TOPIC: SORTING

Introduction

A sorting algorithm is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order. Efficient sorting is important to optimizing the use of other algorithms (such as search and merge algorithms) that require sorted lists to work correctly; it is also often useful for canonicalizing data and for producing human-readable output.

Sorting is any process of arranging items in some sequence and/or in different sets, and accordingly, it has two common, yet distinct meanings:

1. Ordering: arranging items of the same kind, class, nature, etc. in some ordered sequence,
2. Categorizing: grouping and labelling items with similar properties together (by sorts).

Concept

Several types of sorting methods.

- **Selection Sort**
  Selection sort is a sorting algorithm, specifically an in-place comparison sort. Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations.

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>100</td>
<td>8</td>
<td>2</td>
<td>15</td>
</tr>
</tbody>
</table>

i) 2 is the lowest value
No. 2 exchange to the first place, no. 10 to the fifth place

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

ii) 5 is the lowest value
No. 5 remain in the second place

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>100</td>
<td>8</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

iii) 8 is the lowest value
No. 8 exchange place with no. 100

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>100</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

iv) 10 is the lowest value
No. 10 exchange place with no. 100

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>100</td>
<td>15</td>
</tr>
</tbody>
</table>

v) 15 is the lowest value
No. 15 exchange place with no. 100

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>15</td>
<td>100</td>
</tr>
</tbody>
</table>

vi) Ordered list result from the selection sort method
```
• **Insertion Sort**

Insertion sort is a simple sorting algorithm, a comparison sort in which the sorted array (or list) is built one entry at a time. It is much less efficient on large lists than more advanced algorithms.

![Insertion Sort Diagram]

• **Bubble Sort**

Bubble sort is a simple sorting algorithm. It works by repeatedly stepping through the list to be sorted, comparing two items at a time and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort.

![Bubble Sort Diagram]
35 exchange with 25

35 and 65 remain unchanged

65 exchange with 55

65 and 70 remain unchanged

70 and 85 remain unchanged

List resulted from the second loop

25 and 35 remain unchanged

35 and 55 remain unchanged

55 and 65 remain unchanged

65 and 70 remain unchanged

70 and 85 remain unchanged

List resulted from the last loop
- **Merge Sort**
  Merge sort is a comparison-based sorting algorithm. In most implementations it is stable, meaning that it preserves the input order of equal elements in the sorted output. It is an example of the divide and conquer algorithmic paradigm. It was invented by John von Neumann in 1945.

  Conceptually, a merge sort works as follows:
  1. If the list is of length 0 or 1, then it is already sorted. Otherwise:
  2. Divide the unsorted list into two sub lists of about half the size.
  3. Sort each sub list recursively by re-applying merge sort.
  4. Merge the two sub lists back into one sorted list.

  Merge sort incorporates two main ideas to improve its runtime:
  1. A small list will take fewer steps to sort than a large list.
  2. Fewer steps are required to construct a sorted list from two sorted lists than two unsorted lists.
  For example, you only have to traverse each list once if they’re already sorted (see the merge function below for an example implementation).
Quick Sort
As one of the more advanced sorting algorithms, you might think that the Quick sort Algorithm is steeped in complicated theoretical background, but this is not so. Like Insertion Sort, this algorithm has a fairly simple concept at the core, but is made complicated by the constraints of the array structure.

The basic concept is to pick one of the elements in the array as a pivot value around which the other elements will be rearranged. Everything less than the pivot is moved left of the pivot (into the left partition). Similarly, everything greater than the pivot goes into the right partition. At this point each partition is recursively quick sorted.

The Quick sort algorithm is fastest when the median of the array is chosen as the pivot value. That is because the resulting partitions are of very similar size. Each partition splits itself in two and thus the base case is reached very quickly.

In practice, the Quick sort algorithm becomes very slow when the array passed to it is already close to being sorted. Because there is no efficient way for the computer to find the median element to use as the pivot, the first element of the array is used as the pivot. So when the array is almost sorted, Quick sort doesn't partition it equally.
Activity

1. Given is a list of numbers 23  78  45  8  32  56. Sort the list (ascending) using:
   a) Selection Sort
   b) Insertion Sort

2. Write down all steps to sort below sequences of number using bubble sort.
   11  12  10  14  13
Introduction

One of the most common and time-consuming operations in computer science is searching, the process used to find the location of a target among a list of objects. Searching process in data processing context is a process to find the location of data in a list or table given. It’s being done by comparing each data item with the search key.

Concept

Several types of searching methods.

- **Linear Search**
  
  The linear Search is used whenever the list is not ordered. Generally, linear search are used only for small lists or lists that are not searched often. In other cases, we have to sort the list and then search it using the binary search.

  In linear search, we start searching for the target at the beginning of the list and continue until we find the target or we are sure that it is not in the list. This approach gives us two possibilities: either we find it or we reach the end of the list.

  The linear search algorithm needs to tell the calling algorithm two things: First, did it find the data it was looking for? And second, if it did, at what index are the target found?

  Let say, we want to search whether 14 is in the list.

  ![Diagram](image.png)

  **Successful!! Search found.**
**Binary Search**

The linear search algorithm is very slow. If we have an array of 1000 elements, we must do comparisons in the worst case. If the array is not sorted, the linear search is the only solution. However, if the array is sorted, we can use a more efficient algorithm called the binary search. We should use a binary search whenever the list starts to become large.

The binary search starts by testing the data in the element at the middle of the array to determine if the target is in the first or second half of the list. If it is in the first half, we do not need to check the second half. If it is in the second half, we do not need to test the first half. In other words, we eliminate half the list from further consideration.

Let say, we want to search whether 21 is in the list.

![Binary Search Diagram]

Successful!! Search found.

**Hashed Search**

A hash search is a search in which the key, through an algorithmic function, determines the location of the data. We use a hashing algorithm to transform the key into the index that contains the data we need to locate. Another way to describe hashing is as a key-to-address transformation in which the keys map to addresses in a list.

Generally, the population of keys for a hashed list is greater than the storage area for the data. For example, if we have an array of 50 students for a class in which the students are identified by the last four digits of their ICNo, then there are 200 possible keys for each element in the array. Because there are many keys for each index location in the array, more than one student may hash to the same location in the array.
Assessment

Q1. Sorting is
   A. A process of arranging data in some sequence whether ascending or descending by letter or number.
   B. Noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations.
   C. A fairly simple concept at the core, but is made complicated by the constraints of the array structure.
   D. A data structure that consists of a parallels of data records such that in each record there is a field that contains a reference.

Q2. List down 3 methods of sorting.

Q3. List down 3 types of searching method.

Q4. After two passes of sorting algorithm, the following array: 47 3 66 32 56 92 have been rearranged as 3 47 66 32 56 92. Which sorting algorithm is being used?
   A. Selection Sort
   B. Insertion Sort
   C. Bubble Sort
   D. Merge Sort

Summary

- One of the most common applications in computer science is sorting.
- Data may be sorted in either ascending or descending order.
- There are several types of sorting such as selection, insertion, bubble, merge and quick sort.
- Searching is the process of finding the location of a target among list of objects.
- There are several types of searching methods such as linear, binary and hashing.