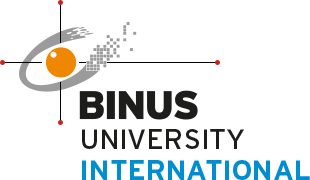
**ALGORITHM DESIGN & ANALYSIS  
FINAL REPORT**



Hansel Faren / 2501990350

Anisa Dzikra Qalbiah / 2502043116

William Jonathan Mulyadi / 2502045683

**TABLE CONTENTS**

Introduction & backgrounds ………………………………………………………………3

Problems ……………………………………………………………………………………3

Proposed solution ………………………………………………………………………….4

Measurement ……………………………………………………………………………….5

Theories ……………………………………………………………………………………..5

The Flow …………………………………………………………………………………….6

Result ………………………………………………………………………………………6-7

Conclusion & recommendation ………………………………………………………….8-9

References …………………………………………………………………………………10

Link to the code ……………………………………………………………………………10

Screenshots and user’s manual ……………………………………………………..11-14

**I. Introduction/Background**

This report is written for our Algorithm Design and Analysis course project that is called “Store Management Project”. This project is aimed to help companies that have bad item placement. This project is made by using the Python programming language. We also used pygame to create the Graphical User Interface to make the project more interesting.

**II. Problems**

The problem is that a company might have many items in its storage. However, the items are placed randomly. As the items are placed randomly, when an employee is asked to get a specific item, the employee will have to search and take a long time. This causes inefficiency in the company process thus, our aim is to help these companies to have a better storage management system. However, as there are many algorithms available, the company will then have to decide on which algorithms that suit them the best. Among many algorithms available, we are planning to use 4 most known algorithms and 1 algorithm that is known to be very slow. These 5 algorithms are

1. Bogo algorithm
2. Merge algorithm
3. Selection algorithm
4. Insertion algorithm
5. Bubble algorithm

With the problems stated we need to pick which one is the most efficient out of all the algorithms available. We will compare the algorithms with the measures that we decide.

**III. Proposed Solution**

We first created data that contains lots of lists of items that a company might have. We created it in an excel file format. We put the list of items alphabetically (sorted). After that, we will create several functions that help our program work. These functions works to

1. Randomize
2. Check whether the array is sorted or not
3. Sorting the list of items
4. Finding the storage

First, we will read the files from excel. After that, we call the function randoms() in order to randomize the file. As the items are placed randomly, the user can then sort it. The user can choose which of the sorting algorithms they want to use. As the sorting runs, the user can also see the runtime and the memory that the sorting algorithms take. From here, the user can list which of these algorithms are the slowest to the fastest. After the items are sorted, the user can search in which storage the items are located.

We created a GUI using pygame to enable the user to execute the code easier. There will be 6 pages:

1. The main page
2. The bogo page
3. The merge page
4. The selection page
5. The insertion page
6. The bubble page

On the main page, the user will be able to choose between the 5 algorithms that they want to use. The user will have to give input by pressing a keyword on the keyboard (will be shown in the documentation).

As they click on the keyword, they will be redirected to the sorting algorithm page and will find 3 instructions for keywords which functions to

1. Execute the sorting algorithm
2. Randomize the list
3. Search (for searching the storage that stores the item)

However, the list of items will be printed in the terminal and for the search function, the user will have to give input in the terminal.

**IV. Measurement**

By choosing algorithms in sorting algorithms, the significant difference we can make is by looking at how long the algorithm took to sort an array, which is the algorithm’s running time, and how much the amount of memory cost of a sorting algorithm. By knowing these 2 components, one can make a conclusion about which algorithm works more efficiently rather than the others.

There are 5 sorting algorithms that we are going to use. Between these sorting algorithms, we will try to find which sorting algorithms will be the best use in a certain array that we have created. The sorting algorithms that we are going to use are Merge Sort, Bubble Sort, Insertion Sort, Selection Sort, and Bogo Sort.

**V. Theories**

1. Bogo Sort

This sorting algorithm shuffles the array and checks if the array is sorted or not. If it is still not sorted, it will continue randomly shuffling the array until the array is sorted. This can be the fastest sorting algorithm if the array is sorted on the first try. Worst time complexity: O((n+1)!)

1. Merge Sort

A sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner. Worst time complexity: O(n log n)

1. Bubble Sort

is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order. Just like the movement of air bubbles in the water that rise up to the surface, each element of the array moves to the end in each iteration. Therefore, it is called a bubble sort. Worst time complexity: O(n²)

1. Insertion Sort

A simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part. Worst time complexity: O(n²)

1. Selection Sort

A sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list. Worst time complexity: O(n²)

**VI. The Flow**

First we will do research on the sorting algorithm implementation. Then, we will create a csv-file that contains several lists of items that are sorted. After that, we will make a program that randomized the items in the csv file that we made. After the items in the csv file is randomized (unsorted), we will implement the algorithms that we mentioned before. We will also put functions to calculate the running time and the memory cost of the algorithm. Finally, we will conclude the result and find out which algorithm works best with the scenario.

**VII. Results**

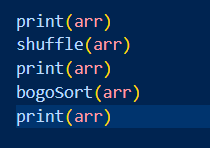
**Note: it is randomize before sorted hence, the cases conditions are unknown. (the case might not be the best case or the worst case)**

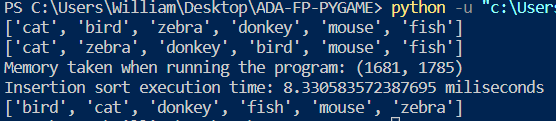
The bogo sorting algorithm takes too long to process the list of items in the file that we made (as there are about 129 items). Hence, we made a special array and code for Bogo in order to help us conclude:

* First we create a new array that consists of 7 strings:



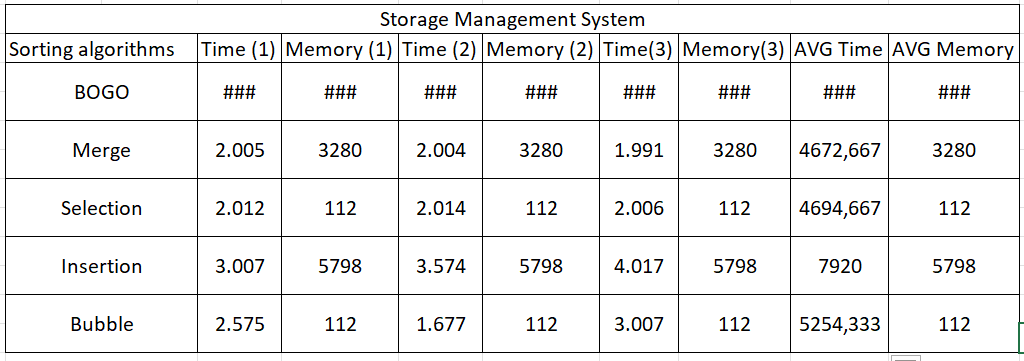
* Then we call the functions:

****

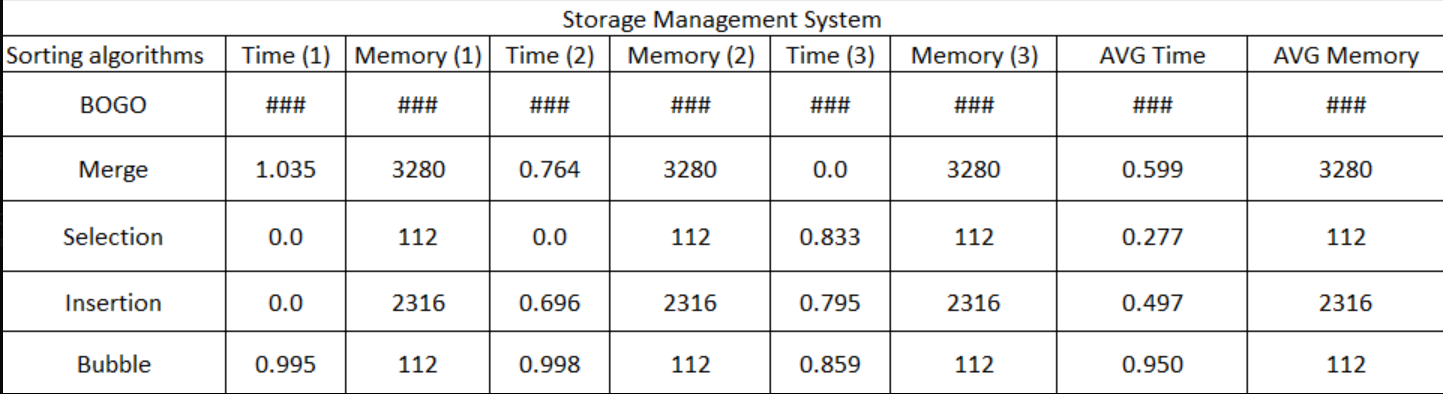
* The result of the functions:****

As you can see, it takes 1785 memory and 8.33 milliseconds just to sort 7 strings.

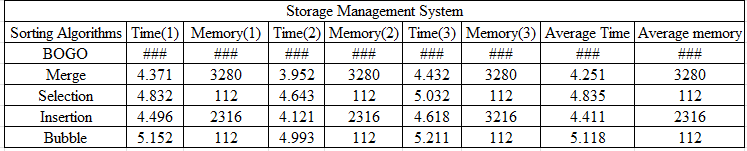
From computer A:



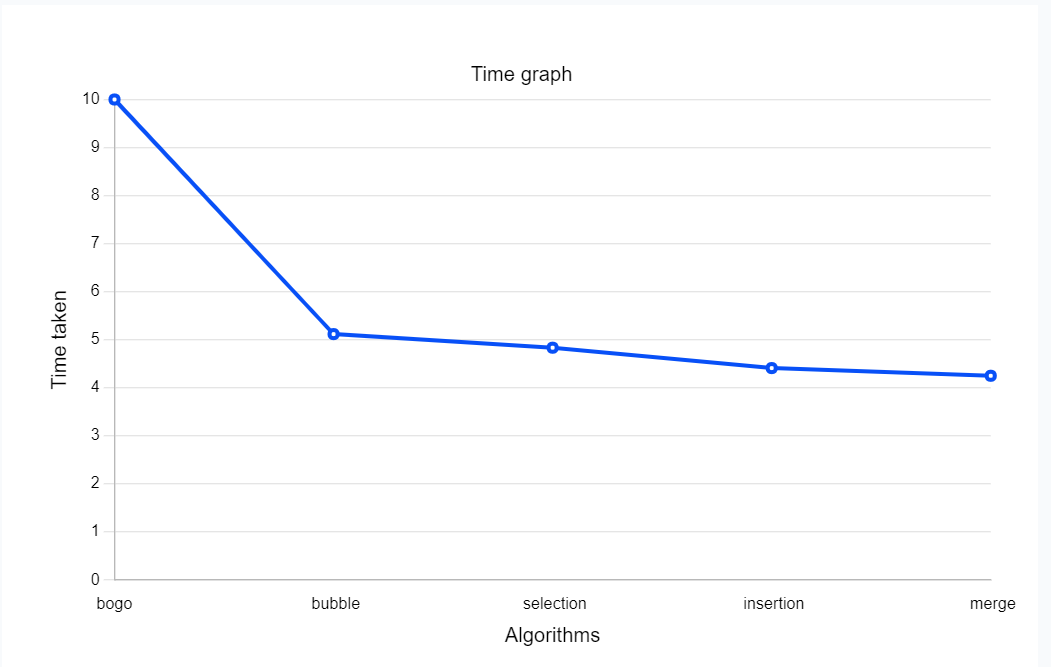
From computer B:



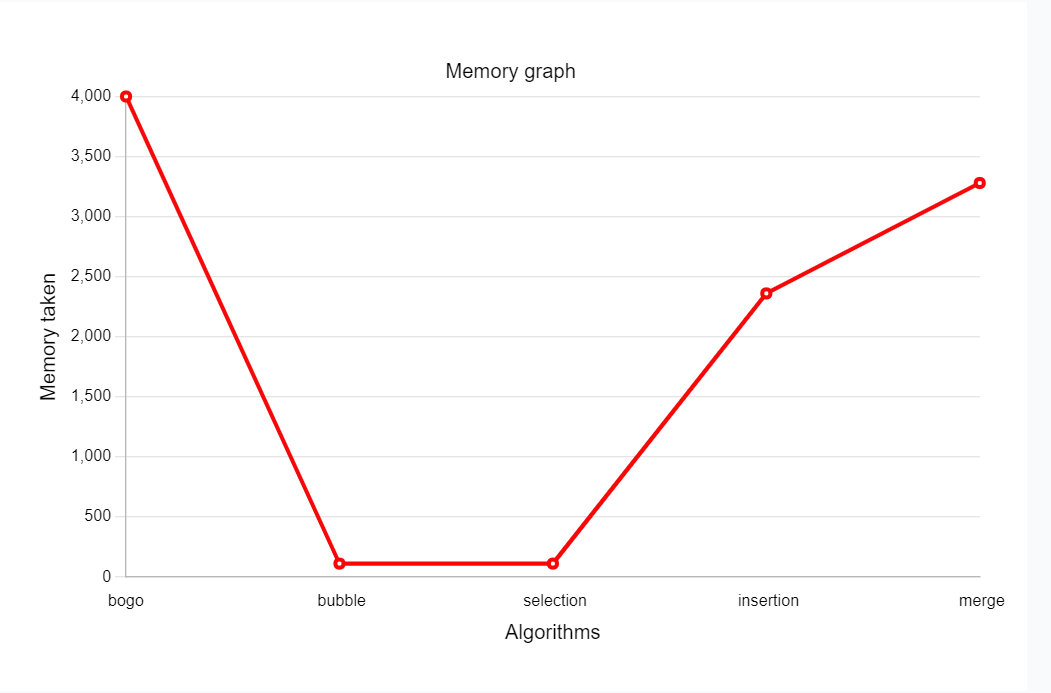
Optimum result after trying several times:



TIME GRAPH:



Memory graph:



**VIII. Conclusion and Recommendation**

Based on the above results, we can take a conclusion on which is the fastest and the slowest sorting algorithm by using an array that we make. The slowest sorting algorithm is bogo algorithm. As you can see from the result, to sort 7 strings, it takes longer compared to sorting 129 strings in other algorithms. The bogo sorting algorithm is the most inefficient compared to the 4 other algorithms. Besides that, it takes more memory compared to the other algorithm.

As for the time, the second slowest one is bubble sort, followed by selection sort. The fourth slowest one is insertion sort and the fastest one is merge sort. Meanwhile, for the memories taken, the biggest memory intake after bogo is merge sor,t followed by insertion sort. The bubble and selection sort take approximately the same which is 112.

Time efficiency from the slowest to the fastest:

Bogo → bubble → selection → insertion → merge

Memory intake from the biggest to the smallest:

Bogo → merge → insertion → bubble, selection

**RECOMMENDATION**

To conclude, the company should never use bogo sorting algorithm as it works really bad and can negatively affect the company efficiency. If the company search for the algorithm that works the fastest, the company should use merge sorting algorithm. On the other hand, if the company wants to search for the algorithms that take up the smallest amount of the memory space, the company should use bubble and selection sorting algorithm. The one that is most optimum in memory and time-wise would be the **selection sort**. As it ranks first in the memory and ranks third in the time taken.

**IX. References**

*Bubble Sort (With Code in Python/C++/Java/C)*. (n.d.). <https://www.programiz.com/dsa/bubble-sort>

Finn, D. (2021, December 9). *The Worst and Most Entertaining Sorting Algorithms - Dylan Finn*. Medium. <https://medium.com/@dylan.finn/the-worst-and-most-entertaining-sorting-algorithms-96efc9025f7>

GeeksforGeeks. (2022b, October 13). *BogoSort or Permutation Sort*. <https://www.geeksforgeeks.org/bogosort-permutation-sort/>

GeeksforGeeks. (2022c, October 18). *Insertion Sort*. <https://www.geeksforgeeks.org/insertion-sort/>

GeeksforGeeks. (2022e, November 15). *Merge Sort Algorithm*. <https://www.geeksforgeeks.org/merge-sort/>

Kiao, U., PhD. (n.d.). *Bogosort vs Bozosort [Differences explained]*. OpenGenus IQ: Computing Expertise & Legacy. <https://iq.opengenus.org/bogosort-vs-bozosort/>

*Selection Sort (With Code in Python/C++/Java/C)*. (n.d.). <https://www.programiz.com/dsa/selection-sort>

Wikipedia contributors. (2022, November 21). *Bogosort*. Wikipedia. <https://en.m.wikipedia.org/wiki/Bogosort>

**X. Links**

The link to the Github repository: <https://github.com/willamjonathan/FP_Algo-Design-Analysis>

The link contains our files and code.

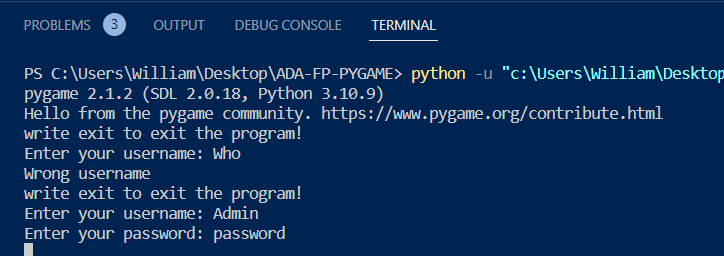
**XI. Screenshots and User Manual**

**Reminder: the bogo sort will take a very long time, it might even cause the system to crash, hence be patient! Or run it after you try the other algorithms. (high probability of causing a crash)**

**RUN THE FILE, runpage.py!**

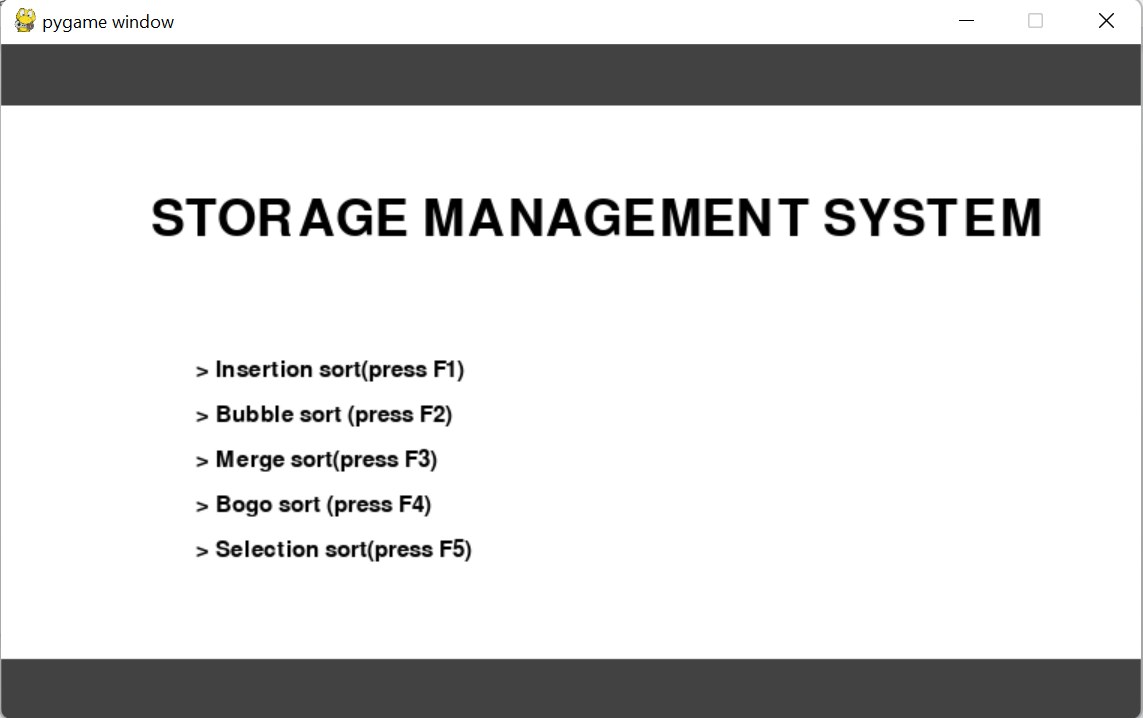
Please download the file from the github repo.

The system’s security:

****

The username must be “Admin” and the password must be “password”. Other than that, the system will not be able to work. We put this as our system security.

As you have logged in, the main page will pop up:

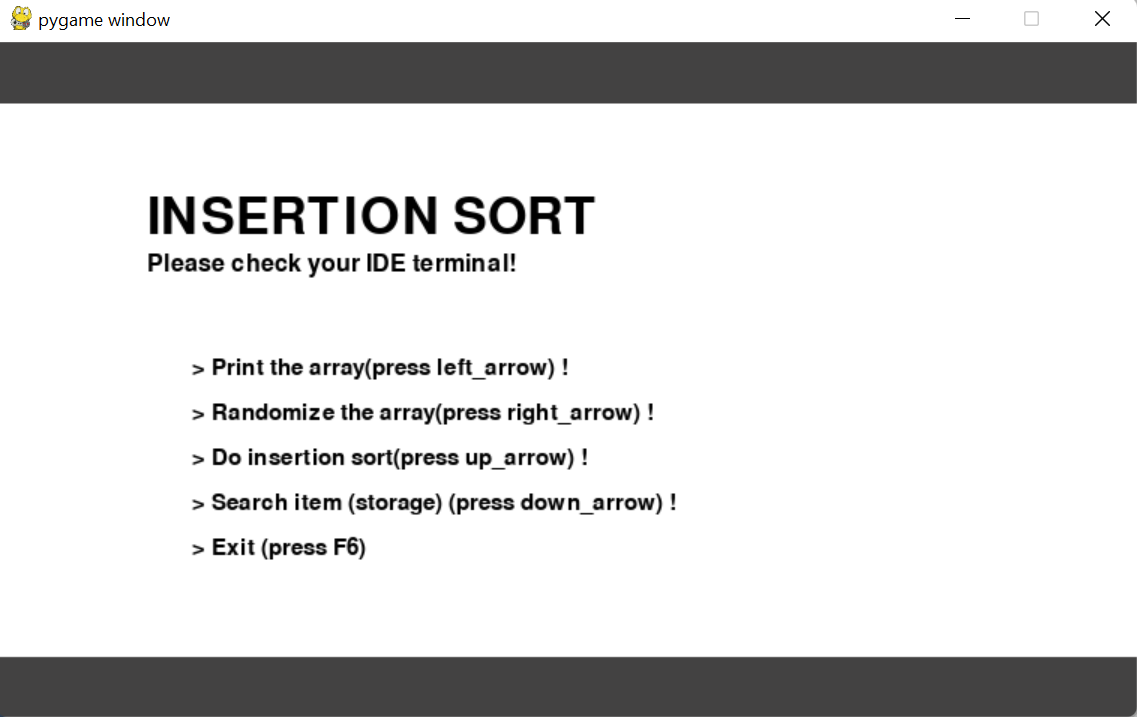


In the mainpage, there are manuals that the user can follow. The user is able to choose between the sorting algorithms available. As mentioned before, we used 5 algorithms. The user is able to use the sorting algorithm only if they press the right keyword.

1. F1 is for insertion sort
2. F2 is for bubble sort
3. F3 is for merge sort
4. F4 is for bogo sort
5. F5 is for selection sort

As the user press the keyword, it will redirect them to the page of the sorting algorithm.

As the user clicks F1, the insertion sort page will pop up:



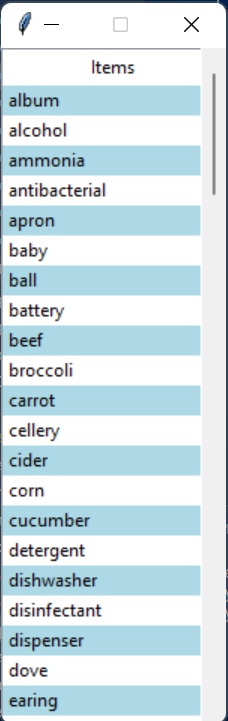
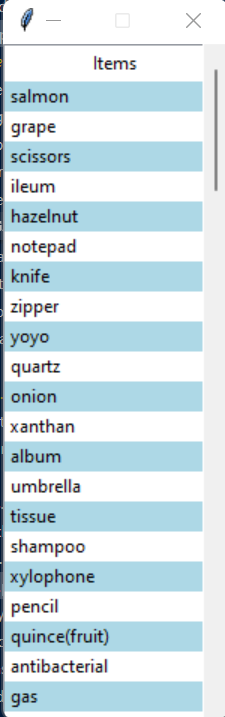
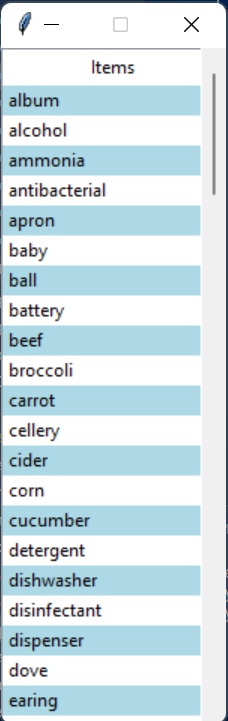
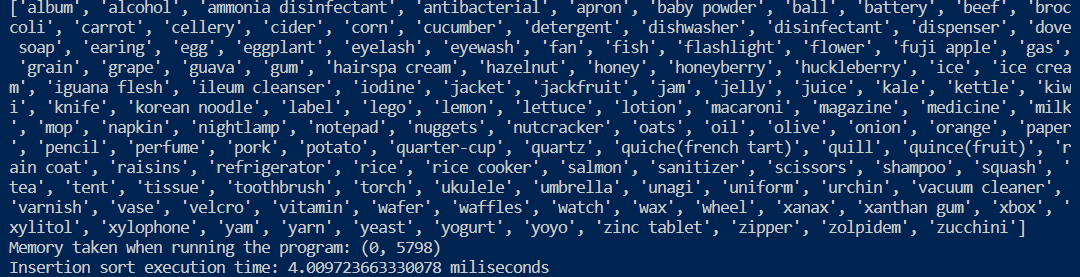
There are other sets of manual that helps the user to understand how to operate the program. Please do not forget to check your IDE terminal!

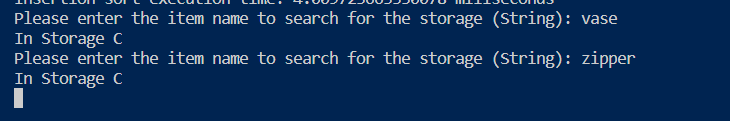
Fig 1 Fig 2 Fig 3

Fig 1 shows the window when the print function is called, Fig 2 shows the randomize function. Meanwhile, Fig 3 shows the window after sorting the Fig 2 and a print function is being called.



As you can see, the memory and time taken are also printed in the terminal.

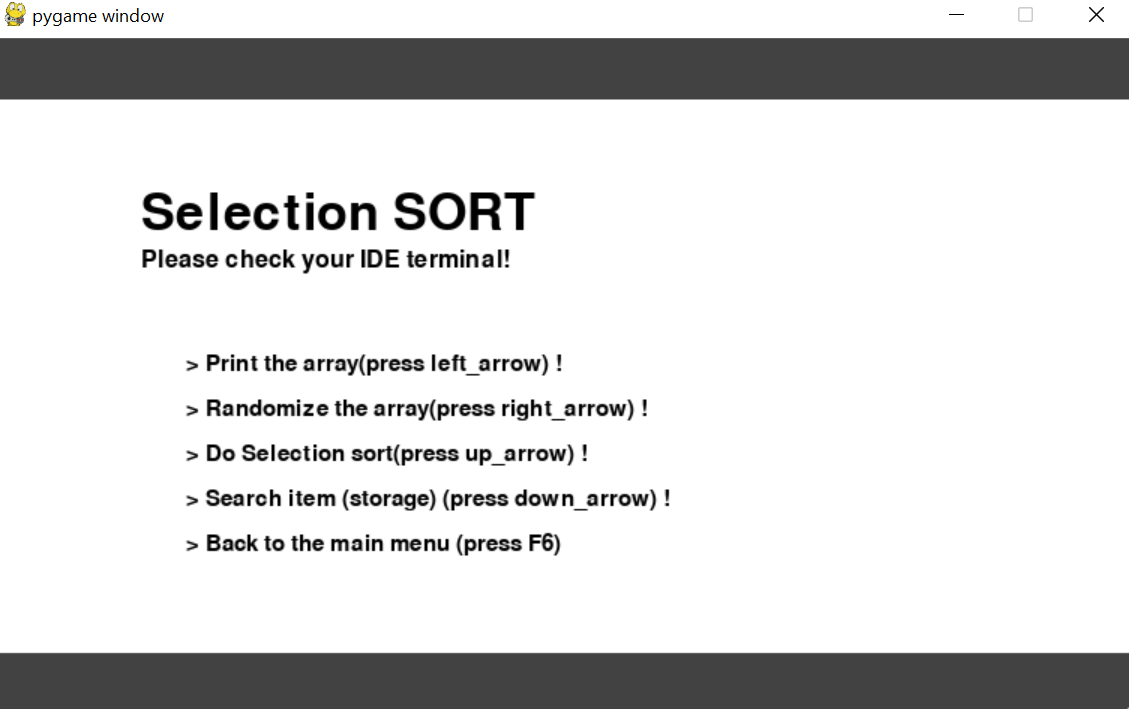
Search item’s location:



And to go back to the main page, the user can click F6.

For the other algorithms, the manuals are the same. Here are the screenshots of the other 4 algorithms:

1. Selection SORT



2. BOGO SORT



3. MERGE SORT

4. BUBBLE SORT