Jyothy Institute of Technology

Data Structures Project

Sorting Techniques

Course Name: Data Structures and Applications

Code: 18CS32

Date of submission: 24/02/2022

Team members:

Sl no	NAME	USN	SIGNATURE
1.	Adithi Shankar	1JT20AI003	
2.	Deepak Prasad	1JT20AI005	
3.	Dheeraj N Kashyap	1JT20AI007	
4.	Girish S	1JT20AI008	
5.	Harshith Bhagle A	1JT20AI010	
6.	Kiran Kumar B.K	1JT20AI015	
7.	Prasanna Kumar K	1JT20AI030	
8.	Venkatesh S	1JT20AI051	
9.	Yukthi A	1JT20AI053	

INDEX

Course Outcomes

CO1: To Describe Different Types of Data Structure and their application.

CO2: To Apply Different Types of Sorting Algorithms for Problem Solving.

CO3: Implementing All the Data Structures in a High Level Programming Language for Problem Solving.

Program Outcome

PO1: Engineering Knowledge: Apply the Knowledge of Mathematics, Science, Engineering Fundamentals and Engineering Specialization To Complex engineering problems.

PO2: Individual and Team Work: Function effectively as an individual and As a member or leader in diverse teams, and in multidisciplinary settings.

Abstract

Sorting is one of the most fundamental problems in computer science, as it is used in most software applications.

Data-driven algorithms especially use sorting to gain an efficient access to data.
Many sorting algorithms with distinct properties for different architectures have been developed.
Searching for items and sorting through items are tasks that we do every day.

Searching and sorting are also common tasks in computer programs. We search for all occurrences of a word in a file in order to replace it with another word.
We sort the items on a list into alphabetical or numerical order.

As searching and sorting are common computer tasks, we have well-known algorithms for doing searching and sorting.

We'll look at three sorting algorithms here in detail, and go through examples of each algorithm and determine the performance of each algorithm, in terms of how **"rapidly"** each algorithm completes its task.

Keywords

- #include-Header File for Standard Input/ Output Operations.
- #include-It Is Used To Add General Utility Functions.
- #define-It Is Used To Define a Constant .
- For Loop- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- While Loop- while loop is a most basic loop in C programming. while loop has one control condition, and executes as long the condition is true. The condition of the loop is tested before the body of the loop is executed, hence it is called an entry-controlled loop.
- Simple If Statement- The statement inside the if block are executed only when condition is true, otherwise not.
- Here In This Program Some Of The User Defined Functions which are Used is as follows
  void display(int a[],int n);
  void insertion_sort(int a[], int n);
  void radix_sort(int a[], int n);
  int addr_sort(int a[], int n);
  int large_dig();

Introduction

Sorting is the process of arranging a list of elements in a particular order(Ascending or Descending).

In simple words, sorting means arranging the given elements or data in an ordered sequence. The main purpose of sorting is to easily & quickly locate an element in a sorted list & design an efficient algorithm around it. In this blog we will understand different sorting algorithms & how to implement them in C.

Examples:

Our telephone directories, English Dictionaries are some examples in which names or words are arranged in alphabetical order.

Ranks based on the scores is another common example.

Problem Statement

To design and develop a menu driven program using the following sorting algorithms:

1. Radix sort
2. Insertion sort
3. Address calculation sort

Solution

The below implemented code consists of many user defined functions which show how the different types of sorting algorithms run.

Implementation

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>


/*****************************************************************************
                                1 INSERTION SORT
*****************************************************************************/

void insertion_sort(int arr[], int n)
{
        int i, j, temp;
        for(i=1; i<n; i++)
        {
                temp = arr[i];
                j = i-1;
                while((temp < arr[j]) && (j>=0))
                {
                        arr[j+1] = arr[j];
                        j--;
                }
                arr[j+1] = temp;
        }
}

void insert_fun()
{
        int arr[100], i, n;
        printf("\nEnter the number of elements: ");
        scanf("%d",&n);
        printf("\nEnter %d elements: ", n);
        for(i=0; i<n; i++)
        {
                scanf("%d",&arr[i]);
        }
        insertion_sort(arr,n);

        printf("\n_____Sorted elements_____\n");
                for(i=0; i<n; i++)
        {
                printf("%d\t", arr[i]);
        }
}
```

```
/*******************************************************************************
                                2 RADIX SORT
*******************************************************************************/


int getMax(int arr[], int n)
{
    int max = arr[0],i;
    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}

void countSort(int arr[], int n, int pos)
 {
    int b[n];
    int i, count[10] = { 0 };
    for (i = 0; i < n; i++)                     //Get required Digit
    ++count[(arr[i] / pos) % 10];
    for (i = 1; i < 10; i++)                    //Adding- Previous+Current
    count[i] =count[i]+count[i - 1];
    for (i = n - 1; i >= 0; i--)                //Built array b
    b[--count[(arr[i] / pos) % 10]]=arr[i];
    for (i = 0; i < n; i++)                     //copy array b to Arr
    arr[i] = b[i];

 }

void radix_sort(int arr[], int n)
{
    int i, m = getMax(arr, n);
    int pos;
    for (pos = 1; m/pos > 0; pos*= 10)
    countSort(arr, n, pos);
    printf("Using Radix Sort \n");
    printf("\n_____Sorted elements_____\n");
        for(i=0; i<n; i++)
        {
                printf("%d\t", arr[i]);              //Output
    }
}

void radix_fun()
{
        int arr[100], i, n;
        printf("\nEnter the number of elements: ");
```

```
        scanf("%d",&n);
        printf("\nEnter %d elements: ", n);
        for(i=0; i<n; i++)
        {
                scanf("%d",&arr[i]);                //Taking input
        }
        radix_sort(arr, n);                         //********

}

/***************************************************************************
                              2 ADDRESS CALCULATION SORT
***************************************************************************/
struct  node
{
        int info ;
        struct node *link;
};
struct node *head[10];
int n,i,arr[100];


void insert(int num,int addr)
{
        struct node *q,*tmp;
        tmp=  (struct node *) malloc(sizeof(struct node));
        tmp->info=num;
                                                                    /*list empty or item to be
added in begining */
        if(head[addr] == NULL || num < head[addr]->info)
        {
                tmp->link=head[addr];
                head[addr]=tmp;
                return;
        }
        else
        {
                q=head[addr];
                while(q->link != NULL && q->link->info < num)
                        q=q->link;
                tmp->link=q->link;
                q->link=tmp;
        }
}                                                                    /*End of insert()*/


int large_dig()                                                     /* Finds number of digits in the
largest element of the list */
```

```
{

        int large = 0,ndig = 0 ;

        for(i=0;i<n;i++)
        {
                if(arr[i] > large)
                        large = arr[i];
        }
        printf("Largest Element is %d \n",large);
        printf("\n");
        while(large != 0)
        {
                ndig++;
                large = large/10 ;
        }

        printf("Number of digits in it are %d\n",ndig);
        return(ndig);
} /*End of large_dig()*/


int hash_fn(int number,int k)
{
                                                        /*Find kth digit of the number*/
        int digit,addr,i;
        for(i = 1 ; i <=k ; i++)
        {
                digit = number % 10 ;
                number = number /10 ;
        }
        addr=digit;
        return(addr);
}/*End of hash_fn()*/


void addr_sort()
{
        int i,k,dig;
        struct node *p;
        int addr;
        k=large_dig();
        for(i=0;i<=9;i++)
                head[i]=NULL;
        for(i=0;i<n;i++)
        {
                addr=hash_fn( arr[i],k );
                insert(arr[i],addr);
```

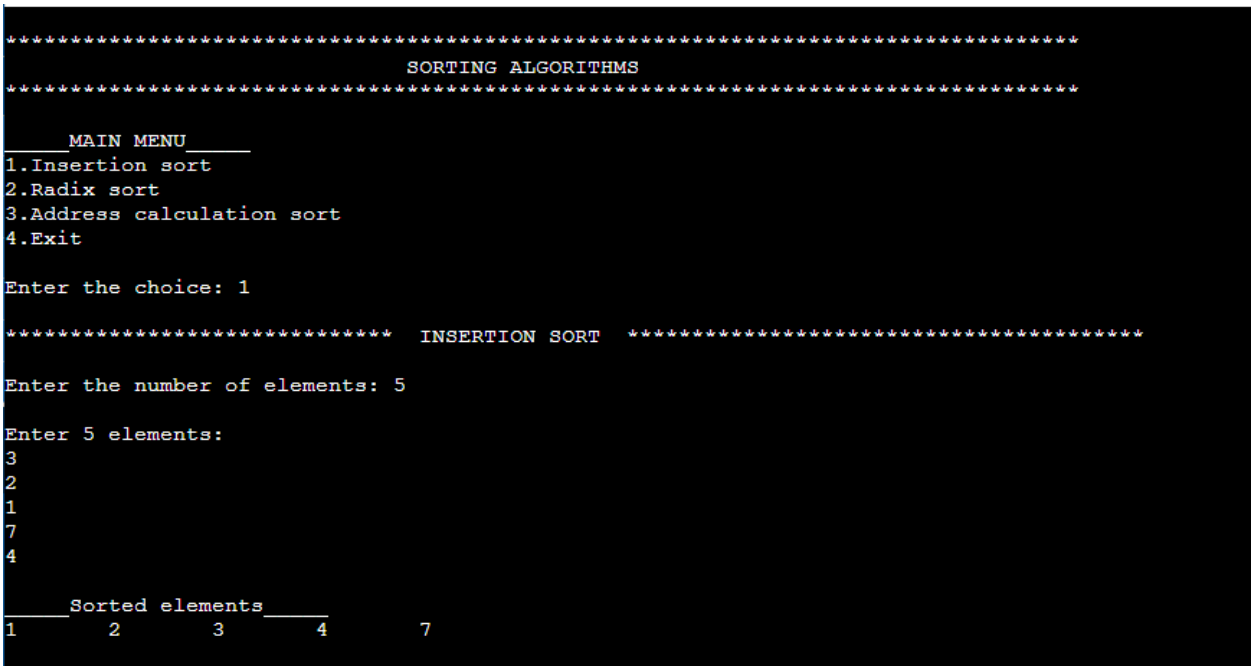
```
        }

        for(i=0; i<=9 ; i++)
        {
                printf("head(%d) -> ",i);
                p=head[i];
                while(p!=NULL)
                {
                        printf("%d ",p->info);
                        p=p->link;
                }
                printf("\n");
        }

        /*Taking the elements of linked lists in array*/
        i=0;
        for(k=0;k<=9;k++)
        {
                p=head[k];
                while(p!=NULL)
                {
                        arr[i++]=p->info;
                        p=p->link;
                }
        }
}/*End of addr_sort()*/


void addcalfun()
{

        int i;
        printf("Enter the number of elements : ");
        scanf("%d", &n);
        printf("Enter %d element : ",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }/*End of for */

        addr_sort();
        printf("Sorted list is :\n");
        for(i=0;i<n;i++)
                printf("%d  ",arr[i]);
        printf("\n");
}
```

```
/*****************************************************************************
                              MAIN FUNCTION
*****************************************************************************/


void main()
{
        int ch;
        printf("\n*********************************************************************************************\n");
        printf("                                SORTING ALGORITHMS
\n");
        printf("*********************************************************************************************\n");
        while(1)
        {
                printf("\n_____MAIN MENU_____\n");
                printf("1.Insertion sort\n2.Radix sort\n3.Address calculation sort\n4.Exit\n");
                printf("\nEnter the choice: ");
                scanf("%d",&ch);
                printf("\n");

                switch (ch)
                {
                        case 1:printf("*****************************  INSERTION SORT
****************************************\n");
                               insert_fun();

printf("\n\n*********************************************************************************************\n");
                               break;

                        case 2:printf("*****************************  RADIX SORT
****************************************\n");
                               radix_fun();

printf("\n\n*********************************************************************************************\n");
                               break;

                        case 3:printf("*****************************  ADDRESS CALCULATION SORT
****************************************\n");
                               addcalfun();

printf("\n\n*********************************************************************************************\n");
                               break;

                        case 4:exit(0);
                               break;

                        default:printf("Invalid choice!!, Try again\n");
```

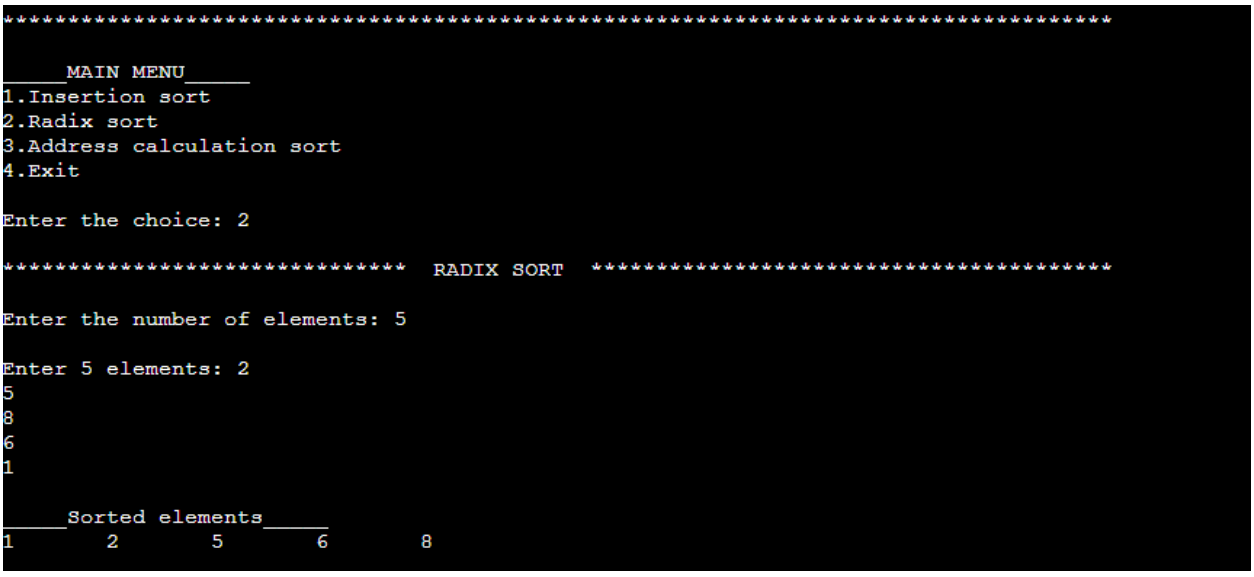
```
printf("\n\n*********************************************************************************\n");
                }
        }
}
```

Output

Insertion Sort



Radix Sort



Address Calculation Sort

```
********************************************************************************************

_____MAIN MENU_____
1.Insertion sort
2.Radix sort
3.Address calculation sort
4.Exit

Enter the choice: 3

*******************************  ADDRESS CALCULATION SORT  ***************************************
Enter the number of elements : 5
Enter 5 element : 3
1
5
4
7
Largest Element is 7

Number of digits in it are 1
head(0) ->
head(1) -> 1
head(2) ->
head(3) -> 3
head(4) -> 4
head(5) -> 5
head(6) ->
head(7) -> 7
head(8) ->
head(9) ->
Sorted list is :
1  3  4  5  7
```

Conclusion

Sorting is important in programming for the same reason it is important in everyday life. It is easier and faster to locate items in a sorted list than unsorted. Sorting algorithms can be used in a program to sort an array for later searching or writing out to an ordered file or report. While Writing the Code /Program We Came Across Many Errors. But we as a team were successful in solving the Complete Problem.