKS\_DANK$
$128 = DA$
$129 = AR$
$130 = RK$
$131 = K\_$

$132 = \_D$
$133 = DAN$
$134 = N\_$
$135 = \_B$
$136 = BA$
$137 = ARK$
$138 = KS$
$139 = S\_$
$140 = \_DA$
$141 = AN$
$142 = NK$
$C = (68)(65)(82)(75)(95)(128)(78)(95)(66)(129)(75)(83)(132)(65)(78)(75)$

Burrows Wheeler Transform

The following key was encoded by the Burrows-Wheeler Transform.

`EPESLPP$ASEAR`

Decrypt it using the method outlined in the slides, showing the array of tuples A, sort(A), and each value of j.

Suffix Trees

Given a string T of length n, suppose we have already constructed the suffix tree for T.

(a) Draw the suffix tree for the string `GCTAGCTAG`.

(b) The longest repeated substring is the longest substring of a string that occurs at least twice. For example, the longest repeated substring of `GCTAGCTAG` is `GCTAG`. Create an algorithm to find the longest repeated substring of T in $O(n)$ time.

$C = EPESLPP\$ASEAR$
$C = (E,0)(P,1)(E,2)(S,4)(L,5)(P,6)(P,7)(\$,8)(A,9)(S,10)(E,11)(A,12)(R,13)$
$C = (\$,8)(A,9)(A,12)(E,0)(E,2)(E,11)(L,5)(P,1)(P,6)(P,7)(R,13)(S,4)(S,10)$
$T = PAPERSPLEASE\$$
(A) nah

(B) find deepest node in suffix tree wrt node idx? suffix stored at that node is the longest suffix?

Question S2021.

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.

(c) $k = 0, h_0(k) = 0, h_1(k) = 0$
$k = 1, h_0(k) = 1, h_1(k) = 0$
$k = p - 1, h_0(k) = p - 1, h_1(k) = 0$
$k = p^2 - 1, h_0(k) = p - 1, h_1(k) = p - 1$
$k = p^2 - p, h_0(k) = 0, h_1(k) = p - 1$
$k = p^2 - p + 1, h_0(k) = 1, h_1(k) = p - 1$
6 keys, 5 slots

6 Compressed Trie Height

Let T be a compressed trie of n strings from an alphabet Σ, where the end-character is not in Σ. Give exact expressions (not order notation) for the following, in terms of n and $|\Sigma|$:

a) What is the maximum height of T?

b) What is the minimum height of T?

(a) n, store n strings of identical chars of all lengths from 1 to n
(b) 1, if all strings differ in their first bit, then the compressed trie only needs to store the root and up to $|\Sigma|$ leaves.

9 Suffix Arrays

Agent Li is trying to acquire some secret text T, consisting of uppercase 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 utilize T' and A^S to recover the original text T in $O(n)$ time.

10 Run-Length Encoding

For the following questions, you may assume that n is divisible by 4.

a) For each $n > 0$, give a string of n bits that achieves the worst compression ratio with Run-Length Encoding from all n-bit strings, and state the exact compression ratio achieved.

b) Same question, but for the best compression ratio. **Hint:** If there are multiple runs in the string, would it help to modify the string by merging two or more runs into a single run?

(9). but radix sort T' in $O(26 * n) \in O(n)$ time, then iterate over sorted T' and copy over each char to its corresponding slot in T based on the offset stored in $A^S[idx]$ also in $O(n)$ time.

(10)

a) Either 0^n or 1^n works. For instance, 1^n compresses to $10^{\lfloor \log n \rfloor}(n)_2$, where $(n)_2$ is the binary representation of n. The compression ratio is $\frac{1+\lfloor \log n \rfloor + \lfloor \log n \rfloor + 1}{n} = \frac{2\lfloor \log n \rfloor + 2}{n}$.

b) Either $(0011)^{n/4}$ or $(1100)^{n/4}$ works. Each run compresses to 010. The compression ratio is $\frac{1+6(n/4)}{n} = 1.5n + 1/n$.