CS3310 Report  
HW1 Searching in Arrays & Linked Lists

Western Michigan University

Instructor: Dr. Gupta

By Anfaal Faisal

**Specifications:**

**Main Goal of this program:**

Come up with a data structure to perfectly simulate an implementation of Searching in the linked list and Arrays

**Sub Goals:**

1. *Use various techniques to search arrays and linked lists*
2. *Practice developing high-performance solutions*
3. *Compare theoretical vs empirical complexities*
4. *Compare linear and binary search of arrays*
5. *Implement a doubly Linked List Implementation to hold and manipulate page classes and text file.*
6. *Report the value of n.*
7. *Evaluate output requirements (Average time spent searching, searched item's names, locations, strengths).*
8. *Sort the Arrays, according to bags which have 25 slots*
9. *Assign maxStrength, minStrength accordingly*
10. *Create Javadoc*
11. *Giving outputs of your program with n = 1, 10, 100, 1000, and 10000*

**Design:**

Methods/global variables List for class **Main**.java

static LinkedList<Weapon> *itemsLinkedList*;  
static Weapon[] *inventory*;

static Random *random* = new Random();

static FileWriter *fileWriter*;

static StringBuilder *stringBuilder*;

static void readFile()

static void fillInventory(int bagsNum)

static void search(long count, boolean isLinear)

static void printInventory()

static void sortByName(int fromIndex, int toIndex)

static void sortByCurrentStrength(int fromIndex, int toIndex)

static void multiMergeSort()

Methods/global variables List for class **Weapon**.java

private String itemName;  
private int minStrength;  
private int maxStrength;  
private int currentStrength;  
private String rarity;

Weapon(String itemName, String minStrength, String maxStrength, String rarity)

public String getItemName()

public int getMinStrength()

public int getMaxStrength()

public int getCurrentStrength()

public String getRarity()

public void setCurrentStrength(int currentStrength)

**EMPIRICAL AND THEORATICAL EVALUATION ALONGSIDE TIME COMPLEXITY GRAPH**

Empirically confirm the theoretical complexity of binary and linear search algorithms, and sorting algorithm.

In this task, the following algorithms were used, which are implemented in JDK: for search – binary search algorithm, linear search algorithm; for sorting – merge sort.

Difficulty for linear search – since you need to go through all elements of the array, the theoretical complexity of the algorithm will be O(n), for binary search – since with each pass the number of elements is reduced by two times the complexity of the algorithm will be O(log n), for merge sort – divides the array into two halves and take linear time to merge two halves its O(n log n) (for sort by name then by currentStrength 2 × O (n log n), for multi-merger sort - 4 × O (n log n).)

**Table 1 – Search (separately)**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of bags | Search name | | |
| Linear search(nanosec) | Binary search(nanosec) | |
| Search(nanosec) | Sort(nanosec) |
| 1 | 2961 | 4127 | 148876900 |
| 10 | 13628 | 2957 | 155973000 |
| 100 | 17003 | 2617 | 202521700 |
| 1000 | 25198 | 5528 | 197789600 |
| 10000 | 24193 | 5376 | 538539800 |

**Table 2 – Search (in a row)**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of bags | Search name | | |
| Linear search(nanosec) | Binary search | |
| Search(nanosec) | Sort(nanosec) |
| 1 | 12016 | 1819 | 585100 |
| 10 | 9462 | 877 | 1729400 |
| 100 | 13890 | 882 | 7613600 |
| 1000 | 5857 | 691 | 88447600 |
| 10000 | 5404 | 355 | 444888200 |

**Table 3 – Merge sort vs multi-merge sort (separately)**

|  |  |  |
| --- | --- | --- |
| Number of bags | Merge sort(nanosec) | Multi-merge sort(nanosec) |
| 1 | 307799200 | 179205500 |
| 10 | 173576700 | 227645400 |
| 100 | 179050200 | 176016500 |
| 1000 | 241774600 | 221077800 |
| 10000 | 634650500 | 805800100 |

**Table 4 – Merge sort vs multi-merge sort (in a row)**

|  |  |  |
| --- | --- | --- |
| Number of bags | Merge sort(nanosec) | Multi-merge sort(nanosec) |
| 1 | 156644500 | 585100 |
| 10 | 969200 | 1729400 |
| 100 | 9313300 | 7613600 |
| 1000 | 37904400 | 88447600 |
| 10000 | 737385500 | 444888200 |

As can be seen from the plots, the empirical plot is similar to the theoretical plot. This similarity means that the theoretical estimate was correct, which the plot obtained on the basis of empirically obtained data confirms.

Although the empirical merge sorting plot is not quite similar to the theoretical one, I believe that if you increase the number of bags, you can achieve similarities with the theoretical plot.