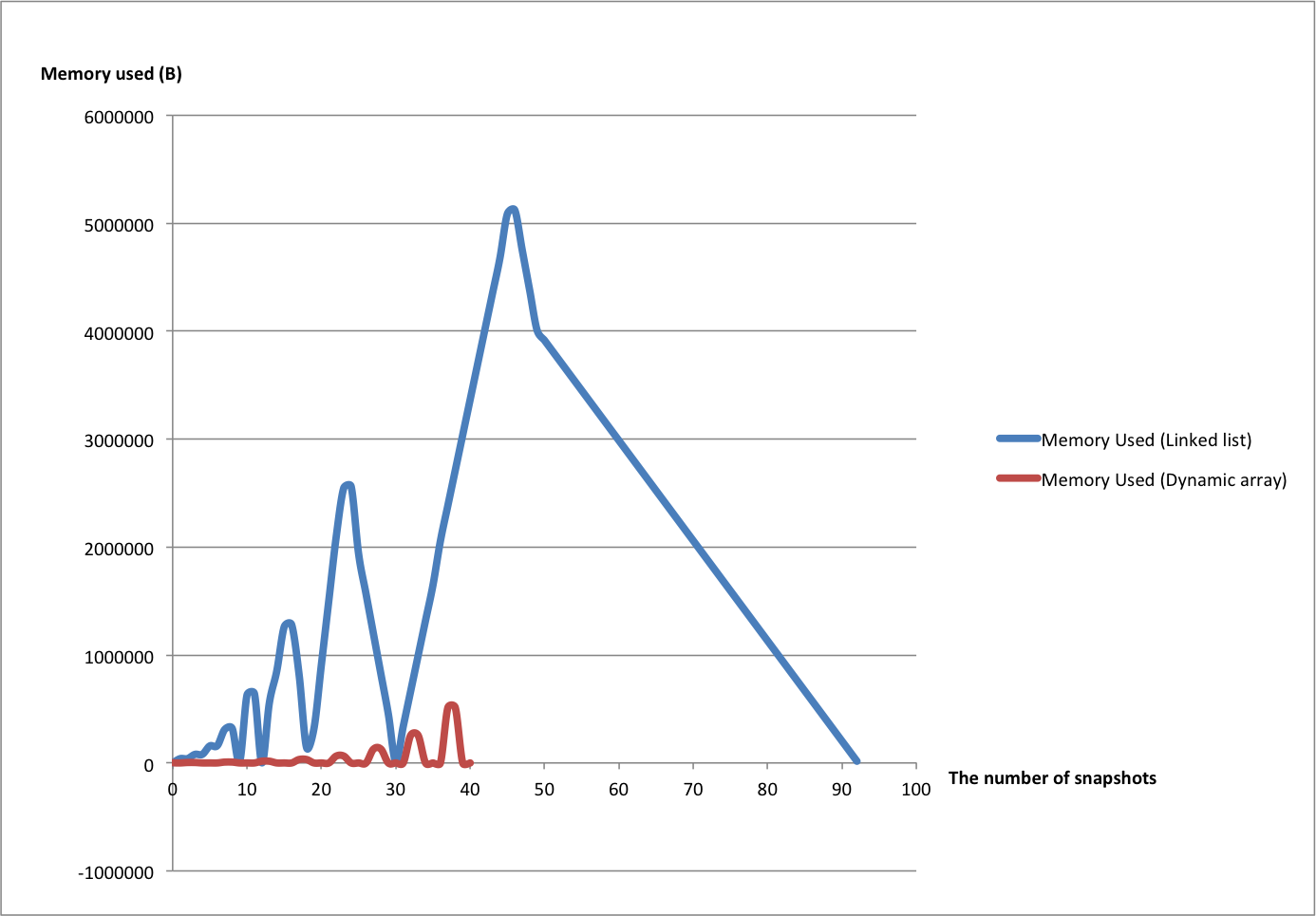
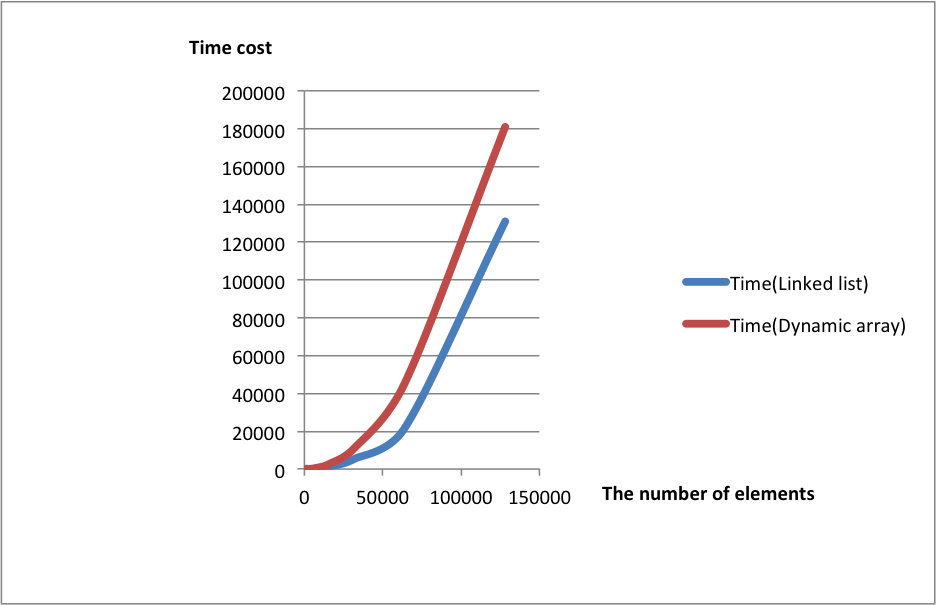
Memory used graph



Memory used table

|  |  |  |
| --- | --- | --- |
| snapshots | Memory Used (Linked list) | Memory Used (Dynamic array) |
| 0 | 0 | 0 |
| 1 | 39,640 | 24 |
| 2 | 40,120 | 4,032 |
| 3 | 79,400 | 4,032 |
| 4 | 80,120 | 24 |
| 5 | 157,760 | 0 |
| 6 | 160,120 | 24 |
| 7 | 310,280 | 8,032 |
| 8 | 320,120 | 8,032 |
| 9 | 15,320 | 24 |
| 10 | 629,720 | 0 |
| 11 | 640,120 | 24 |
| 12 | 1,000 | 16,032 |
| 13 | 556,080 | 16,032 |
| 14 | 853,040 | 24 |
| 15 | 1,261,360 | 0 |
| 16 | 1,280,120 | 24 |
| 17 | 816,840 | 32,032 |
| 18 | 143,240 | 32,032 |
| 19 | 319,280 | 24 |
| 20 | 913,200 | 0 |
| 21 | 1,507,120 | 24 |
| 22 | 2,101,040 | 64,032 |
| 23 | 2,546,480 | 64,032 |
| 24 | 2,560,120 | 24 |
| 25 | 1,936,280 | 0 |
| 26 | 1,565,080 | 24 |
| 27 | 1,193,880 | 128,032 |
| 28 | 822,680 | 128,032 |
| 29 | 451,480 | 24 |
| 30 | 11,160 | 0 |
| 31 | 338,520 | 24 |
| 32 | 665,880 | 256,032 |
| 33 | 993,240 | 256,032 |
| 34 | 1,320,600 | 24 |
| 35 | 1,647,960 | 0 |
| 36 | 2,057,160 | 24 |
| 37 | 2,384,520 | 512,032 |
| 38 | 2,711,880 | 512,032 |
| 39 | 3,039,240 | 24 |
| 40 | 3,366,600 | 0 |
| 41 | 3,693,960 |  |
| 42 | 4,021,320 |  |
| 43 | 4,348,680 |  |
| 44 | 4,676,040 |  |
| 45 | 5,085,240 |  |
| 46 | 5,120,120 |  |
| 47 | 4,748,880 |  |
| 48 | 4,377,680 |  |
| 49 | 4,006,480 |  |
| 50 | 3,913,680 |  |
| 51 | 3,820,880 |  |
| 52 | 3,728,080 |  |
| 53 | 3,635,280 |  |
| 54 | 3,542,480 |  |
| 55 | 3,449,680 |  |
| 56 | 3,356,880 |  |
| 57 | 3,264,080 |  |
| 58 | 3,171,280 |  |
| 59 | 3,078,480 |  |
| 60 | 2,985,680 |  |
| 61 | 2,892,880 |  |
| 62 | 2,800,080 |  |
| 63 | 2,707,280 |  |
| 64 | 2,614,480 |  |
| 65 | 2,521,680 |  |
| 66 | 2,428,880 |  |
| 67 | 2,336,080 |  |
| 68 | 2,243,280 |  |
| 69 | 2,150,480 |  |
| 70 | 2,057,680 |  |
| 71 | 1,964,880 |  |
| 72 | 1,872,080 |  |
| 73 | 1,779,280 |  |
| 74 | 1,686,480 |  |
| 75 | 1,593,680 |  |
| 76 | 1,500,880 |  |
| 77 | 1,408,080 |  |
| 78 | 1,315,280 |  |
| 79 | 1,222,480 |  |
| 80 | 1,129,680 |  |
| 81 | 1,036,880 |  |
| 82 | 944,080 |  |
| 83 | 851,280 |  |
| 84 | 758,480 |  |
| 85 | 665,680 |  |
| 86 | 572,880 |  |
| 87 | 480,080 |  |
| 88 | 387,280 |  |
| 89 | 294,480 |  |
| 90 | 201,680 |  |
| 91 | 108,880 |  |
| 92 | 16,080 |  |

Time cost graph



Time cost table

|  |  |  |
| --- | --- | --- |
| Elements | Time(Linked list) | Time(Dynamic array) |
| 1000 | 0 | 10 |
| 2000 | 20 | 50 |
| 4000 | 80 | 170 |
| 8000 | 350 | 690 |
| 16000 | 1350 | 2780 |
| 32000 | 5430 | 11220 |
| 64000 | 21760 | 45460 |
| 128000 | 131000 | 180990 |

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

Answer: The Linked List uses more memory. The reason is that every linked list node need to create two extra pointer in order to point their previous node and next node. These two pointers also need to allocate the same memory as their linked list node. It means every time when we create a new node, there will be allocated three block of memory of linked list node. Regarding dynamic array, in the structure of dynamic array, there is only one pointer need to allocate. Additionally it just needs to allocate a chunk of new memory as it is filled. Therefore the dynamic array cost less memory than Linked list.

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

Answer: The time cost of linked list and dynamic array is nearly same. The reason is that for the two data structure, performing contain() they all need to go through the whole data structure. Their big-oh complexity is all O(n).

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

Answer: If performing the remove() instead of contains() in the loop, The linked list will faster than dynamic array. The reason is that when we perform remove() in linked list, for this test case, every time we can find the node that needs to be remove at front of linked list, so does dynamic array. But for dynamic array, when it removes an element from the front, it also need to shift other elements back one to fill the gap which means more cost.