NAME: PREETHI S
USN: 1NH17IS072

TITLE: SORT ACCORDING TO PERSON’S HEIGHT USING COCKTAIL SORT.

CHAPTER 1

INTRODUCTION

One of the most fundamental algorithmic problem that was faced in the early days on computing is Sorting. As a result, significant computer science research was conducted to find a way to sort the set of data. We can realize now that there should be a good reason to make sorting that very important.

Sorting means arranging the data in either ascending or descending order. The Sorting process came into picture, as and when humans recognized the importance of searching on quicker basis.

There are a lot of things in real life that we need to search for, for example:

Search for a particular database record, searching roll numbers in a particular merit list, searching for a telephone no. in telephone directory, searching for a particular page in a book etc.

All this would finally result in a mess if the data was kept unordered as well unsorted, but luckily the process of sorting came into existence, making it very much easier for everyone to arrange the data in an order that is ascending or descending, hence making it very easier to search.

Thus Sorting arranges data in a sequence which makes searching easier.

In computer programming, Sorting some set of items is one of the critical tasks that always occur. This task of higher importance is performed intuitively by humans very often. In order to achieve this, computer programming has to follow a certain sequence of correct instructions. This sequence of instructions is called an algorithm. Thus, by Sorting Algorithm we mean to say that it is a method in which a list of unordered set of items are arranged or placed in ordered sequence. This sequence is determined by a Key for ordering. In terms of the performance and efficiency, the sorting algorithms exists as well as differs

Sorting generally divided into 2 types;

Internal Sorts: This classification of sorting uses actually primary storage or memory while processing. The complete set of data items are stored in Main memory itself and there is no need for secondary memory during sorting process. When this entire data which needs to be sorted can be accommodated at the same time in this memory, we call it Internal sorting. There is a dis-advantage in internal sorting, as it can only process relatively small lists due to memory constraints. There are 3 types of internal sorts.

1. SELECTION SORT: For example- Heap sort algorithm, Selection Sort algorithm
2. INSERTION SORT: For example- Shell sort algorithm, Insertion Sort algorithm
3. EXCHANGE SORT: For example- Quick Sort Algorithm, Bubble sort algorithm

External Sorts: while Internal sorting displays shortage of memory, in order to sort large data, there is need for Secondary memory or External memory. This process uses external memory such as HDD, to store the data which does not fit into the main memory. Thus, primary memory stores the currently being sorted data only. All external sorts are based on the process of merging. Different parts of data are sorted separately and merged together. For example- Merge Sort

Sorting Efficiency

Since the very start of the programming age, computer experts and the scientists have been working extensively on solving the problem of sorting by inventing various different algorithms to sort set of data items.

The two main criterias to judge which algorithm is better is mentioned below:

1. Time taken to sort the required data.
2. Memory Space required

COCKTAIL SORT

Cocktail sort is one of the variations of Bubble sort that is both a consistent sorting algorithm as well as a comparison sort. The algorithm differs from a bubble sort in a way that it sorts in both directions on each pass through the list. A slight change between both sorts is that Cocktail sort is comparatively more difficult to implement than a bubble sort, and solves the problem of bubble sorts.

The cocktail sort is called by various names like cocktail shaker sort, bidirectional bubble sort, shaker sort, shuffle sort, ripple sort, happy hour sort or shuttle sort.

In Simple, Cocktail sort is a type of Bubble Sort which traverses the list in both directions alternatively. It is different from bubble sort in the sense that, bubble sort traverses the list in forward direction only, while this algorithm traverses in forward as well as backward direction in one iteration

Different Sorting Algorithms

There are various techniques available for sorting, differentiated in terms of their efficiency and also space requirements. The below mentioned are some sorting techniques for our knowledge.

1) Bubble Sort
2) Selection Sort
3) Insertion Sort
4) Quick Sort
5) Heap Sort

Bubble Sort

This kind of sorting algorithm works by continuously swapping elements which are not in order until the whole list of items is in proper sequence. In this way, all the items can be seen as bubbling up the list in accordance with their prominant values.

The major benefit of the bubble sort is that it is very popular as well easy to implement. The items are swapped without use of additional memory, thus space consumption is also less. One of the main hindrances of the bubble sort is that it doesn't handle well a list containing a huge items. This is because the bubble sort requires n-squared processing steps for every n number of elements to be sorted. And so, it is majorly apt for academic teaching and not for real-life applications.

SELECTION SORT

The sorting algorithm works by continuously running through the list of items, each time selecting an item according to its order and also placing it in the right position in the sequence.

The major benefit of the selection sort is that it performs well on a small list. Furthermore, here no additional temporary storage is required beyond what is needed. Similar to the bubble sort, the selection sort requires n-squared number of steps for sorting n elements. In addition to this, the performance is very much influenced by starting ordering of the items before the actual sorting process. Because of the above mentioned information, this sort is suitable for a list of few elements that are in the random order.

INSERTION SORT

The insertion sorting continuously will scan the list of items, each time inserting the item in the unordered sequence into the right position.

The primary benefit of this sort is its simplicity. It also well exhibits a great performance when dealing with a small list of items. The insertion sort is an in-place sorting algorithm, because of this the space requirement is also less. The disadvantage is that it does not perform as much as the other, better sort algorithms. With n-squared steps needed for every n element to be sorted, the insertion sort does not deal well with a large list. Thus, the insertion sort is specifically useful only when sorting a list of few items.

QUICK SORT

This sort works on divide-and-conquer principle. It is basically divided into two sublist, in every element of first sublist to be arranged to be smaller than the pivot element. And second sublist greater than the pivot element. In the same way all elements are arranged.

The quick sort is the best sort among the other sorting techniquies , because of the sublists. And it does not required any additional storage space for it . the main disadvantage of it is similar to the other sorting like bubble, insertion etc.

MERGESORT

It similar to the quick sort and steps involved are

- it divides into two major sublist.
- And each sublist is known as the conquer.
- And merge them.

HEAP SORT

It is known as the comparison sort ,it is compared to the binary heap sory. It is similar to the selection to the , where it find the largest element and place at the end. And repeat the same process to all the elements.

COUNTING SORT

This sorting finds the specific key in the element. It works on the finding the distinct key.
And then doing some arithmetic operation it sort the elements.

RADIX SORT

It is similar to the other sorting techniquies , it sorts by comparing the lower bound element and then sorts the elements.

1.1 MOTIVATION OF PROJECT

The purpose of this project is to arrange a person according to their height. This process may help in many of the situations for example: arranging a student in the assembly hall, in many government jobs like police training etc they select a person through his/her height.

Cocktail sort helps to sort in bidirectional means in both forward and backward direction. Whereas in other sorting techniques they would sort in one direction like bubble sort, heap sort, merge sort ,bubble sort , quick sort, etc.

Fundamentally the algorithm for Cocktail sort and Bubble sort is the same. The difference is that the Cocktail Sort iterates through a given *data set* in **both** directions when sorting. So let’s break it down as below:

Each iteration of the algorithm is broken up into two stages.

Stage-1

This first stage will loop through the *data set* from bottom to top, just like Bubble Sort. During the looping process, adjacent items are compared. If at any point the “value on the left < than the value on the right”, the items are “swapped”. At the end of this first iteration, the largest number will actually reside at the end of the set.

Stage-2

The second stage would loop through the *data set* in the **opposite** direction that is from top to bottom - starting from the item just before the most recently sorted item, and then moving back towards the start of the list. Again, adjacent items are swapped if necessary.

The Cocktail Sort also is similar to the **Exchange Sorts** in a manner that elements are moved inside the *data set* during the sorting process.

1.2 PROBLEM STATEMENT

To design and implement a sorting, to arrange the person according to the height using cocktail sorting technique.

CHAPTER 2

SYSTEM REQURIMENT SPECIFICATION

PURPOSE

The purpose of the project is to provide the height of a person and then arrange then in particular order. This is once you enter the data in the system of each person height, it is not necessary to give the person height again and again. And it gives the output in the arranging order.
This helps many job training, in school assembly etc.

SCOPE: This soring recently came existence and this is used in many fields, compare other sorting this more far better , this sorting happens fast because of bidirectional forward and backward direction , the consumption is low where has in other sort it consumes more time due to the direction ,that is from left to right.

Whereas in this it starts comparing the data from left to right and left to right.

2.1 HARDWARE SYSTEM CONFIGURATION:

Process	-Intel core i5
Speed	-1.8 GHz
Ram	-256 MB(min)
Hard disk	-10 GB

2.2 SOFTWARE SYSTEM CONFIGURATION:

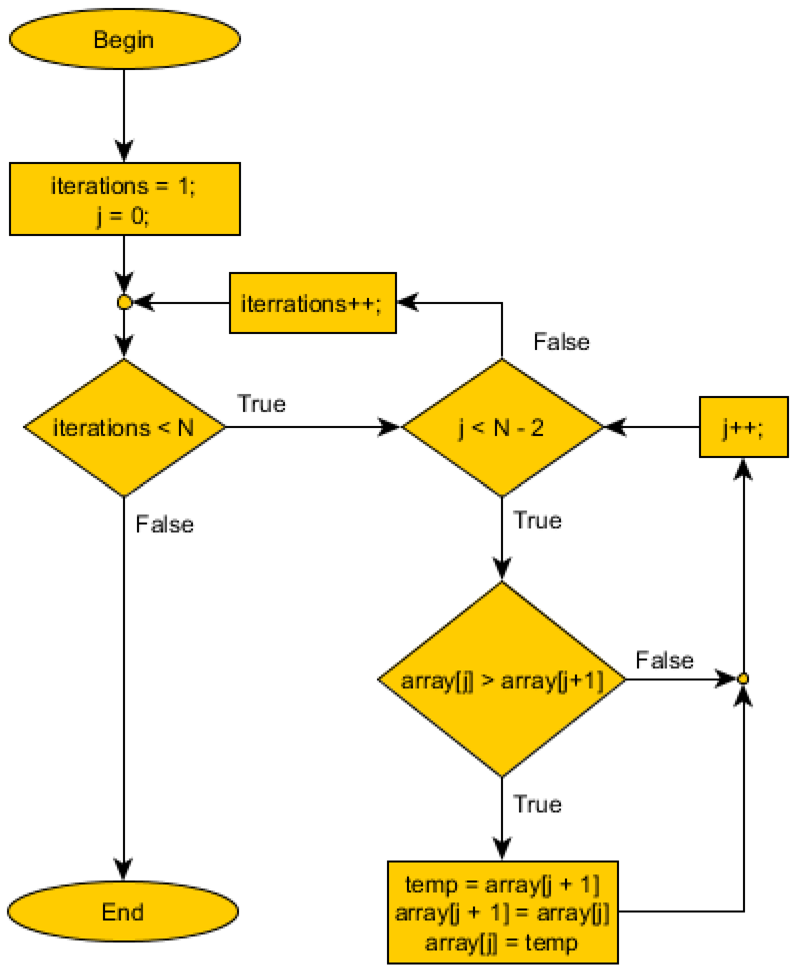
Operating system	-windows10
Programming language	- C
Compiler	-code::block

CHAPTER 3

METHODOLOGY

Cocktail sort is similar to the bubble sort in which both are even and comparison-based sorting al gorithms as shown in the flow chart below.
The Cocktail algorithm differs from a bubble sort in that it sorts in both directions on each pass t hrough the list .
This sorting algorithm is more difficult to implement than the bubble sort.
The first rightward pass will shift the largest element to its correct place at the end, and the following leftward pass will shift the smallest element to its correct place at the beginning.
The second complete pass will shift the second largest and second smallest elements to their correct.

3.1 FLOW CHART



3.2 ALGORITHM

Step1: Start

Step2: Each iteration of the algorithm is broken up into 2 stages:

Step3: This stage will loop through the *data set* from bottom to top, just like Bubble Sort. During the looping process, adjacent items are compared. If at any point the “value on the left < than the value on the right”, the items are “swapped”. At the end of this first iteration, the largest number will actually reside at the end of the set.

Step4: This stage would loop through the *data set* in the **opposite** direction that is from top to bottom - starting from the item just before the most recently sorted item, and then moving back towards the start of the list. Again, adjacent items are swapped if necessary.

Step5: Stop

Example : Let us consider an example array (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.

Second Forward Pass:

(0 1 4 2 5 2 8) ? (0 1 4 2 5 2 8)

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

(0 1 2 4 5 2 8) ? (0 1 2 4 5 2 8)

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

Second Backward Pass:

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

Now, the array is already sorted, but our algorithm doesn't know if it is completed. The algorithm needs to complete this whole pass without any swap to know it is sorted.

(0 1 2 2 4 5 8) ? (0 1 2 2 4 5 8)

(0 1 2 2 4 5 8) ? (0 1 2 2 4 5 8

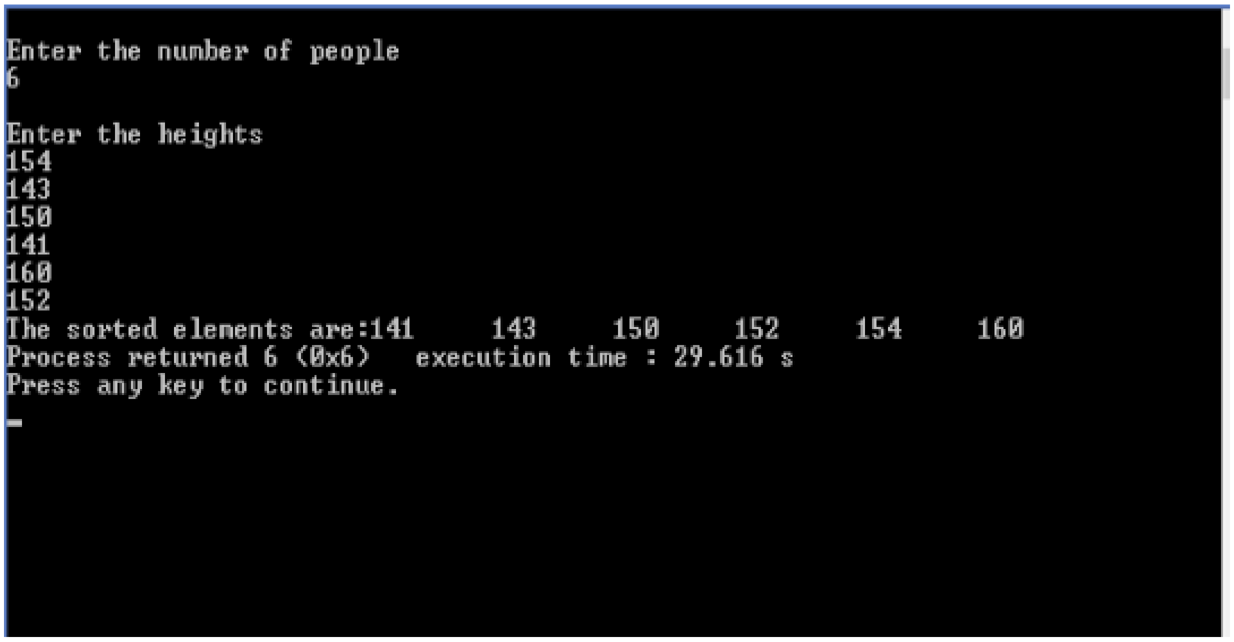
3.3 PROGRAM

```
# include <stdio.h>
int  main()
{
int data[50];
int i=0,j=0,n=0,c=0,MAX=0;
printf(“\n enter the number of people\n” );

scanf(“%d”,&MAX);
printf(“\n enter the heights” );
for(i=0 ; i<MAX; i++)
{
Scanf(“%d”, &data[i]);
}
n = MAX;
do
{
/* rightward pass will shift the largest element to its correct place at the end * /
for(i=0; i<n; i++)
{
if( data[i]>data[i+1])
{
data[i] = data[i] +data[i+1];
data[i+1]=data[i]-data[i+1];
data[i]= data[i]-data[i+1];
}
}
n=n-1;
/* leftward pass will shift the smallest element to its correct place at the beginning */
for( i=MAX-1, c=0; i>=c; i-- )
{
if(data[i]<data[i-1])
{
data[i]= data[i]+data[i-1];
data[i-1]=data[i]-data[i-1];
data[i]=data[i]- data[i-1];
}
}
c=c+1;
}
while(n!=0 && c!=0)
printf(“the sorted elements are:”);
for(i=0; i<MAX; i++)
{
printf( “%d\t”, data[i]);
}
}
```

CHAPTER 4

RESULTS



```
Enter the number of people
6

Enter the heights
154
143
150
141
160
152
The sorted elements are:141     143     150     152     154     160
Process returned 6 (0x6)   execution time : 29.616 s
Press any key to continue.
```

CONCLUSION AND FUTURE ENHANCEMENT

Cocktail is one of the best techniques to sort the data in a particular order. This not only saves the time, but also sorts the set of data in a bidirectional from left to right as well as right to left. This is considered as a major advantage of cocktail sort.

While compared to other sorting techniques, the cocktail sort is very fast and less time consuming with simple steps.

Sorting algorithm is one of the most basic research techniques in computer science. Sort is considered a very essential operation in computer programming. Thus In computer programming, a sorting algorithm is an efficient algorithm which performs a cardinal task that puts elements of a list or array in a certain order or arranges a collection of items into a particular pattern or order.

REFERENCE

https://www.geeksforgeeks.org/**cocktail-sort**

https://rosettacode.org/wiki/**Sorting**_algorithms/**Cocktail_sort**

https://www.sanfoundry.com/c-program-implement-**cocktail-sort**

https://buffered.io/posts/**sorting-algorithms-the-cocktail-sort**

bing.com/images

https://rosettacode.org/wiki/**Sorting**_algorithms/**Cocktail_sort**