REQUIREMENTS

INTRODUCTION

Sorting techniques have a wide variety of applications. Computer-Aided Engineering systems often use sorting algorithms to help reason about geometric objects, process numerical data, rearrange lists, etc. In general, therefore, we will be interested in sorting a set of records containing keys, so that the keys are ordered according to some well-defined ordering rule, such as numerical or alphabetical order. Often, the keys will form only a small part of the record. In such cases, it will usually be more efficient to sort a list of keys without physically rearranging the records.

OBJECTIVER OF THE PROJECT

This deals with the presentation of information or data in an appreciable form and also to minimize the time used in searching for items. However, it can be seen that a well-sorted or organized file enhances easy searching of data while an unsorted one will pose little or great problems to locate a given item in a large list. However, the comparison process will be based on the following:

(1) Memory space used by methods

(2) Different techniques used for sorting

(3) Number of comparisons made during sorting process

(4) Coding based on the sorting algorithm

Signification of the Project

designed Sorting algorithm was to enable the people and the society to be acquainted with the arrangement of data and items. Above topic will let us know the organization of data in the memory location and also make proper use and utilization of the computer time.

Benefits of Sorting Techniques

The advantage of an algorithm is that it can provide a method to follow when attempting to solve a real problem.

The other common form of problem-solving is "trial and error", which is often much more inefficient than

- studying the problem,
- looking for an algorithm that can solve that problem,
- then use the algorithm to build a concrete solution.

Performance Criteria

There are several criteria to be used in evaluating a sorting algorithm:

- Running time: - Typically, an elementary sorting algorithm requires O(N^2) steps to sort N randomly arranged items. More sophisticated sorting algorithms require O(N log N) steps on average. Algorithms differ in the constant that appears in front of the N^2 or N log N. Furthermore, some sorting algorithms are more sensitive to the nature of the input than others. Quicksort, for example, requires O(N log N) time in the average case, but requires O(N^2) time in the worst case.
- Memory requirements: - The amount of extra memory required by a sorting algorithm is also an important consideration. In place sorting algorithms are the most memory efficient, since they require practically no additional memory. Linked list representations require an additional N words of memory for a list of pointers. Still other algorithms require sufficient memory for another copy of the input array. These are the most inefficient in terms of memory usage.

- Stability: - This is the ability of a sorting algorithm to preserve the relative order of equal keys in a file.

Benefits of Sorting Techniques

The advantage of an algorithm is that it can provide a method to follow when attempting to solve a real problem.

The other common form of problem-solving is "trial and error", which is often much more inefficient than

- studying the problem,
- looking for an algorithm that can solve that problem,
- then use the algorithm to build a concrete solution.

Performance Criteria

There are several criteria to be used in evaluating a sorting algorithm:

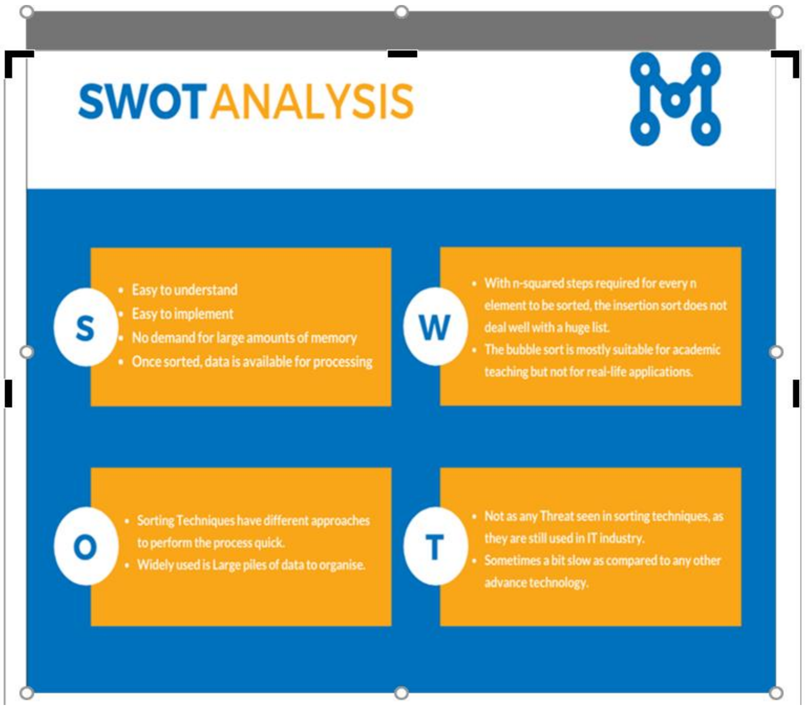
- Running time: - Typically, an elementary sorting algorithm requires $O(N^2)$ steps to sort N randomly arranged items. More sophisticated sorting algorithms require O(N log N) steps on average. Algorithms differ in the constant that appears in front of the N^2 or N log N. Furthermore, some sorting algorithms are more sensitive to the nature of the input than others. Quicksort, for example, requires O(N log N) time in the average case, but requires $O(N^2)$ time in the worst case.
- Memory requirements: - The amount of extra memory required by a sorting algorithm is also an important consideration. In place sorting algorithms are the most memory efficient, since they require practically no additional memory. Linked list representations require an additional N words of memory for a list of pointers. Still other algorithms require sufficient

memory for another copy of the input array. These are the most inefficient in terms of memory usage.

- Stability: - This is the ability of a sorting algorithm to preserve the relative order of equal keys in a file.

SWOT Analysis

Below is the SWOT Analysis of different sorting algorithms:



SWOTANALYSIS

S
- Easy to understand
- Easy to implement
- No demand for large amounts of memory
- Once sorted, data is available for processing

W
- With n-squared steps required for every n element to be sorted, the insertion sort does not deal well with a huge list.
- The bubble sort is mostly suitable for academic teaching but not for real-life applications.

O
- Sorting Techniques have different approaches to perform the process quick.
- Widely used is Large piles of data to organise.

T
- Not as any Threat seen in sorting techniques, as they are still used in IT industry.
- Sometimes a bit slow as compared to any other advance technology.

Detail requirements

High Level Requirements:

ID	Description	Category	Status
H_SA01	User should be able to give input of their choice	Technical	To Be Implemented

H_SA02	User should be able to input array of their choice	Technical	To Be Implemented
H_SA03	User should be able to perform bubble sort	Technical	To Be Implemented
H_SA04	User should be able to perform insertion sort	Technical	To Be Implemented
H_SA05	User should be able to perform quick sort	Technical	To Be Implemented
H_SA06	User should be able to perform merge sort	Technical	To Be Implemented

Low Level Requirements:

ID	**Description**	**HLR ID**	**Status**
L_SA01	User should be able to give array input for the bubble sort technique	H_SA02, H_SA03	To Be Implemented
L_SA02	User should be able to give array input for the insertion sort technique	H_SA02, H_SA04	To Be Implemented
L_SA03	User should be able to give array input for the quick sort technique	H_SA02, H_SA05	To Be Implemented
L_SA04	User should be able to give array input for the merge sort technique	H_SA02, H_SA06	To Be Implemented