Data Structures and Algorithms Lab

09. Sorting Techniques

Subject Code: 17ECSP201 **Lab No:** 09 **Semester:** III

Date: Oct 2017 **Batch:** MSM

Objective: Implementation of sorting techniques

Implement the below mentioned sorting techniques:

1. Cocktail Sort

This technique is a variation of Bubble sort. The Bubble sort algorithm always traverses elements from left and moves the largest element to its correct position in first iteration and second largest in second iteration and so on.

Cocktail Sort traverses through a given array in both directions alternatively.

Algorithm:

- The first stage loops through the array from left to right, just like the Bubble Sort. During the loop, adjacent items are compared and if value on the left is greater than the value on the right, then values are swapped. At the end of first iteration, largest number will reside at the end of the array.
- The second stage loops through the array in opposite direction- starting from the item just before the most recently sorted item, and moving back to the start of the array. Here also, adjacent items are compared and are swapped if required.

Example : 5 1 4 2 8 0 2

First Forward Pass:

(5 1 4 2 8 0 2) → (1 5 4 2 8 0 2), Swap since 5 > 1
(1 5 4 2 8 0 2) → (1 4 5 2 8 0 2), Swap since 5 > 4
(1 4 5 2 8 0 2) → (1 4 2 5 8 0 2), Swap since 5 > 2
(1 4 2 5 8 0 2) → (1 4 2 5 8 0 2)
(1 4 2 5 8 0 2) → (1 4 2 5 0 8 2), Swap since 8 > 0
(1 4 2 5 0 8 2) → (1 4 2 5 0 2 8), Swap since 8 > 2

After first forward pass, greatest element of the array will be present at the last index of array.

First Backward Pass:

(1 4 2 5 0 2 8) → (1 4 2 5 0 2 8)

(1 4 2 5 0 2 8) → (1 4 2 0 5 2 8), Swap since 5 > 0
(1 4 2 0 5 2 8) → (1 4 0 2 5 2 8), Swap since 2 > 0
(1 4 0 2 5 2 8) → (1 0 4 2 5 2 8), Swap since 4 > 0
(1 0 4 2 5 2 8) → (0 1 4 2 5 2 8), Swap since 1 > 0

After first backward pass, smallest element of the array will be present at the first index of the array. The process continues for all the remaining iterations.

2. Intelligent Design Sort

Algorithm:

The probability of the original input list being in the exact order it's in is 1/(n!). There is such a small likelihood of this that it's clearly absurd to say that this happened by chance, so it must have been consciously put in that order by an intelligent Sorter.

Therefore it's safe to assume that it's already optimally sorted in some way that transcends our naïve mortal understanding of "ascending order".

Any attempt to change that order to conform to our own preconceptions would actually make it less sorted.

Analysis:

This algorithm is constant in time, and sorts the list in-place, requiring no additional memory at all. In fact, it doesn't even require any of that suspicious technological computer stuff. Praise the Sorter!

3. Stooge Sort

Stooge sort algorithm can be explained as below:

Step 1 : If value at index 0 is greater than value at last index, swap them.
Step 2: Recursively,
  a) Stooge sort the initial 2/3rd of the array.
  b) Stooge sort the last 2/3rd of the array.
  c) Stooge sort the initial 2/3rd again to confirm.

4. Recursive Insertion Sort

Implement a recursive version of Insertion sort.

5. BogoSort

Algorithm:

```
while not isInOrder(array):
   shuffle(array)
```

Or put it this way:

```
while Array not sorted
    rearrange the array in a random order
```

As per algorithm, we need to keep shuffling until it gets sorted. How stupid is that? Well, it's also called 'stupid sort'.

If you think it's a monkey job, it's also called as 'monkey sort'. No wonder people also call it as 'slow sort'.

6. Max, Min, Mid Sort

Design a sorting technique given the following functions:

- getMax() – returns maximum element of array
- getMix() – returns minimum element of array
- getMid() – returns middle element of array

7. Gnome Sort

Algorithm:

Step 01: If you are index 0, increment to index 1

Step 02: If the current array element is larger or equal to the previous array element, then increment the index by 1

Step 03: If the current array element is smaller than the previous array element then swap these two elements and decrement the index by 1

Step 04: Repeat Step 02 and Step 03, until index i reaches the end of the array. When end of the array is reached then stop and the array is sorted.

**** Happy Coding ****