/* CS261- Assignment 3 - Part 2: Comparison Questions*/
/* Name: Sky Wilson
 * Date: 4/21/2105
*/

1) Which of the implementations uses more memory? Explain why.

* The linked list implementation used more memory than the dynamic array implementation. The linked list began getting memory feedback at n = 16,000 whereas, the dynamic array didn't get its first memory feedback until n = 64,000.

* The linked list utilized more memory than the dynamic array because for one, the linked list had a greater intial overhead due to the pointers that are member variables of the struct LinkedList and struct DLink's. So if as n increases, more elements are created that require extra memory for their own respective member pointers. This would also explain why both implementations got memory feedback at differnt n's. The linked list would at n = 16,000 would have to have memory for the several member pointers that each of the 16,000 link's would require. The dynamic array would not require this extra allottment of memory and so at n = 16,000 there would only need to be memory for the elements of the array and their 1 TYPE member variable. Even if the linked list only had 1 pointer member, for n elements, you need extra memory for n pointers.

2) Which of the implementations is the fastest? Explain why.

* The dynamic array implementatin had the fastest run times.

* The dynamic array's runtime advantage can be attributed to several variables. First, the linked list has guaranteed O(1) on insertion and removal (depending on whether the list is single or doubly linked), the dynamic array has an amortized O(1)+ on insertion on insertion and removal operations by virtue of the nature of the dynamic array as a structure. However, the dynamic array's allocations and deallocations of memory do not occur every insertion or removal operation, only when size = capacity or when size = 1/2 capacity (or some variation). Second, due to the nature of the linked list as a structure, the elements of the linked list do not exist contiguously in memory as they do in the dynamic array. So when performing an insertion or removal operation on a linked list, the positioning of the next element is not a constant distance, and so execution times can be slowed. The dynamic array has the benefit of a cache unit arrangement which optimizes the execution of insertion and removal operations, and the dynamic array's allocations and deallocations of memory occur less frequently than the linked list, resulting in an amortized run time of O(1)+. So the relational constraints that make up the dynamic array abstraction, result in faster run times in relation to linked list structures.

3) Would you expect anything to change if the loop performed remove() instead of contains()? If so, what?

* Using removeList() in place of containsList() I would expect an increase in the run time of the linked list because in removeList(), if the value of the element in question (index i in a loop) is equal to the value being removed, then the next and prev pointers of the list need to be reassigned to the their logically respective neighbors (shifted) to represent the removal of an element. Memory would also need to be allocated for a struct DLink* temp that is used to reattach the links post removal and then this temp must be freed to remove the value to be removed. The linked list has the benefit not requiring a nested loop within its containsList(), however, memory must be allocated to a temp pointer (which

also must be freed) and member pointers of neighbor links must be reassigned to represent the reattachment of the linked list post link removal. For these reasons I would expect the swapping of the containsList() and the removeList() to increase the run time of this linked list.

* Using removeList() in place of containsList() I would also expect an increase in the run time of the dynamic array because in removeList(), if the value of the element in question (the element used as an argument of the conditional) another loop runs within the initial n outside loop. This inner loop runs from the index of the element with the matching value of the value to be removed, until the index of the size of the dynamic array. This inner loop shifts all values from the removed element til the last element in the dynamic array. In the containsList() the was a loop that ran n times, but even if the conditional within that loop was satisfied, there were no more loops and there was no shifting of elements and values. So by swapping removeList() for containsList() the dynamic array's runtime increases due to a nested loop within the function and the required shifting of values when a value to be removed is found within the dynamic array.

Course Modules: DATA ST... | linked_list_demo.mp4 | CS261_Assign2_Part2_Timi...

https://oregonstateuniversity-my.sharepoint.com/personal/wilsonsk_onid_oregonstate_edu/_layouts/15/WopiFrame.aspx?sourcedoc={8FF927EE-70FB-4906-A406-C398934A71A4}&file=CS261_Assign2_Part2_Timing_and_Memo

Search

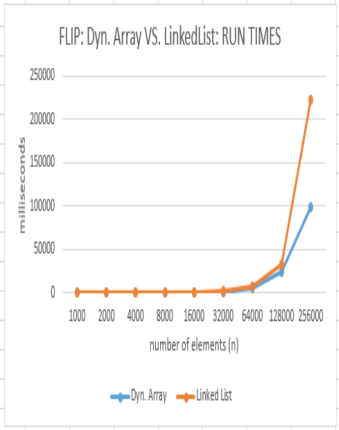
DEFINITIONS | Inbox (10) - wilsonsk@... | DATA STRUCTURES (C... | Blackboard Learn | College of Engineering... | Student Contact Infor... | Vim Cheat Sheet | MyProgrammingLab | ... | KeepVid: Download on... | WebDev | EDUCATION | CS261 | OSU | CS275 | Vim Cheat Sheat for Pr...

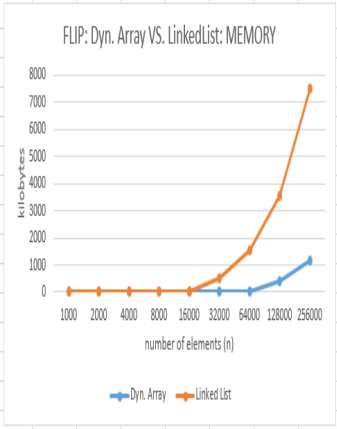
Excel Online | Wilson, Skyler M - ONID | CS261_Assign2_Part2_Timing_and_Memory | Share | Wilson, Skyler M - ONID

FILE | HOME | INSERT | DATA | REVIEW | VIEW | Tell me what you want to do | OPEN IN EXCEL

Undo | Clipboard (Paste, Cut, Copy) | Font (Calibri, 11, B I U) | Alignment (Wrap Text, Merge & Center) | Number (Number Format) | Tables (Survey, Format as Table) | Cells (Insert, Delete) | Editing (AutoSum, Clear, Sort, Find)

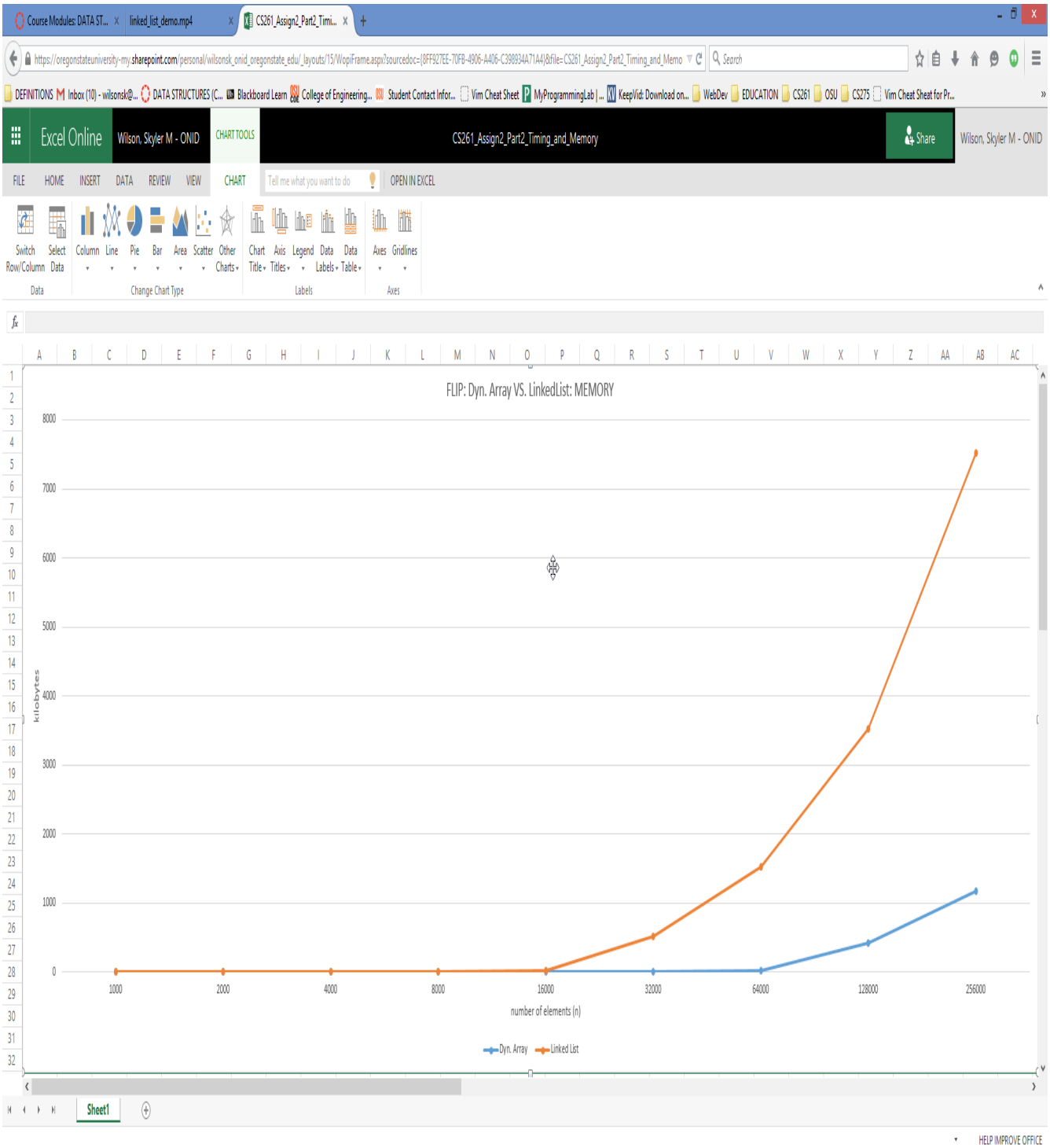
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1					NON FLIP	TIMING					FLIP	TIMING				NON FLIP	MEMORY				FLIP	MEMORY							
2					n	Dyn. Array	Linked List					Dyn. Array	Linked List			n	Dyn. Array	Linked List				Dyn. Array	Linked List						
3					1000	0 ms	10 ms				1000	0	0			1000	0 KB	0 KB			1000	0	0						
4					2000	20 ms	60 ms				2000	0	0			2000	0 KB	0 KB			2000	0	0						
5					4000	120 ms	210 ms				4000	20	30			4000	0 KB	0 KB			4000	0	0						
6					8000	290 ms	860 ms				8000	100	120			8000	0 KB	0 KB			8000	0	0						
7					16000	1210 ms	3460 ms				16000	380	510			16000	0 KB	0 KB			16000	0	24						
8					32000	4880 ms	9980 ms				32000	1540	2060			32000	0 KB	0 KB			32000	0	524						
9					64000	19640 ms	52040 ms				64000	6160	8170			64000	0 KB	0 KB			64000	28	1524						
10					128000	76730 ms	223650 ms				128000	24670	32820			128000	0 KB	1132 KB			128000	416	3524						
11					256000	2820000 m	689890 ms				256000	98610	223260			256000	0 KB	5144 KB			256000	1168	7524						

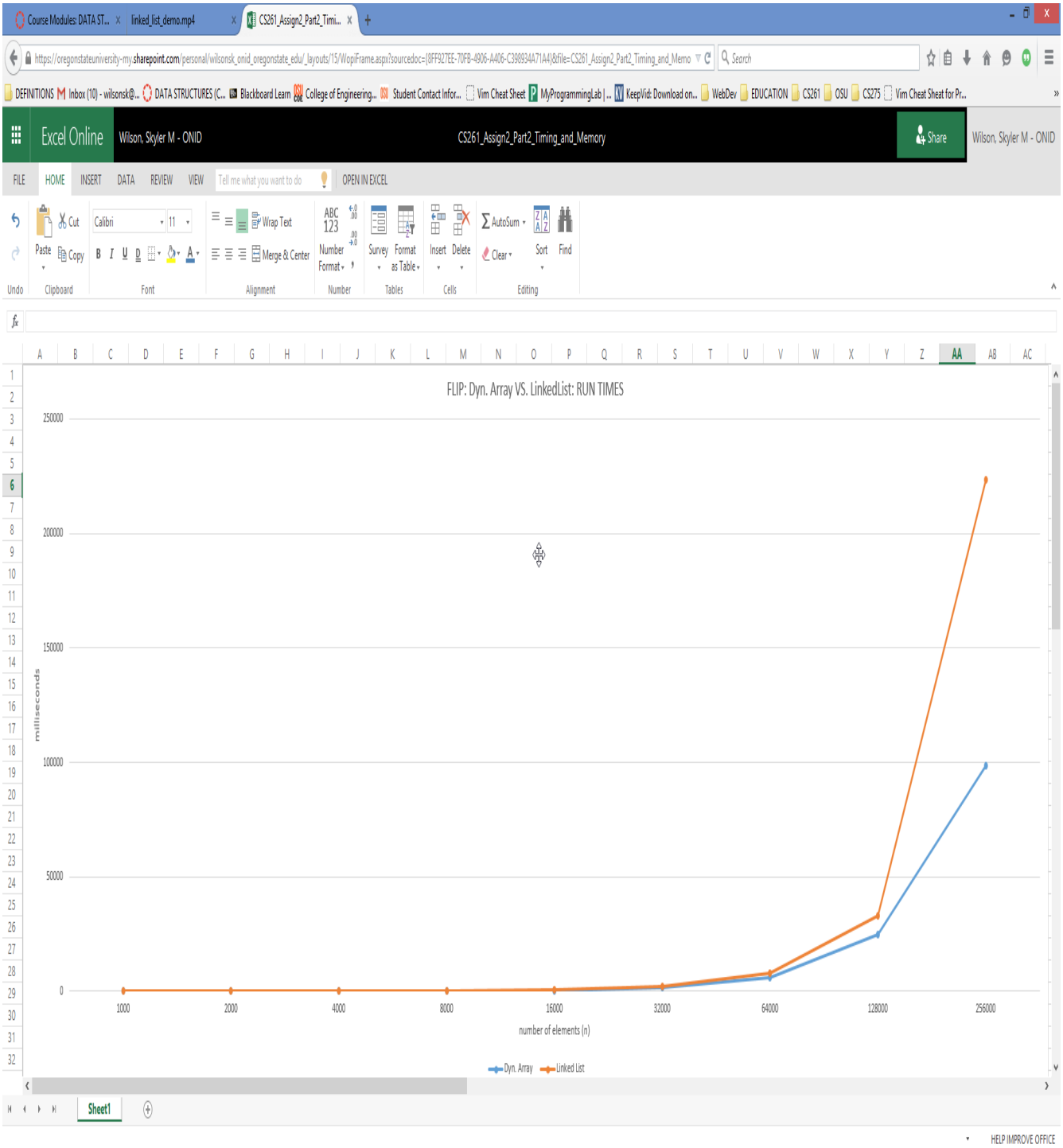




Sheet1

HELP IMPROVE OFFICE


Course Modules: DATA ST...
linked_list_demo.mp4
CS261_Assign2_Part2_Timi...
Excel Online
Wilson, Skyler M - ONID
CHART TOOLS
CS261_Assign2_Part2_Timing_and_Memory
Share
FILE
HOME
INSERT
DATA
REVIEW
VIEW
CHART
Tell me what you want to do
OPEN IN EXCEL
Switch Row/Column
Select Data
Column
Line
Pie
Bar
Area
Scatter
Other Charts
Chart Title
Axis Titles
Legend
Data Labels
Data Table
Axes
Gridlines
Data
Change Chart Type
Labels
FLIP: Dyn. Array VS. LinkedList: MEMORY
kilobytes
8000
7000
6000
5000
4000
3000
2000
1000
0
1000
2000
4000
8000
16000
32000
64000
128000
256000
number of elements (n)
Dyn. Array
Linked List
Sheet1
HELP IMPROVE OFFICE


Course Modules: DATA ST...
linked_list_demo.mp4
CS261_Assign2_Part2_Timi...
Excel Online
Wilson, Skyler M - ONID
CS261_Assign2_Part2_Timing_and_Memory
Share
FILE
HOME
INSERT
DATA
REVIEW
VIEW
OPEN IN EXCEL
FLIP: Dyn. Array VS. LinkedList: RUN TIMES
milliseconds
250000
200000
150000
100000
50000
0
1000
2000
4000
8000
16000
32000
64000
128000
256000
number of elements (n)
Dyn. Array
Linked List
Sheet1
HELP IMPROVE OFFICE