Analysis on problem2

Woohyuk Yang

yangwo@oregonstate.edu

Firstly, I made a table to compare two different data structures(Dynamic Array and Linked List) when it comes to memory usage and running time. Running time for contains() function had been measured with proper tool provided on flip server. I input various number from 2^10 to 2^18 following the instruction.

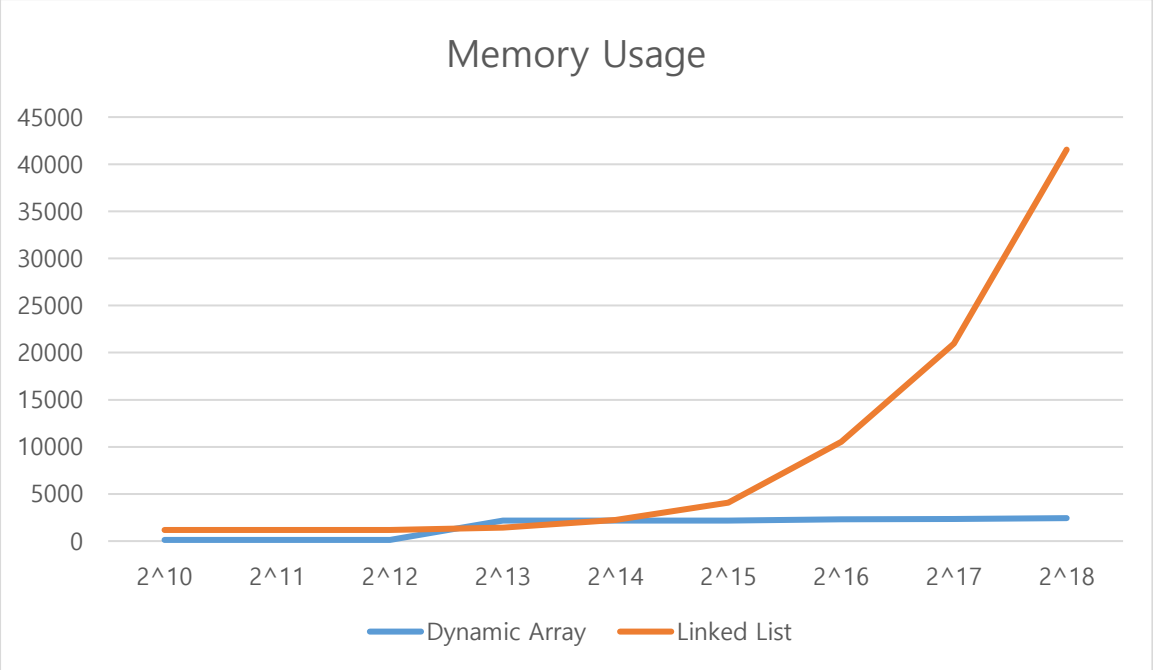
To compare two data structure visually, I made graphs comparing Dynamic Array and Linked List. The graphs below the table were made with the table below.

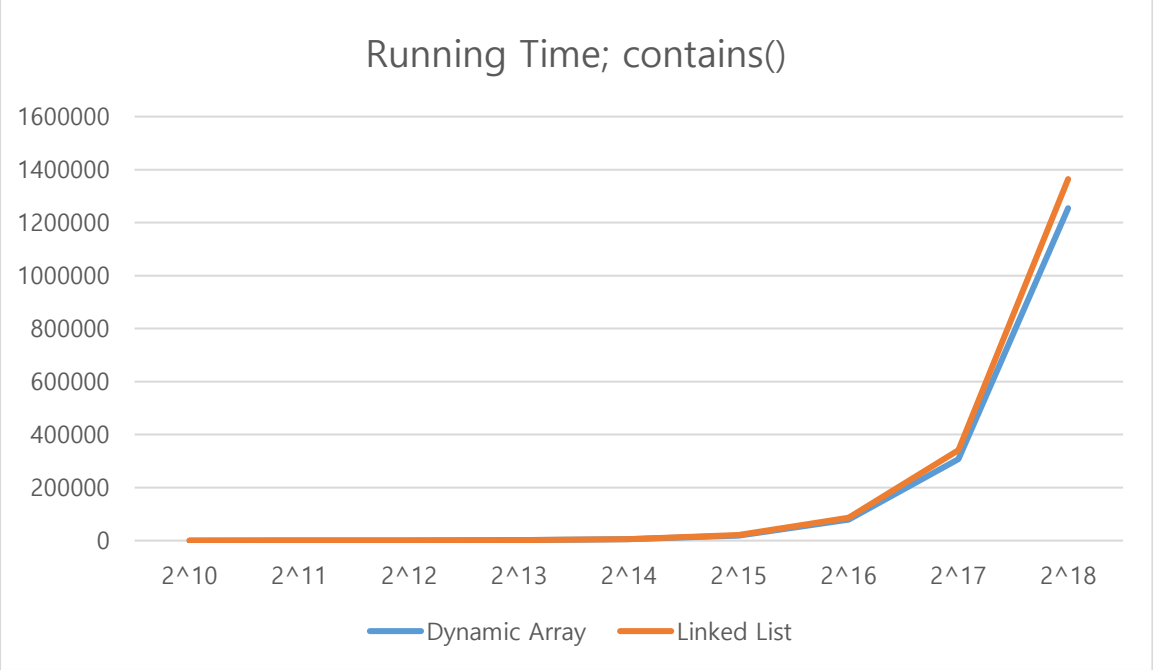
Dynamic array; Running time is for contains() function.

	2^10 1024	2^11 2048	2^12 4096	2^13 8192	2^14 16384	2^15 32768	2^16 65536	2^17 131072	2^18 262144
Memory usage	124KB	124KB	124KB	2,172KB	2,172KB	2,172KB	2,308KB	2,356KB	2,436KB
Running time	20ms	90ms	320ms	1260ms	4980ms	19090ms	79160ms	306890ms	1254380ms

Linked List; Running time is for contains() function.

	2^10 1024	2^11 2048	2^12 4096	2^13 8192	2^14 16384	2^15 32768	2^16 65536	2^17 131072	2^18 262144
Memory usage	1180KB	1180KB	1180KB	1436KB	2228KB	4076KB	10,540KB	20,964KB	41548KB
Running time	30ms	90ms	350ms	1300ms	5210ms	21330ms	86240ms	341340ms	1364310ms


Memory Usage
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
2^10
2^11
2^12
2^13
2^14
2^15
2^16
2^17
2^18
Dynamic Array
Linked List


Running Time; contains()
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
2^10
2^11
2^12
2^13
2^14
2^15
2^16
2^17
2^18
Dynamic Array
Linked List

Questions.

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

   As we can see from the previous graph, linked list uses more memory. Because every single element in the array only required the size of the type of the array for its own memory usage while the linked list required more that. Each link of the linked list required pointer variable next, and prev in addition to a variable which a value could be stored.

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

   Using dynamic array is faster than using linked list for contain() function. contain function is the function which look for the certain value in the chunk of memory allocated. As the elements in the dynamic array are next to each other. So it is faster to go over from index 0 to where we need to access to. But each node in linked list is connected each other with pointer variable which means there are pointer overheads also to go from the first link to the link we need to access to.

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

I expected there would be change of running time when we measure running time of remove() instead of contain(). Dynamic array and linked list are different when it comes to the way of removing its own elements or nodes. So I expected running time of remove() function for dynamic array would increase a lot more than that of linked list. I expected the memory usage will be same also because we are only changing the function to measure running time of.

I predicted the result based on the way of removing elements. Talking about how that differs and affect the performance for each data structure, firstly, dynamic array removes an element moving every element in the array one index forward so the element which needs to

be removed gets changed to the value of the element which was just right after the will-be removed element. We should focus on this way of remove() function. As the size of the array increases, the number of elements which should move forward increases also. So it does take more time.

Secondly, linked list removes the element just changing the value stored in the pointer variables of nearby links of the node which should be removed. So it does not require any move of nodes like dynamic array does. So I guessed running time for remove() function will be really short in this case.

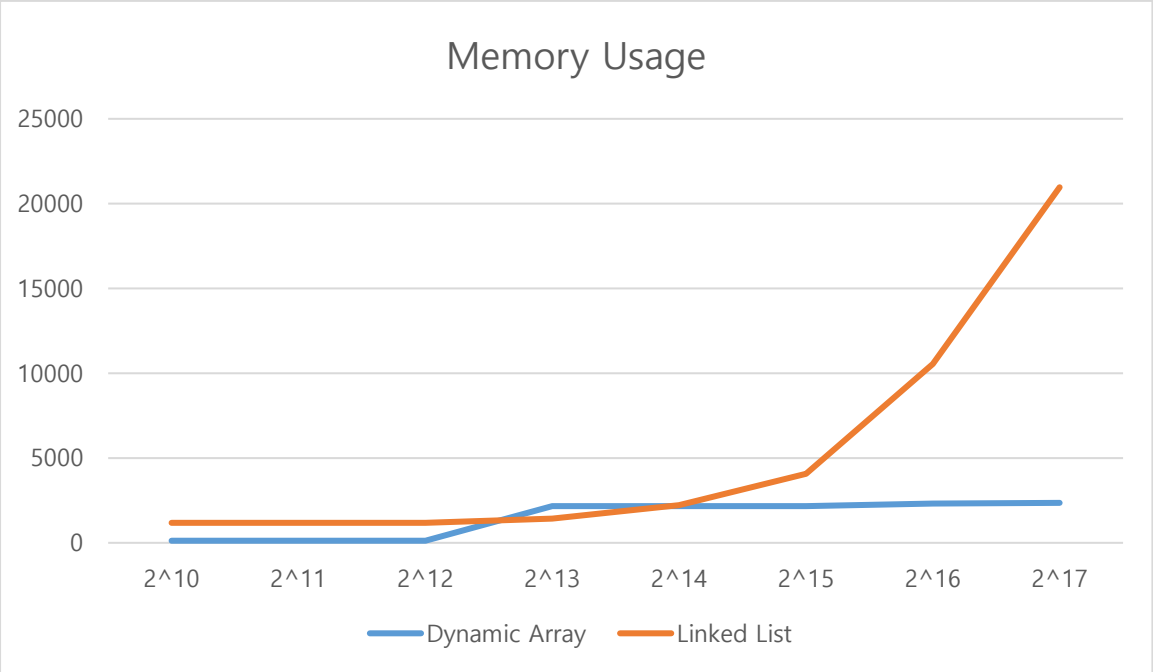
To prove my prediction, I measured running time of remove function for each data structure using the same code provided to measure running time for contain() function. The results are below on the table and the graph.

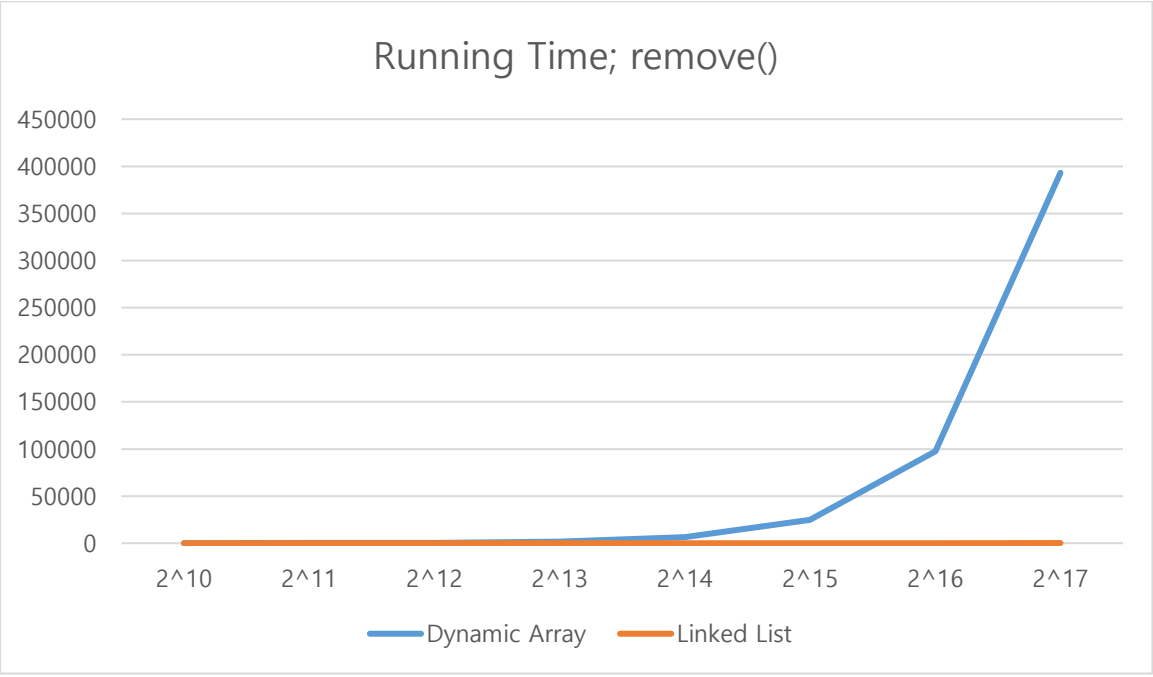
Dynamic array; Running time is for remove() function.

	2^10 1024	2^11 2048	2^12 4096	2^13 8192	2^14 16384	2^15 32768	2^16 65536	2^17 131072
Memory usage	124KB	124KB	124KB	2172KB	2172KB	2172KB	2308KB	2356KB
Running time	60ms	160ms	450ms	1660ms	6550ms	24730ms	97530ms	393130ms

Linked List; Running time is for remove() function.

	2^10 1024	2^11 2048	2^12 4096	2^13 8192	2^14 16384	2^15 32768	2^16 65536	2^17 131072
Memory usage	1180KB	1180KB	1180KB	1436KB	2228KB	4076KB	10,540KB	20,964KB
Running time	10ms	10ms	10ms	10ms	30ms	50ms	120ms	220ms





As I expected, memory usage is same for both data structure. While it is clear that it takes more time for remove function from the dynamic array than linked list as the input grows. I tried to measure time for 2^18 but it took too much time for dynamic array and I was able to find out the tendency so I did not measure running time for that input.