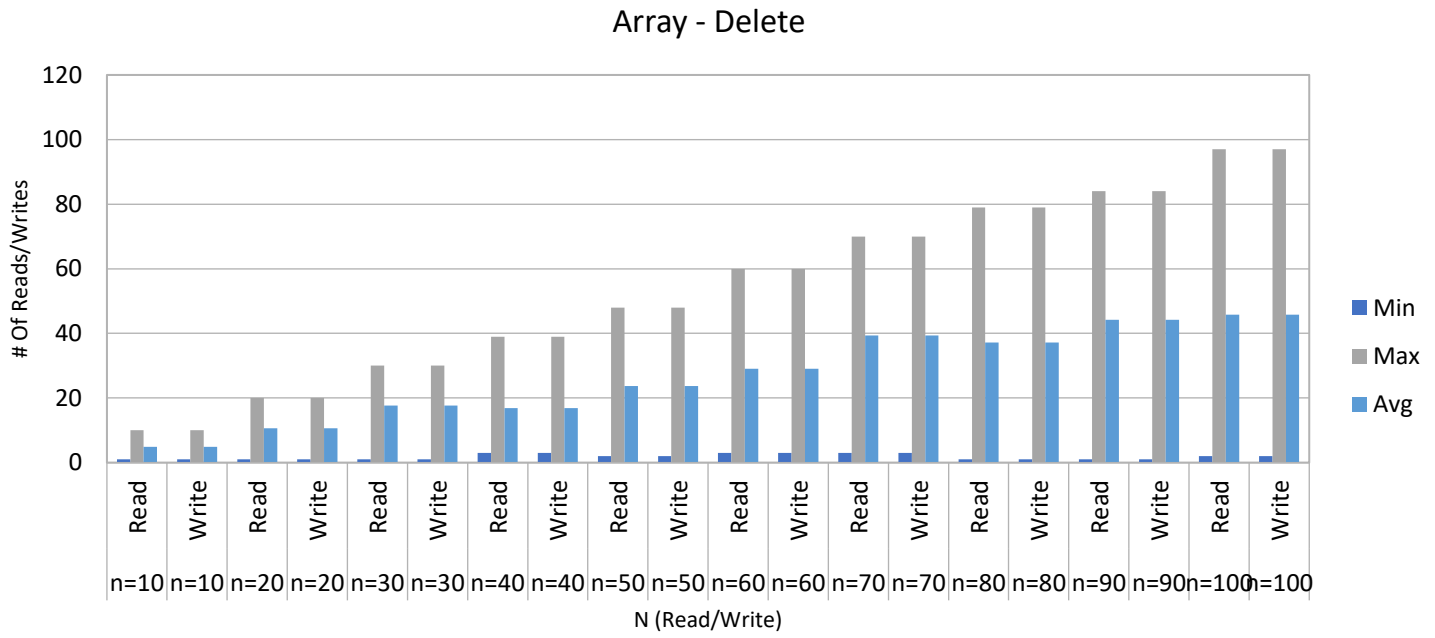
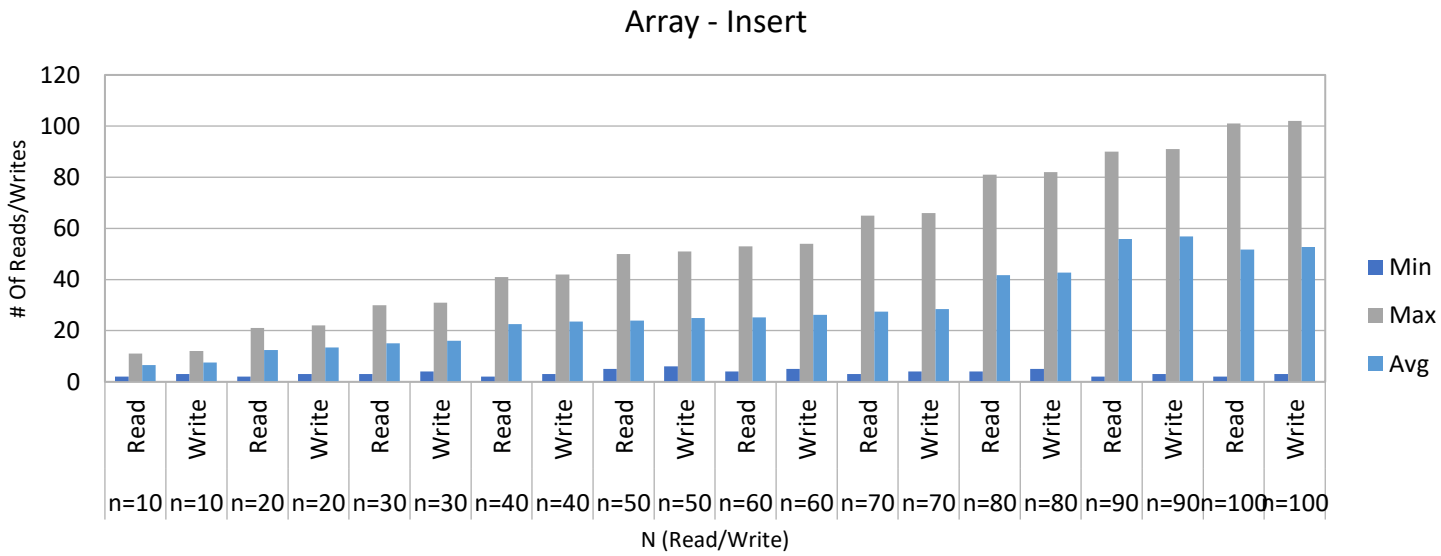
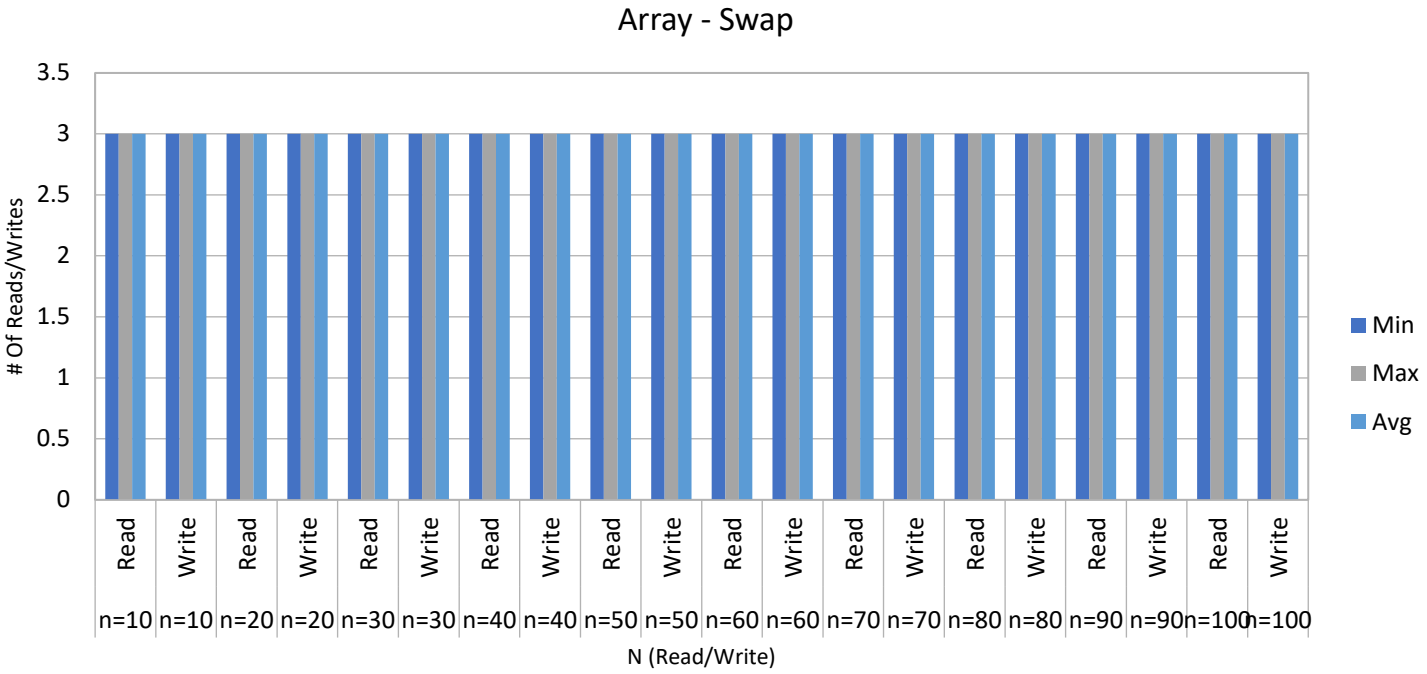
CIS 2520 – Assignment 1

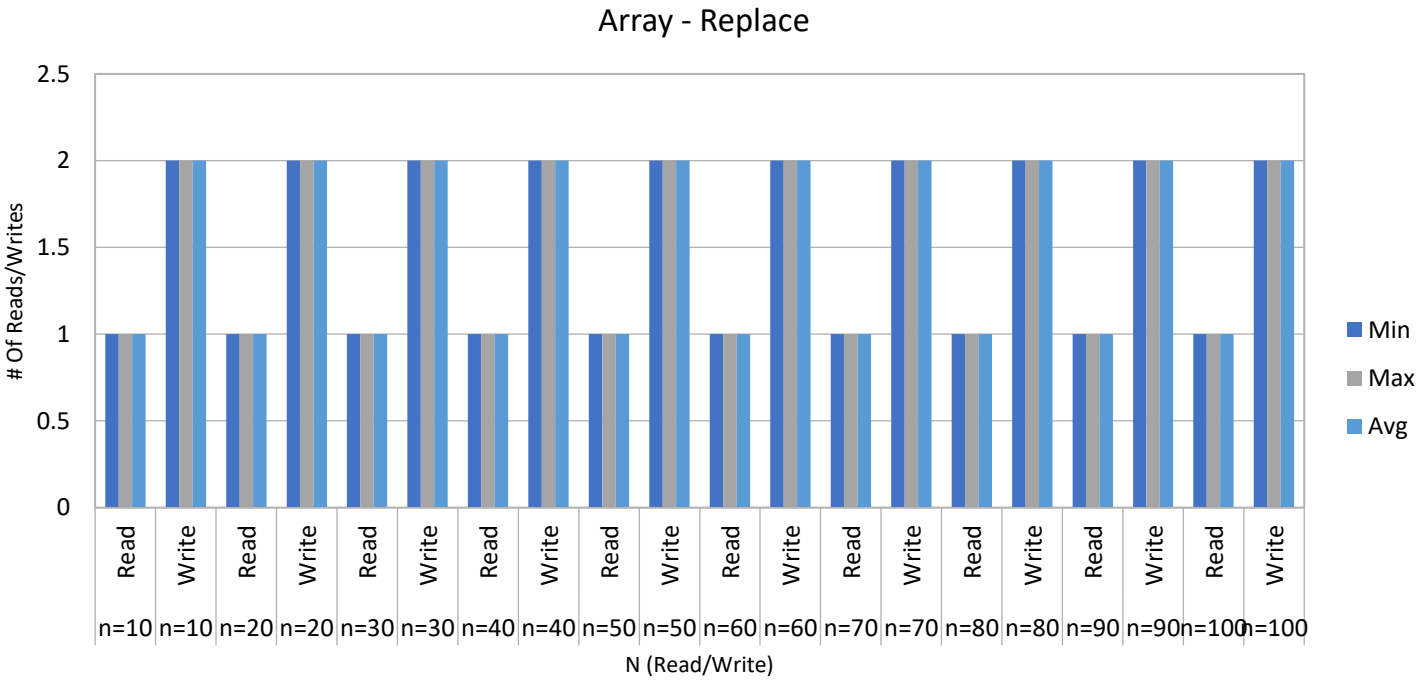
Graphs

Array:

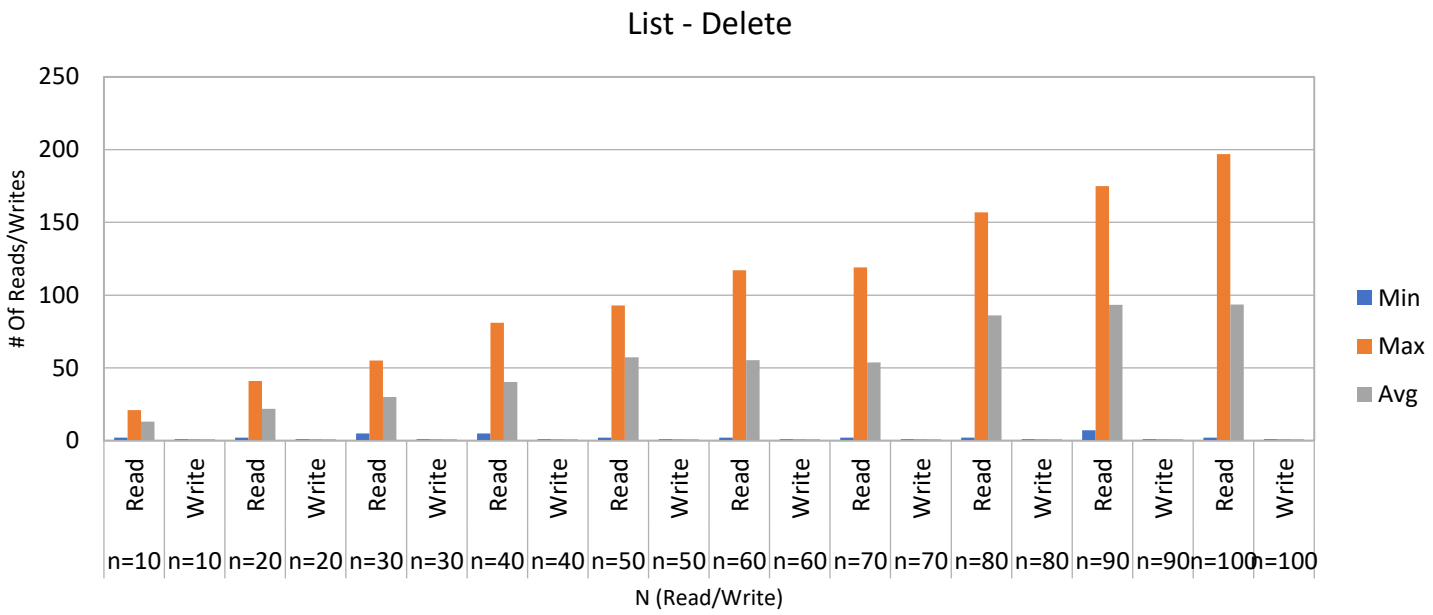


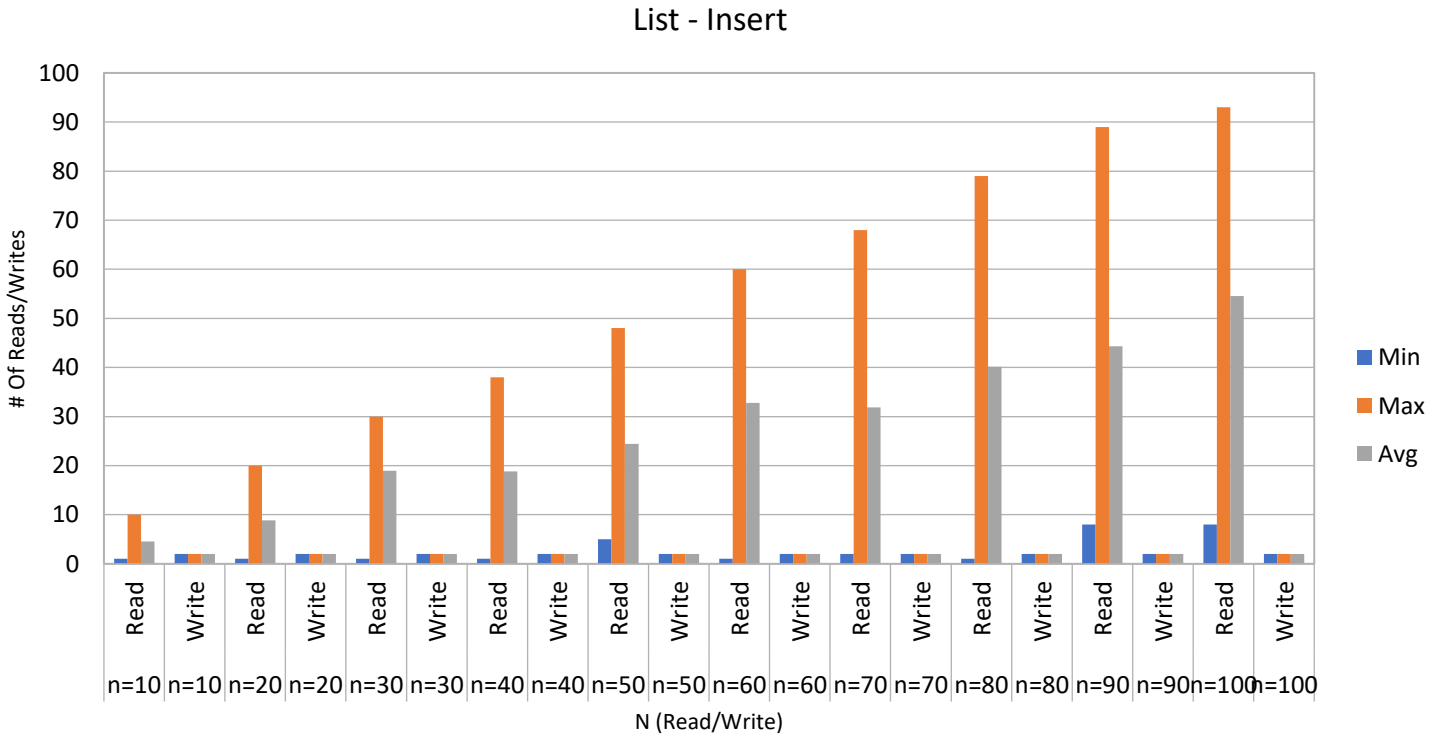


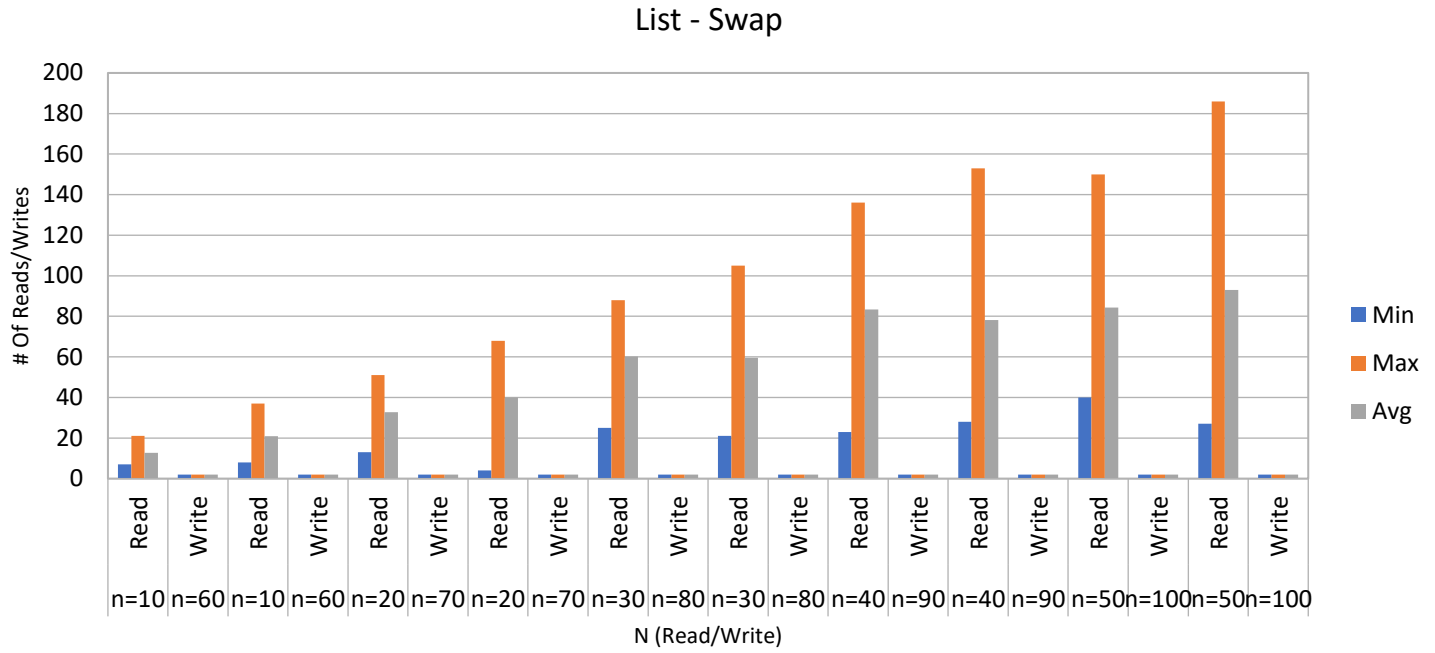

Array - Swap
3.5
3
2.5
2
1.5
1
0.5
0
Of Reads/Writes
Read
Write
n=10
n=20
n=30
n=40
n=50
n=60
n=70
n=80
n=90
n=100
N (Read/Write)
Min
Max
Avg

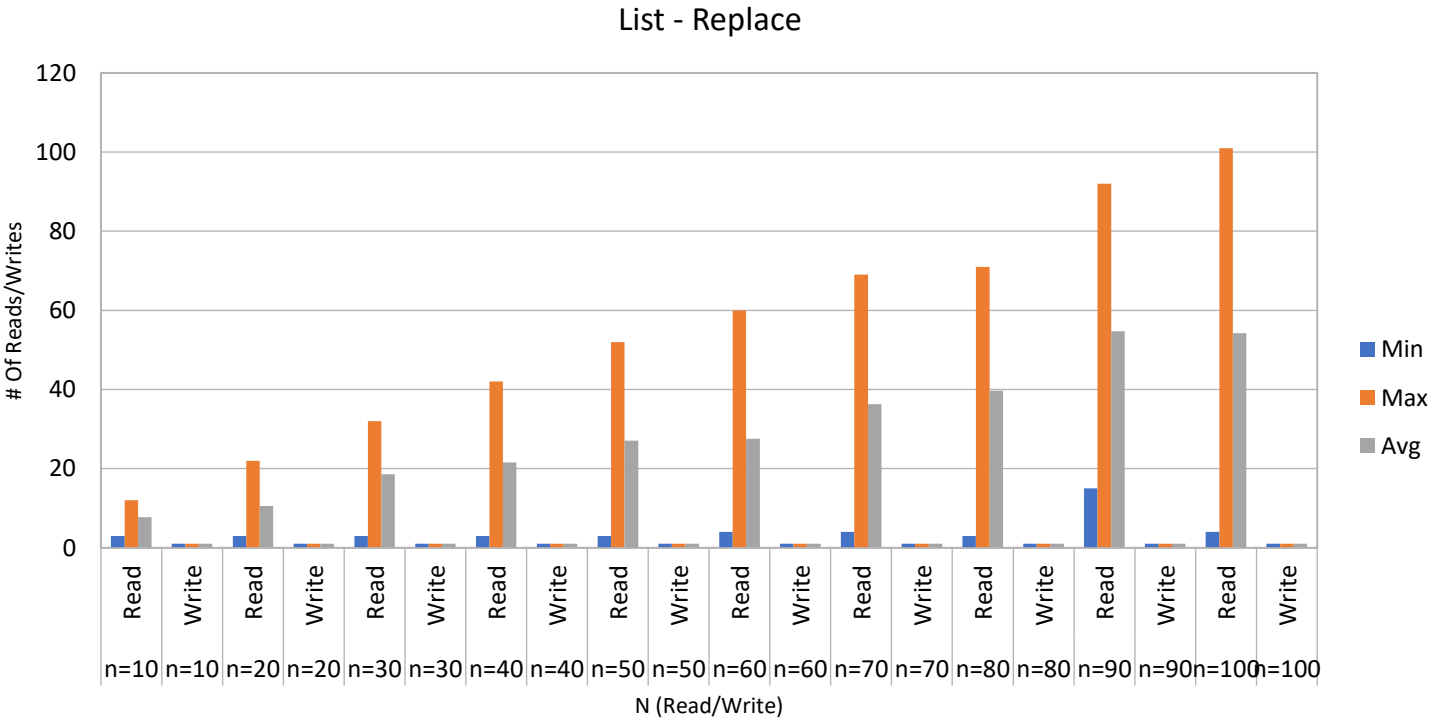

Array - Replace
2.5
2
1.5
1
0.5
0
Of Reads/Writes
Read
Write
n=10
n=20
n=30
n=40
n=50
n=60
n=70
n=80
n=90
n=100
N (Read/Write)
Min
Max
Avg

List


List - Delete
250
200
150
100
50
0
Of Reads/Writes
Read
Write
n=10
n=20
n=30
n=40
n=50
n=60
n=70
n=80
n=90
n=100
N (Read/Write)
Min
Max
Avg


List - Insert
100
90
80
70
60
50
40
30
20
10
0
Of Reads/Writes
Read
Write
n=10
n=20
n=30
n=40
n=50
n=60
n=70
n=80
n=90
n=100
N (Read/Write)
Min
Max
Avg


List - Swap
Of Reads/Writes
200
180
160
140
120
100
80
60
40
20
0
Read
Write
n=10
n=60
n=20
n=70
n=30
n=80
n=40
n=90
n=50
n=100
N (Read/Write)
Min
Max
Avg


List - Replace
Of Reads/Writes
120
100
80
60
40
20
0
Read
Write
n=10
n=20
n=30
n=40
n=50
n=60
n=70
n=80
n=90
n=100
N (Read/Write)
Min
Max
Avg

Thoughts

Based on the above graphs, it would certainly be better to use a linked list when using an SSD (Solid State Drive) as their lifespans are limited by write cycles, whereas read cycles don't affect them as much. Linked List's are much better suited for applications where reads can be processed significantly faster than writes. This is due to LL's writing only once or twice for many reads. LL's will also scale much better in this sense as it appears that for any of the functions tested, the writes required to perform the function are constants, meaning they stay the same for all values of n. For all cases where one might need to insert a plethora of data into the data structure, a Linked List would lend itself much better to the application. For example, in a contact book, an implementation making use of a Linked List would make more sense than an array. In a contact book, the user will want to insert contacts and have them be automatically sorted. This can be achieved by finding the proper position to store the new contact and inserting it, taking less read/writes cycles than an array performing a similar act.

On the other hand, Arrays are better suited for applications that make heavy use of swapping and replacing specific elements. The read and write values for these functions appear to both be constants, having no correlation to the size of the dataset. A good use-case for an array would be anything where you have a fixed number of elements, insertion and deletion are not needed, and swapping and replacing are heavily used. For example, when storing and displaying images, an implementation with an array would be faster than a Linked List. This is due to screens having a fixed pixel count, and replacement is the most used function on the array. Another example where an implementation using an array would be beneficial is when the size of the dataset can grow and shrink but only from the far end. This is because inserting and deleting at the end of the array is very quick, but

very slow at the start. This is opposite to a linked list as inserting and deleting at the beginning of a linked list is fast but slow at the end.

William Moore

1061752