Sorting Algorithms

Implementation and Analysis of sorting Algorithms

Bubble Sort.

```
Step 1:  Start.
Step 2:  Loop through all elements of list
                 Loop through elements falling ahead
                         If current element is greater than next element
                                 Swap them to bubble up the highest element
                         End if
Step 3:  Stop.
```

```
#include
<stdio.h>
#include<conio.h
> #define  MAX
10
int list[MAX] = {1,8,4,6,0,3,5,2,7,9};

void display()
{
        int i;
        printf("[[
        ");
        for(i = 0; i < MAX; i++)
        {
                printf("%d   ",list[i]);
        }
        printf("]]");
}

void bubbleSort()
{
        int
        temp;
        int i,j;
        for(i = 0; i < MAX-1; i++)
        {
                printf("Iteration %d :
```

```
\n",i+1); for(j = 0; j <
MAX-1-i; j++)
{
        printf(" Items compared: [ %d, %d ] ", list[j],list[j+1]);
        if(list[j] > list[j+1])
        {
                temp = list[j];
                list[j] =
                list[j+1];
                list[j+1] =
                temp;
                printf(" => swapped [%d, %d]\n",list[j],list[j+1]);
            }
            else
            {
                printf(" => not swapped\n");
            }
        }

        printf("\nAfter Iteration %d#:
        ",(i+1)); display();
        printf("\n\n")
        ; getch();
    }
}

void main()
{
    clrscr();
    printf("\n\nInput Array: \n\n\t");
    display();
    printf("\n\n");
    getch();
    bubbleSort();
    printf("\nOutput Array:
    \n\n\t"); display();
    getch();
}
```

2. Insertion Sort

```
#include <stdio.h>

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

```
void printArray(int arr[], int n) {
   for (int i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

int main() {
   int arr[] = {12, 11, 13, 5, 6};
   int n = sizeof(arr)/sizeof(arr[0]);

   insertionSort(arr, n);
   printf("Sorted array: \n");
   printArray(arr, n);
   return 0;
}
```

1. Merge Sort.

Step 1: Start.
Step 2: If it is only one element in the list it is already sorted, return.
Step 3: Divide the list recursively into two halves until it can no more be divided.
Step 4: Merge the smaller lists into new list in sorted order.
Step 5: Stop.

```
#include
<stdio.h>
#include<conio.h
> #define max
10  int ct_merge
= 0; int
ct_divide = 0;
int a[max] = {1,8,4,6,0,3,5,2,7,9};
int b[max];
```

```
void display()
{
        int i;
        printf("[[
        ");
        for(i = 0; i < max; i++)
        {
                printf("%d   ",a[i]);
        }
        printf("]]");
}

void merging(int low, int mid, int high)
{
        int l1, l2, i;
        for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++)
        {
                if(a[l1] <= a[l2])
                        b[i] = a[l1++];
                else
                        b[i] = a[l2++];
        }
        while(l1 <= mid)
                b[i++] = a[l1++];
        while(l2 <= high)
                b[i++] = a[l2++];
        printf("\nMerge  %d:--  ",ct_merge);
        ct_merge++;
        for(i = low; i <= high; i++)
        {
                a[i] = b[i];
                printf("%d
                ",a[i]);
        }
}

void sort(int low, int high)
{
        int mid,i;
        printf("\nDivide %d:--
        ",ct_divide); ct_divide++;
```

```
for(i = low; i <= high; i++)
{
        printf("%d  ",a[i]);
        }
        if(low < high)
        {
                mid = (low + high) /
                2; sort(low, mid);
                getch();
                sort(mid+1,
                high); getch();
                merging(low, mid, high);
        }
        else
        {
                return;
        }
}


void main()
{
        clrscr();
        printf("\n\nList before sorting\n\n\t");
        display();
        printf("\n");
        sort(0, max-1);
        printf("\n\n\nList after sorting\n\n\t");
        display();
        getch();
}
```

To implement and understand the working of Linear Search and Binary Search

Linear Search

```
#include <stdio.h>

int linearSearch(int arr[], int n, int key) {
    for(int i = 0; i < n; i++) {
        if(arr[i] == key)
            return i;  // return index
    }
    return -1; // not found
}

int main() {
    int arr[] = {5, 3, 8, 6, 2};
    int key = 6;
    int n = sizeof(arr)/sizeof(arr[0]);

    int result = linearSearch(arr, n, key);
    if(result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```

Binary Search

```
#include <stdio.h>

int binarySearch(int arr[], int n, int key) {
    int low = 0, high = n - 1;

    while(low <= high) {
        int mid = (low + high) / 2;

        if(arr[mid] == key)
            return mid;
        else if(arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}

int main() {
    int arr[] = {2, 4, 6, 8, 10, 12};
    int key = 10;
    int n = sizeof(arr)/sizeof(arr[0]);

    int result = binarySearch(arr, n, key);
    if(result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```