CS310 Data Structures

Final Exam

Total Points: 100 **Number of Questions: 5**

Name: Stephen Giang

REDID: 823184070

Please read the questions thoroughly before answering.

Submit the exam as a PDF (written & scanned or filled in word and exported)

1. **(20 Points) Time Complexities**

   Assume the most time-efficient way to implement the operations listed below, assume no duplicate values and that you can implement the operation as a member function of the class – with access to the underlying data structure, including knowing the number of values currently stored (N).

   • **Then, give the tightest possible upper bound for the worst-case running time for each operation in terms of N, Big-O**.

   For any credit, **you must explain why it gets this worst-case running time (No points will be awarded without explanation, even if Big O is correct)

a. Sorting the elements of an array with Merge Sort

   O(nlogn). Merge Sort works as halving the array until its down to its separated induvial elements. That takes O(logn). Then it stiches all the separated elements back into the array in order by comparing each individual element from the left array to the right array from left to right in each array. This takes O(n). With merge sort, if we choose the pivot to always be the middle, then it always run the same way and always run to O(nlogn), even in the worst-case.

b. Given an input integer from the user, find if the integer is odd / even

   O(1). This would simply take the mod 2 of the integer. If 1, then odd, else its even. Because of only 2 comparisons, it runs to O(1). This will also always run the exact same way so in all cases it will run to O(1), even in the worst-case.

c. Finding an element on a sorted array with Binary Search

   O(logn). Binary Search works by choosing the middle index as a pivot. It then compares the element it is searching for with the pivot. If it is smaller, it will rerun with the left half, or else it will rerun with the right half. Each case, the method halves the array until the element is found or not found. This will always run the exact same way, if we choose the pivot in the middle, so in all cases it will run to O(logn) even in the worst-case.

d. Given a string input from the user (with n characters), find all permutations of the string.

O(n!). The first character of the new string has n choices, and the next has (n-1) characters it can be. To find the permutations, you would have to implement 'n' amount of nested loops that goes over each character's # of choices. Thus it will be n(n-1)...2*1 = n!. Thus giving us O(n!). This will always run the exact same way so in all cases it will run to O(n!) even in the worst-case.

e. Given a set of n numbers from the user, find all subsets of the set.
Example:
- Set is {1,2,3} Subsets are {},{1},{2},{3}, {1,2}, {1,3}, {2,3}
- {1,2} and {2,1} are considered the same.

O(2^n). Because order doesn't matter, the first number will have 2 choices, either be the first number from the original set, or be null. The second number will also have 2 choices, either be the second number from the original set, or be null. To find all subsets, you would have to implement 'n' amount of nested loops that goes over the 2 choices. This will always run the exact same way so in all cases it will run to O(2^n) even in the worst-case.

2. (20 Points) Hashing

Given below is the ASCII Table for characters:

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

You are given the following set of keys to add to the hashtable

a b e d

a b c d

d c b a

s e a x

c d e f

x a e s

Dictionary objects are (K,V) pairs. Here V Value does not matter. K keys is listed as above for the purpose of comparing hashfunctions.

The keys are 4-character Strings.

Draco and Harry have each written a hashfuction to map the above keys into **a hashtable of size 10**.

Draco's Hashfunction:

- Find the **ASCII -> OCT** value of each character from the given table (*pay attention to the case, uppercase and lowercase are different OCT value*)
- Add the remaining values and mod by 10 to get the index
- In case of collision, use **Open Addressing with Linear Probing**. If you reach the end of the table loop back to top.

Example:

Reading OCT values from the given table, all lowercase letters w,x,y,z

w x y z = 167+170+171+172 = 680 mod 10 = 0

Using Draco's function add the keys given at the start of the question to the following table and show the working for each key, *you must enter the values in the table and show the steps for each key in order to get any credit.*

TABLE INDEX	KEY
0	a b c d
1	d c b a
2	a b e d
3	x a e s
4	
5	
6	
7	
8	c d e f
9	s e a x

1) a b e d – 141 + 142 + 145 + 144 => 572 mod 10 = 2

2) a b c d – 141 + 142 + 143 + 144 => 570 mod 10 = 0

3) d c b a – 144 + 143 + 142 + 141 => 570 mod 10 = 0
(goes to Index 1 because 0 is already filled)

4) s e a x – 163 + 145 + 141 + 170 => 619 mod 10 = 9

5) c d e f – 143 + 144 + 145 + 146 => 578 mod 10 = 8

6) x a e s – 170 + 141 + 145 + 163 => 619 mod 10 = 9
(goes to Index 0 because 9 is already filled)
(goes to Index 1 because 0 is already filled)
(goes to Index 2 because 1 is already filled)
(goes to Index 3 because 2 is already filled)

Harry's Function:

- Assign number to each character in the key based on their position in the Alphabet (A is 1, B is 2, C is 3 and so on)
- Concatenate the first two numbers (join numbers do not add) and concatenate the next two numbers (join numbers do not add)
- Add mathematically both concatenated numbers from previous step
- Mod the number resulting from the previous step by 10 to get the index
- In case of collision use **Open Addressing with Quadratic probing** (checking 1,4,9,16th position and so on). If you reach the end of the table, loop back to top.

Example:

w is position 23, x is position 24, y is position 25, z is position 26

w x y z = SUM ("23"+"24" , "25"+"26") = 2324 +2526 = 4850 mod 10 = 0

Using Harry's function add the keys given at the start of the question to the following table and show the working for each key, *you must enter the values in the table and show the steps for each key in order to get any credit.*

TABLE INDEX	KEY
0	c d e f
1	x a e s
2	
3	
4	d c b a
5	
6	a b e d
7	a b c d
8	
9	s e a x

1) a b e d – SUM("1" + "2", "5" + "4") = 12 + 54 = 66 mod 10 = 6

2) a b c d – SUM("1" + "2", "3" + "4") = 12 + 34 = 46 mod 10 = 6
(goes to Index 7 because 6 is already filled)

3) d c b a – SUM("4" + "3", "2" + "1") = 43 + 21 = 64 mod 10 = 4

4) s e a x – SUM("19" + "5", "1" + "24") = 195 + 124 = 319 mod 10 = 9

5) c d e f – SUM("3" + "4", "5" + "6") = 34 + 56 = 90 mod 10 = 0

6) x a e s – SUM("24" + "1", "5" + "19") = 241 + 519 = 760 mod 10 = 0
(goes to Index 1 because 0 is already filled)

Extra Credit (2 Points) Based on the above results, whose hash function is better and why?

Draco's function is better because Harry's is not good. Harry's function is completely dependent on the 2nd and 4th letter. This doesn't give enough diversity to map a wider range of data. For example, trying to find "a b c b" will have the same key as all the other combinations with the second letter and fourth letter both being 'b'. Draco's function is not dependent on any two letters, so we would expect more diversity in the mapping.

3. (20 Points) Data Structure applications

Answer the following questions based on the applications of Data structures.

a. In the project you are developing, adding each data must be a constant time operation. Data need not be maintained in any certain order. The amount of data grows dynamically. What data structure would you use and why?

We would use a Linked List. Linked List can add elements into in constant time as all it does is sets the data to be the current node's 'nextNode'. It also is known for being the data structure that grows dynamically. It will also hold data in the order put in so that satisfies the "unneed" for maintenance.

b. You are writing a program to make a turtle walk through a maze. In order for the turtle to make the correct decisions, you must use backtracking to keep track of all the moves made before. What data structure would you use and why?

I would use a directed and weighted graph. This way we can see the route of the turtle and backtrack its route by going backwards on the graph. And if there are multiple routes, the turtle could have gone, we can determine which one based on the weight of the route the turtle is currently on.

c. A Salesman is starting from his headquarters City A and has to visit 5 other cities. Given the distance between each of the cities and considering that all roads travel both ways (no direction to roads) , you need to find the shortest path that connects all the cities. Assume you are starting from City A and assume all cities are connected. What algorithm would you use and why?

I would use Prims Algorithm. It is preferred when there are more edges than nodes. Because all the cities are connected, then there will be more edges. It is also preferred for undirected weighted graphs. The weight in this case being the distance from each city. Lastly, Prim's algorithm will always be connected and starts at a source. Meaning we can find one minimum path that will connect all cities in the shortest path.

d. My phonebook is now digitized, and my database is sorted alphabetically. What algorithm should I use to find a person from the database and why?

I would use the Binary Search algorithm. Because the phonebook is already sorted alphabetically, using a binary search will take the quickest, O(logn). So it would keep halving the phonebook until it reached the name I was finding. It's the most similar to how we would do it manually as well.

4. (20 Points) Graphs

Given a weighted, undirected graph with V nodes, answer the following as best as possible, with a brief explanation. Assume all weights are non-negative.

a) If each edge has a unique weight, what can you say about the MST?

There exists only one MST for this graph

b) If the cost of an MST is ***c***, what can you say about the shortest distances returned by Dijkstra's algorithm when run with an arbitrary vertex ***s*** as the source?

The shortest distances will always be less than or equal to c

c) How many edges exist in the MST with V nodes?

In the MST, there are always V-1 edges

d) If one vertex in the graph does not have any edges attached to it. What does that tell you about the Minimum Spanning Tree?

The MST will not exist. A MST must be connected and with no edges, that it is impossible.

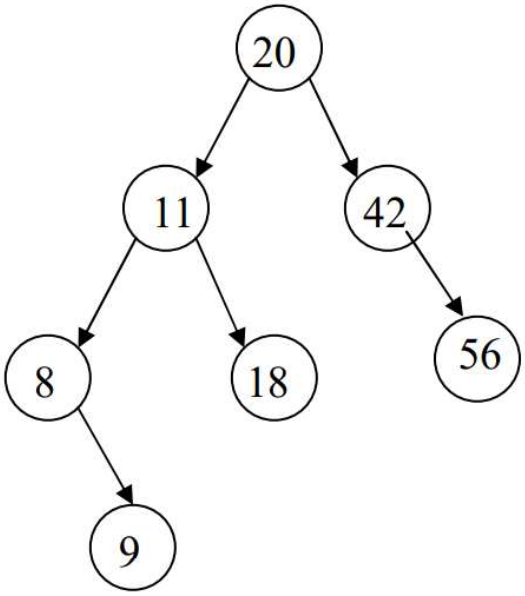
e) What algorithm would you use to find the MST if there are two edges between every given pair of vertices.

I would use Prim's Algorithm because it is preferred for graphs with more edges than vertices.

f) (1 Point Extra credit) – Other than Prim's and Kruskal's find one other algorithm to find Minimum Spanning Tree

5. (20 Points Total) Trees

Consider the following Tree:



a. (6 Points) Write traversals of the tree shown above:

Pre-Order: 20 – 11 – 8 – 9 – 18 – 42 – 56

Post-Order: 9 – 8 – 18 – 11 – 56 – 42 - 20

In-Order: 8 – 9 – 11 – 18 – 20 – 42 – 56

b. (6 Points) Consider the above Tree:
Circle yes or no to indicate whether the tree above might represent each of the following data structures.
If you circle no, you must describe one way to modify the tree in order to make the answer into a yes.

- AVL tree yes no
- Binary Search Tree yes no
- Full Binary Tree yes no

To be full, each non-leaf needs 2 children. Add one more leaf to 8 and you have a full binary tree.

c. (8 Points) AVL Trees (Independent question not relating to above tree)
Draw the AVL tree that results from inserting the keys:
56, 78, 90, 23, 1, 4, 76, 34
in that order into an initially empty AVL tree.
Showcase all intermediate steps and circle your final answer. Partial credit is awarded for correct intermediate trees.