

**PA5 Report**

By

**Grant Hedley**

**CSI 281**

**Data Structures & Algorithms**

**Spring 2019-20**

1. **Introduction**

The purpose of this experiment is to determine the efficiency of 3 different data structures (array, linked list and doubly linked list) by measuring how long it takes to complete comon tasks, suck as retiving and adding data.

1. **Background**

Linked Lists have the advantage of being resizable and a new element can quickly be added to them. The down side is you can only get to elements by traversing the entire list.

Doubly Linked Lits are the same as linked lists but they can be traversed in either direction.

Arrays can not be resized like Linked Lists but all of its elements are in adjacent order so that it can perform random access to allow quick access to specified index quickly.

1. **Experiment Plan**

the data structures will be graded on how fast they can compleet the following tasks

- make insertions (x100,000)

- retrive random elements (x1,000)

- traverse and print all elements

- delete all elements

1. **Experimentation Detail**

:

* 1. 16 GB
  2. Intel i7-5820K (12) @ 3.600GHz
  3. 3.4 GHz
  4. System type: 64 bits

1. **Discussion and Conclusion**

When it came to adding new elements, 100,000 elements were added to each structure and they were added so that the data structure was always sorted. The array was able to complete its task faster than either linked list at around 18 seconds. Due to the extra pointer for each node that needs to be adressed for the doubly linked list, it was a bit slower at adding new elements at around 42.5 seconds. The linked list was only a bit slower at around 38.5 seconds.

Retreving random elements is where arrays are supposed to shine, and they did. 1,000,000 get element calls were made on each data structure. Both the linked lists took only 3 seconds but the array took only 2 thousands of a second.

Traversing and printing the data was exactly the same for all 3 structures, about 1 second. However when doing it backwards the normal linked list took substantialy longer at 1 minute and 25 seconds, while the doubly linked list and array still took 1 second.

The last test was to delete all the data in the data structures, since the array was only in the stack and could not be resized or deleted all the data was just set to 0. The array was the fastest to compleete this at only 0.2 thousands of a second. The second fastest was the linked list at 3 thousands of a second. And the slowest was the doubly linked list at almost 1 second.

In all the array preformed the best or tied for the best in every category or was tied for best. The one area the array did not preform as well was how it could not actualy resize so it always took up the same amount of space. It was much larger than it needed to be when it was empty and it couldn’t hold anymore than how many elements it was originaly set to hold.

1. **Appendix**

Test LinkedList

make 100,000 insertions: 00:00:38.693067

retrive 1,000,000 random elements: 00:00:03.089765

traverse and print all elements: 00:00:00.945634

traverse and print all elements backwards: 00:01:25.630543

delete all elements: 00:00:00.003337

Test DoublyLinkedList

make 100,000 insertions: 00:00:42.492531

retrive 1,000,000 random elements: 00:00:03.105211

traverse and print all elements: 00:00:00.971958

traverse and print all elements backwards: 00:00:00.963697

delete all elements: 00:00:00.967756

Test Array

make 100,000 insertions: 00:00:18.210953

retrive 1,000,000 random elements: 00:00:00.002271

traverse and print all elements: 00:00:00.975868

traverse and print all elements backwards: 00:00:00.965865

delete all elements: 00:00:00.000211