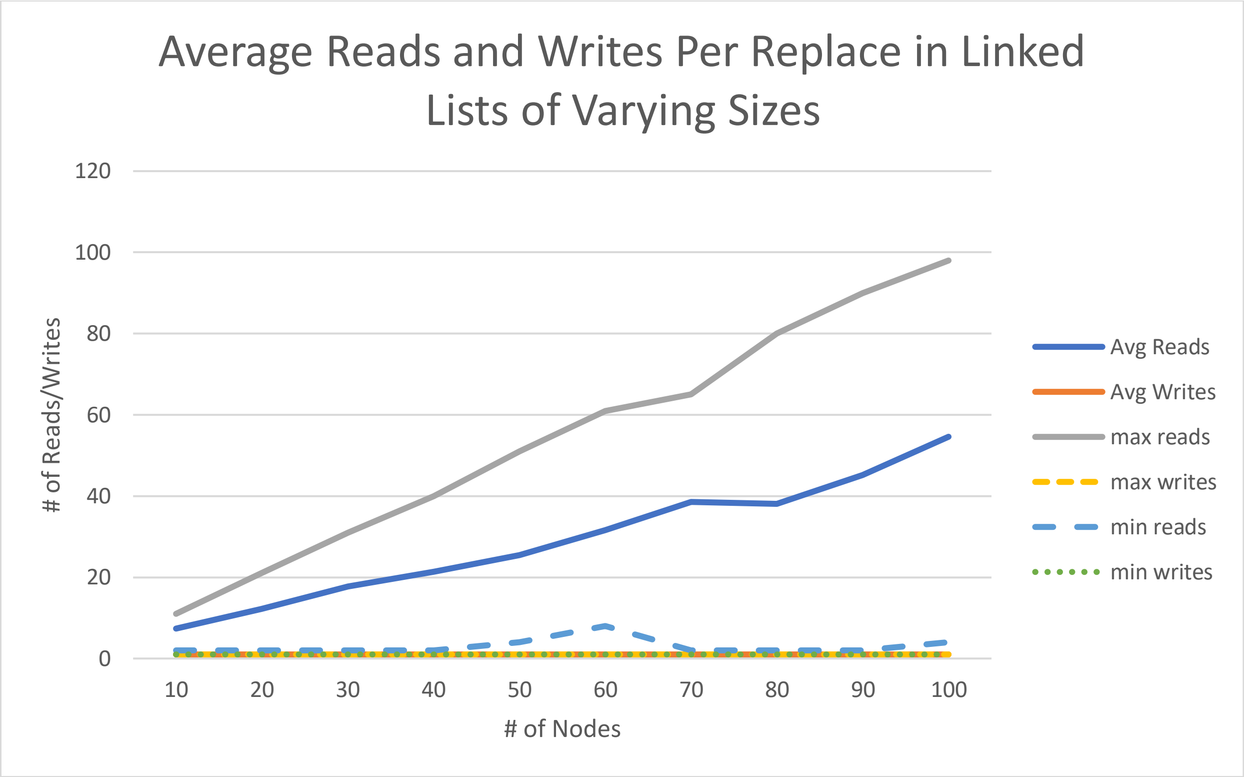
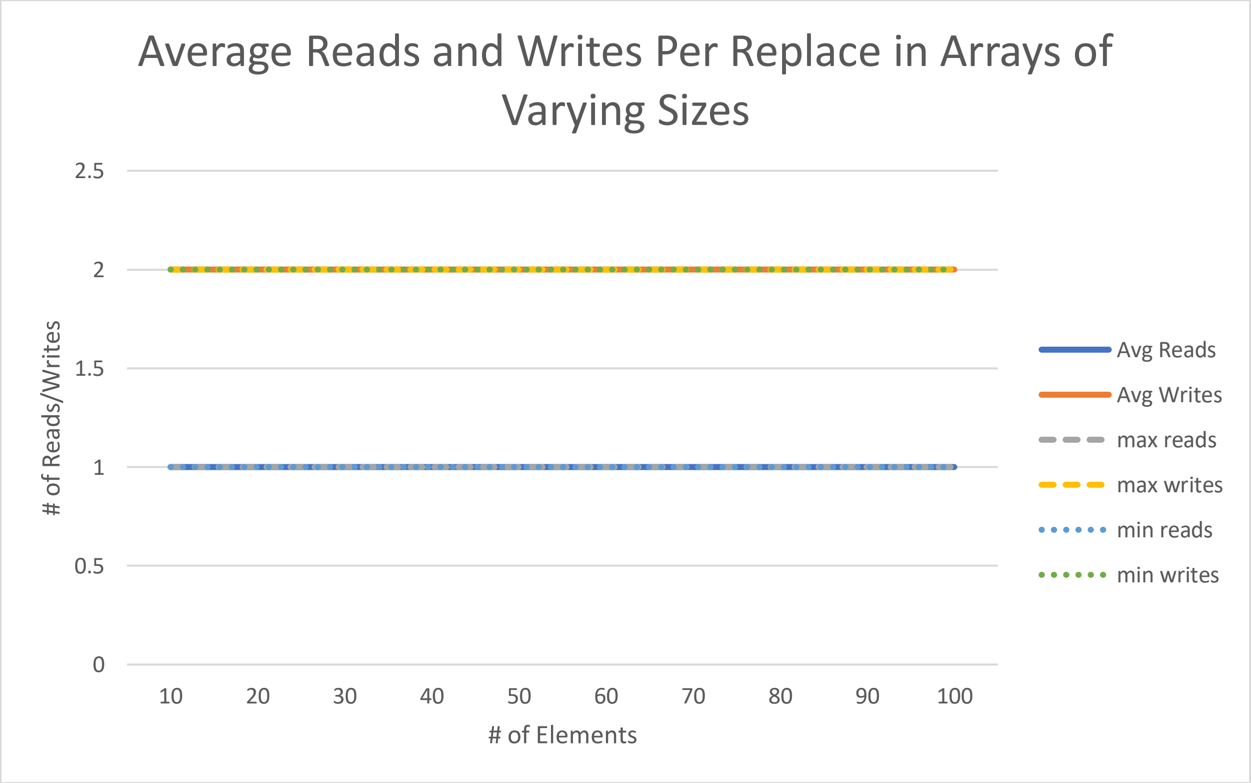
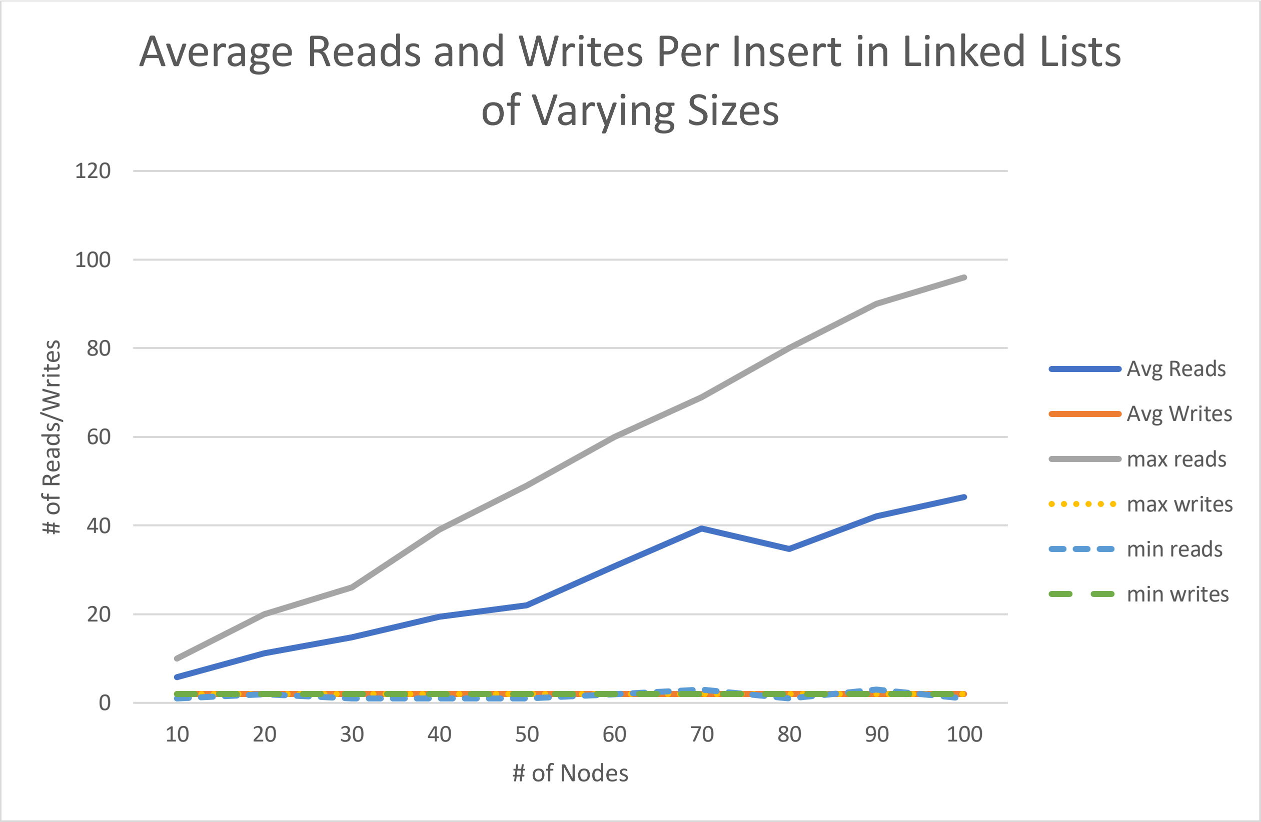
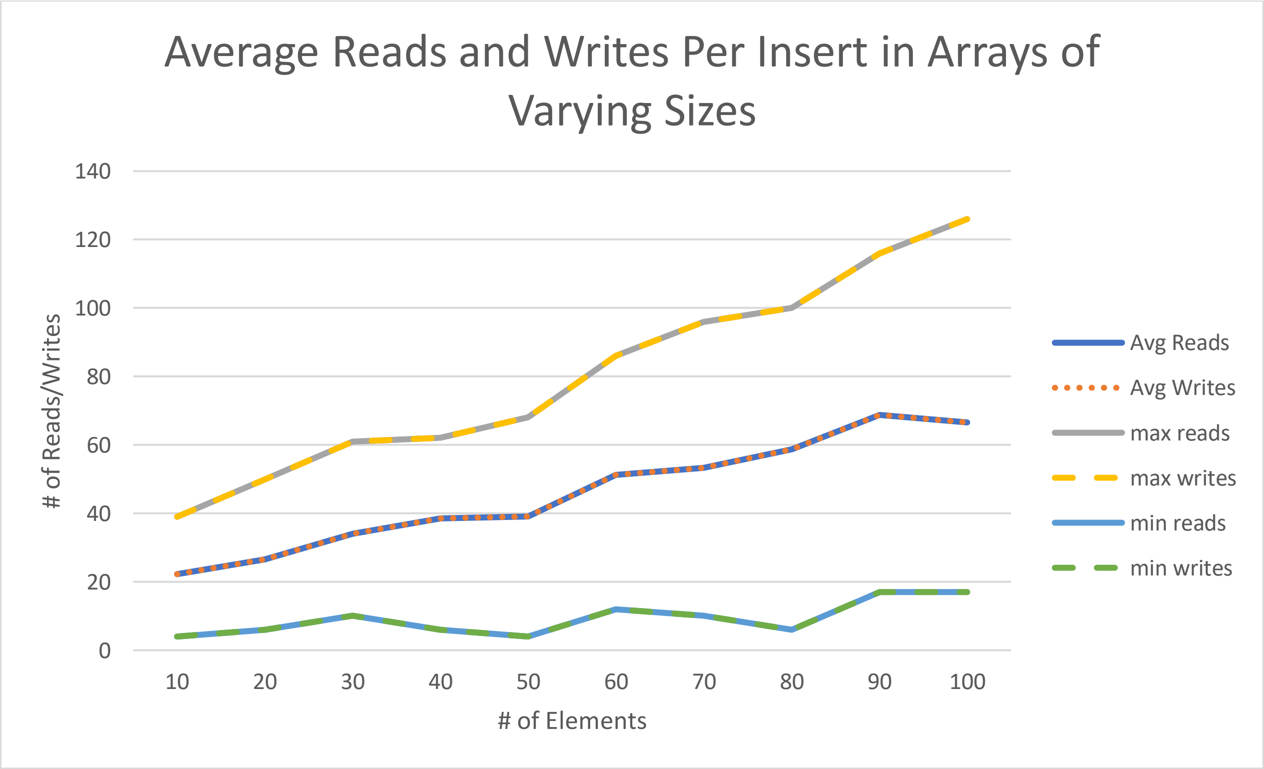
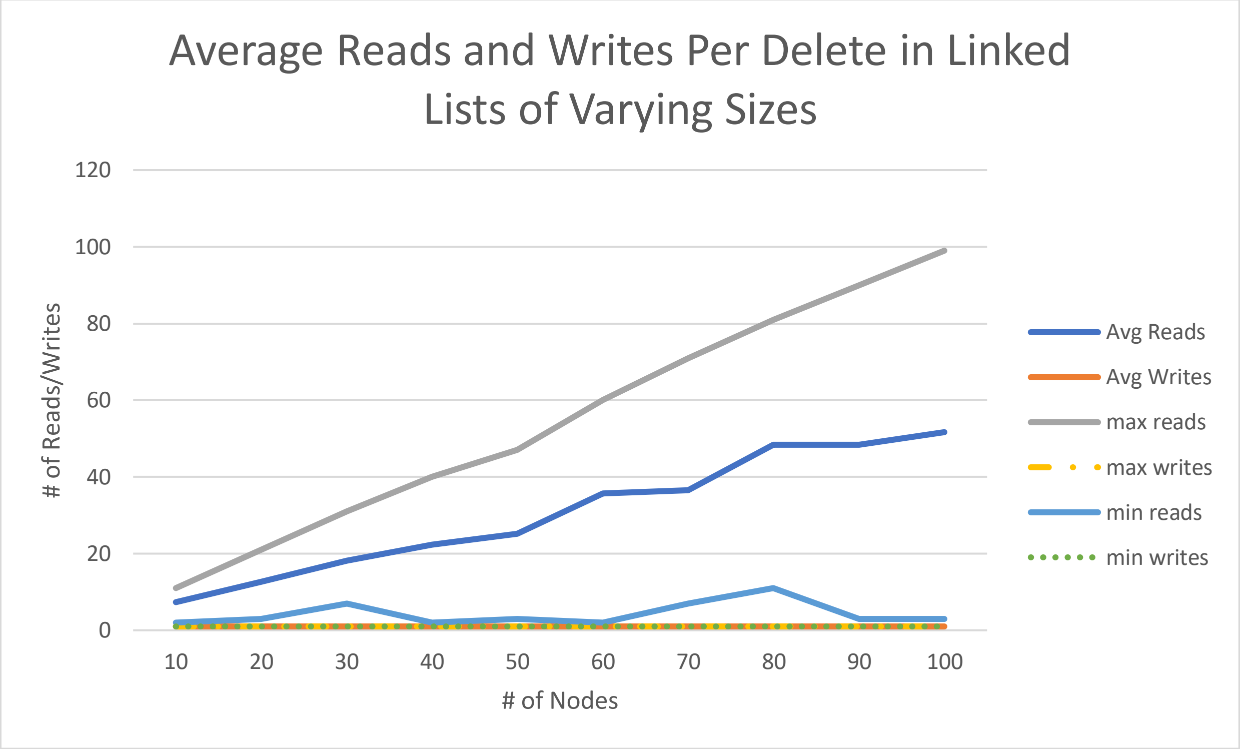

Average Reads and Writes Per Replace in Linked Lists of Varying Sizes
120
100
80
60
40
20
0
# of Reads/Writes
10
20
30
40
50
60
70
80
90
100
# of Nodes
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes

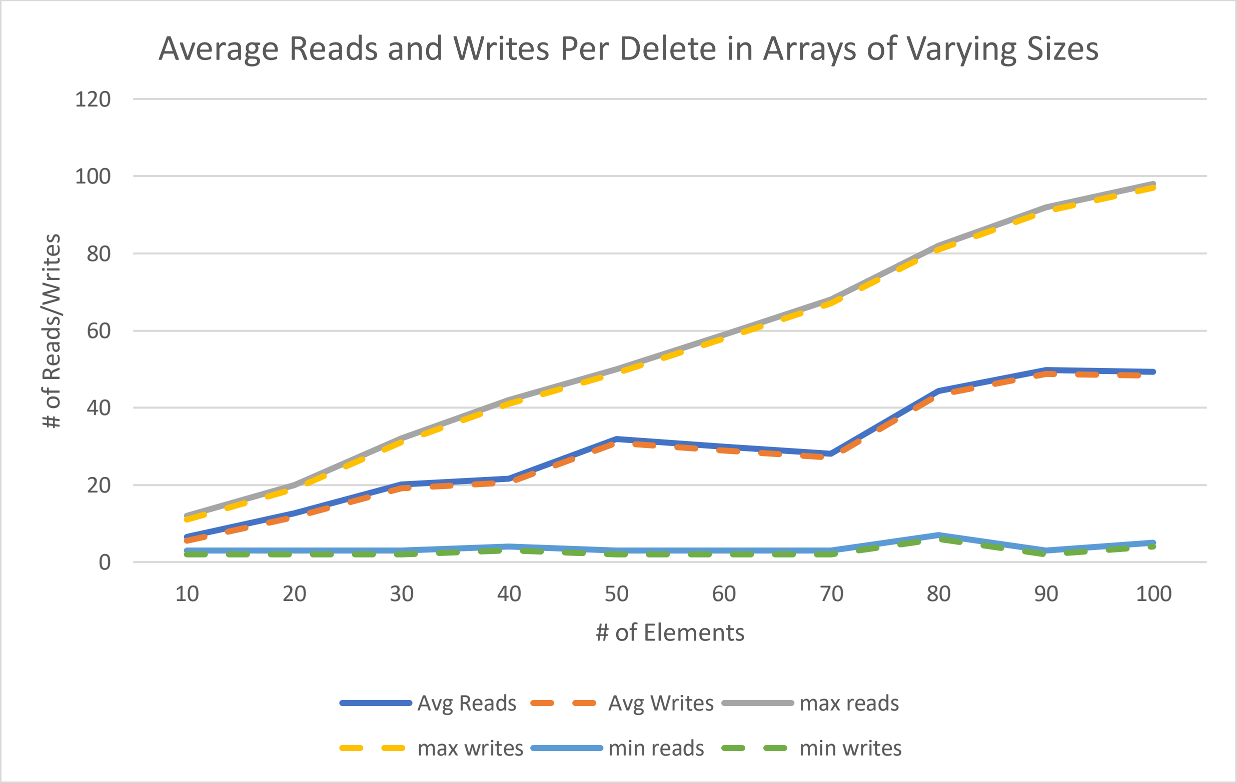

Average Reads and Writes Per Replace in Arrays of Varying Sizes
2.5
2
1.5
1
0.5
0
# of Reads/Writes
10
20
30
40
50
60
70
80
90
100
# of Elements
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes

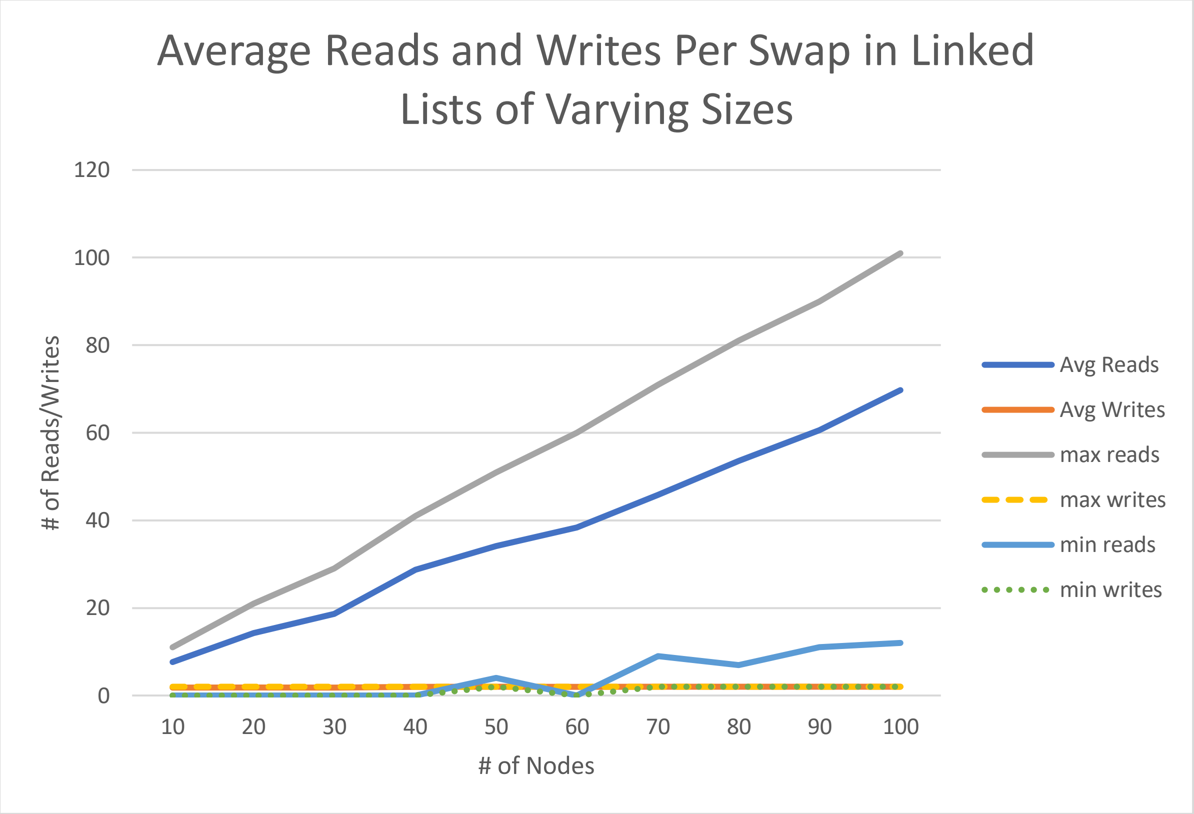

Average Reads and Writes Per Insert in Linked Lists of Varying Sizes
120
100
80
60
40
20
0
# of Reads/Writes
10
20
30
40
50
60
70
80
90
100
# of Nodes
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes

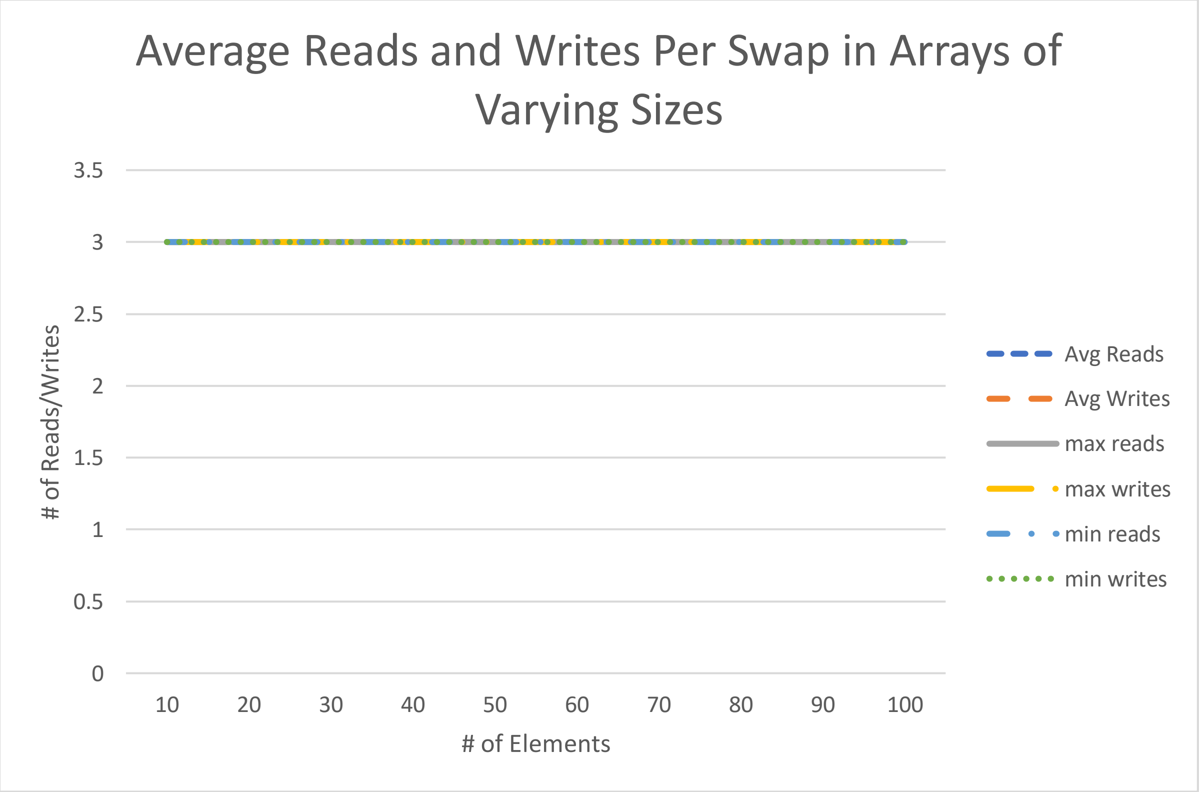

Average Reads and Writes Per Insert in Arrays of Varying Sizes
140
120
100
80
60
40
20
0
# of Reads/Writes
10
20
30
40
50
60
70
80
90
100
# of Elements
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes

Shawn Kaplan (0966499) Sept 29, 2019






Average Reads and Writes Per Swap in Linked Lists of Varying Sizes
# of Reads/Writes
120
100
80
60
40
20
0
10
20
30
40
50
60
70
80
90
100
# of Nodes
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes


Average Reads and Writes Per Swap in Arrays of Varying Sizes
# of Reads/Writes
3.5
3
2.5
2
1.5
1
0.5
0
10
20
30
40
50
60
70
80
90
100
# of Elements
Avg Reads
Avg Writes
max reads
max writes
min reads
min writes

Discussion

Based on the data shown above, arrays and linked lists have different strengths and weaknesses for a given type of task.

Replacing values within a data structure

While their best-case reads are very low, and the number of writes needed is constant, linked lists have a linearly scaling increase in the average number of reads needed to replace a value within the list. Meanwhile, arrays have a constant 2 writes and 1 read for any number of elements. This makes arrays far more viable in situations where values within the structure are frequently changed, such as a player's inventory in a videogame.

Inserting new values to a data structure

Linked lists have a linearly scaling increase in the number of reads needed to insert a new element to the structure, but a constant number of writes. Compared to arrays, which have both linear scaling reads and writes, this puts linked lists at an advantage because there are fewer total actions required to achieve the same goal. Cases where insertion is frequent and a linked list would be preferred include structures containing a list of users, where new users are must be added as they are entered into the system (i.e. create a new account).

Deleting elements in a data structure

Linked lists always write only 1 time to delete and have an average number of N/2 reads (where N is the number of nodes in the list) and a worst case of N reads. Meanwhile, arrays have an average of N/2 reads and writes with a worst case of N reads and writes. This means that linked lists will, in both average cases and worst cases, require fewer actions to perform the delete function. A linked list would be preferred over an array for implementing a stack or queue, as elements of the structure are expected to be deleted.

Swapping elements in a data structure

Linked lists always write twice when swapping 2 elements but have a linear increase in the number of reads correlating with the size of the list. Arrays have a constant 3 reads and 3 writes for an array of any size, making arrays preferable in situations where swapping elements happens a lot, such as in grid-based games, like chess, checkers, and Candy Crush.