Which of the implementations uses more memory? Explain why.

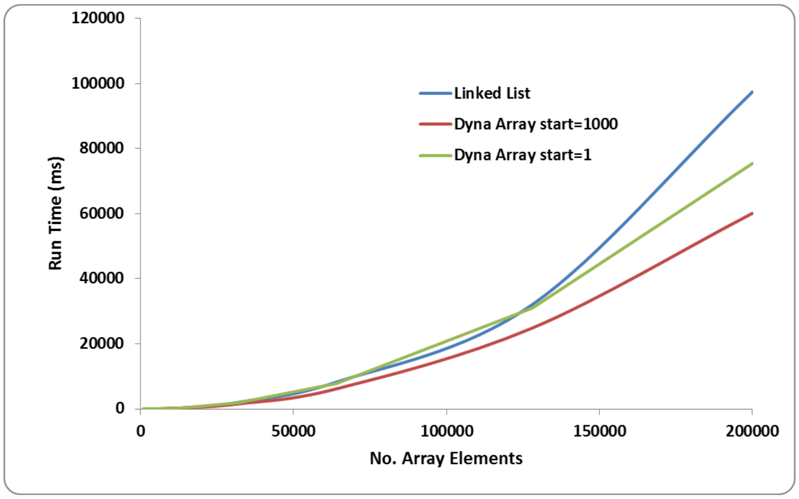
The linked list uses more memory. This is because each link is a struct that contains the value and pointers to both the next and previous links. This uses up more memory than just the dynamic array which only needs to hold the value.

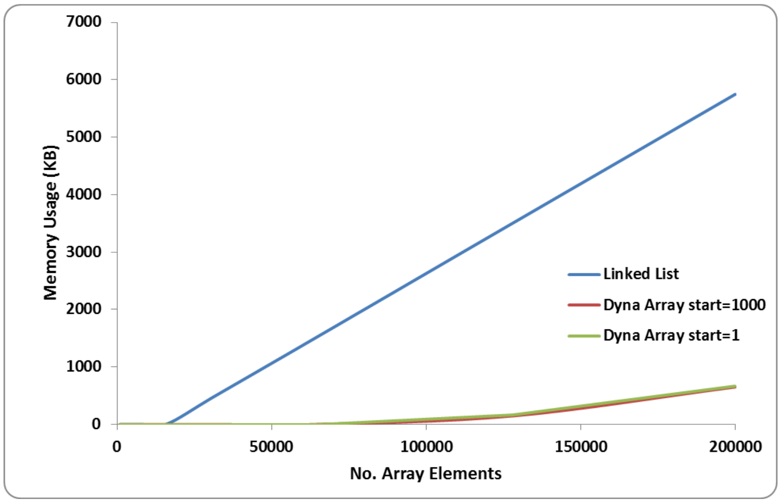
Which of the implementations is the fastest? Explain why.

The dynamic array is faster. They are pretty much the same for lower number of array elements but start to diverge at higher numbers. Inserting a new element in the linked list will take O(1) and $O(1)^+$ for the dynamic array. So for large values the two are basically the same time for inserting elements. Searching in both cases will be O(n) but the difference is in how each is searched. The dynamic array has direct access and the linked list needs a link pointer to traverse the list. This means that searching for n elements in a linked list requires the creation of n link pointers. I think the extra overhead of doing this can account for some of the difference. I was interested in seeing what would happen if the dynamic array was initialized to 1 instead of 1000. I included the data for that below. The extra cost of resizing the array makes it about as slow as the linked list for the lower number of elements. But this extra cost matter less for larger arrays and it still ends up being faster than the linked list.

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

Running remove() is still O(n) for both. I would expect the times to go up slightly due to the time it takes to free each element in the array but the overall trends should remain the same.





	Size (num)	Time (ms)	Memory (KB)
LinkdedList	1000	0	0
	2000	0	0
	4000	30	0
	8000	120	0
	16000	490	4
	32000	2040	504
	64000	8230	1500
	128000	32020	3504
	200000	97440	5748
Dyna Array	1000	0	0
starting size 1000	2000	0	0
	4000	20	0
	8000	100	0
	16000	400	0
	32000	1560	0
	64000	6160	0
	128000	24830	148
	200000	60130	652
starting Size 1	1000	0	0
	2000	0	0
	4000	30	0
	8000	120	0
	16000	480	0
	32000	1930	0
	64000	7710	0
	128000	30890	160
	200000	75430	672