**Title:**

**Scientific Report of Sorting and Searching Algorithms**

**Introduction:**

The aim of this report is to give feedback on the algorithms’ efficiency on sorting numerical lists. The sort selection method was compared with the sort merge method. The comparison was done using time as factor to determine which was more efficient. The list size was made constant for each experiment, so that it wouldn’t render the results inaccurate.

**Description of Algorithms:**

**Selection Sort Algorithm**

The basic principle with which selection sort uses is, on the position of the first item in list, it swaps it with the smallest value in the list. Then continues doing the same principle but starting in the second position until the position is the last position in the list hence resulting the list being sorted. Looking at the algorithm mathematically. The selection method sorting algorithm taking n as size of the list makes n + (n-1) + (n-2) + ... + 3 + 2 + 1 = n(n + 1) / 2 comparisons and n exchanges. This can be simplified into mathematical expressions comparisons resulting n(n+1)/2 and exchanges being n. The code for selection sort is perceived simple.

**Merge sort Algorithm**

The basic principle of the merge sort method is to split the list into pairs and compare each pair, taking the bigger value of the pair placing it on the first position so that each pair is sorted and combined with another pair hence list is sorted. The mathematical expression for this n log(base2)n, for a list of 8 there will be 3 merges and if 32 there will be 5 merges. The code for merge sort is perceived as intricate.

**Hypothesis**

The merge sort will be more efficient than the selection sort meaning if we have a list of 2000 or any size, the time it takes for merge sort to sort the list will be less than the selection sort**.**

**Experiment steps**

The steps are

1. Prompt the user to enter size list
2. Choose sorting type method
3. Time the time that it takes for each sort method to sort list.
4. Repeat the steps 3 times for each data size keeping list size constant
5. Take the average of those three values
6. Create a table for your data
7. Create another table for averages

**Analysis**

Analyzing three sorting methods’ data, namely bubble, selection and merge it cannot be mistaken that merge comes out on top with less than a second to sort numerical list of integers greater than 7000. The bubble sorting method, proved to be the less efficient because it took much longer than selection and merge methods to sort the list. The most useful algorithm to getting the sorting job fast is the merge method. The selection sort isn’t that bad also though it lacks with bigger data size, one would need to use a more powerful processor for selection sort to be more efficient.

**Conclusion**

The results validate the hypothesis made above and though there is uncertainty in the instantaneous speed of CPU it’s negligible since it has minimum effect.

**References**

[**https://vula.uct.ac.za/access/content/group/f5e9a0e6-f133-421b-8d58-485d2a0864ef/Extra%20Files/sortsearch.py**](https://vula.uct.ac.za/access/content/group/f5e9a0e6-f133-421b-8d58-485d2a0864ef/Extra%20Files/sortsearch.py)

**Table of contents**

1. **Introduction – briefly describing what this report is about**
2. **Description – briefly describe each algorithm**
3. **Hypothesis – statement indicating the expected result**
4. **Experiment – briefly describing the experiment steps**
5. **Results – use tables and graphs using Microsoft Excel**
6. **Analysis – comparing the results and making recommendations about which algorithm is more “useful” and for what reason.**
7. **Conclusion – briefly summarize and conclude**
8. **References – list of references in the proper format**

**CSC1010H**

**EXPERIMENTAL COMPUTER SCIENCE**

**SORTING ALGOTHIMS AND METHODS**