Experiment No. (1)

AIM: Consider a student database of SY COMP class (at least 10 records). Database contains different fields of every student like roll No., Name and SGPA. (array of object of class)

(a) Design a roll call list, arrange list of students according to roll numbers in ascending order (use bubble sort)

(b) Arrange list of students alphabetically (use insertion sort)

(c) Arrange list of students to find out first ten toppers from class. (use quick sort)

(d) Search students according to SGPA. If more than one student having same SGPA, the print list of all students having same SGPA.

(e) Search a particular student according to name using binary search without recursion.

OBJECTIVE:

1. To study the concepts of array of structure
2. To understand the concepts, algorithms & sorting applications
3. To understand the concepts, algorithms and searching applications.

OUTCOME:

1. Understand linear data structure – array structure

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

② Apply different sorting techinques on array of structure (Bubble sort, Insertion and Quicksort) with output for every pass.
③ Apply different searching techniques on array of structure (Linear Search, Binary search) and display the output for every pass.
④ Calculate time complexity.

THEORY :

① Structure ⟶

Structure is collection of hetrogeneous types of data. In C++, a structure collects different data items in such a way that they can be referenced as a single unit. Data items in structure are called fields or members of the structure.

Creating a structures ⟶

```
struct student
{
  int roll_no;
  char name[15];
  float sgpa;
};
```

— Here struct is keyword used to declare a structure.
— Student is the name of structure.

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

structure is represented as follows :

Declaring structure variables :

After declaring a structure we can define a structure variable for instance, the following structure variables are defined with structure data type of automobile from previous section :

```
Struct student S1, S2, S3;
```

Here there structure variable S1, S2, S3 are defined by the structure of student all three structure variable contain three members of the structure student.

Referencing structure members with the dot operator :

Given the struct student and S_1 is variable of type struct student we can access the fields in the following way : ⟶

```
S1.rollno
S1.name
S1.sgpa
```

Hence, here the structure name and its member's name are seprated by the dot (.) operator.

Camlin

pointer to structure →

- A pointer variable capable of storing address of a structure variable as follows:

```
struct student *ptr;
```

Let structure variable declared as,

```
struct student S1;
```

Then, ptr can store the address of S1 by:

```
ptr = &S1;
```

Referencing a structure member with → (arrow operator)

using ptr we can access the fields of S1 as:

ptr → name or (*ptr).name

Because of this clearness, the → operator is more frequently used in programs that of dot operator.

(2) Array structure →

Declaration →

```
struct student s[20];
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Bubble sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

This algorithm is not suitable for large data sets as its average and worst case time complexity are $O(n^2)$ where n is the number of items.

How Bubble sort Work?

We take an unsorted array for our example.
Bubble sort takes $O(n^2)$ time so we're keeping it short and precise

14	33	27	35	10

Bubble sort starts with the very first two elements, comparing them to check which one is greater.

14	33	27	35	10

In this case, value 33 is greater than 14, so it is already in sorted locations, Next, we compare 33 with 27.

rudraksh.karpe.cs@ghrcem.raisoni.net

14	33	27	35	10

We find that 27 is smaller than 33 and the two values must be swaped.

14	33	27	35	10

The New array →

14	27	33	35	10

Next we Compare 33 and 35. We find that both are in already sorted position.

14	27	33	35	10

Then we move to next values. 35 and 10.

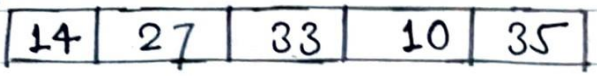
14	27	33	35	10

We know then that 10 is smaller than 35. Hence they are not sorted.

14	27	33	~~10~~ 35	~~35~~ 10

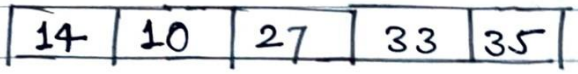
rudraksh.karpe.cs@ghrcem.raisoni.net

We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this

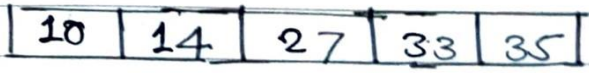


To be precise, we are now showing how an array should look like after each iteration. it should be like this ——→

Notice, after each iteration one value moves at the end.



And when there's no swap required, bubble sort learn that an array is Completely sorted.



rudraksh.karpe.cs@ghrcem.raisoni.net

Insertion sort

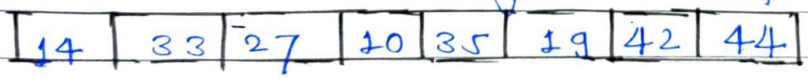
— This is an in-place comparison based sorting algorithm. Here a sublist is maintained which is always sorted. For example. ——. The lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sublist, has to find its ~~approach~~ appropriate place and then it has to be inserted there. Hence, the name insertion sort.

— The array is searched sequentially and unsorted items are moved and inserted into the sorted sublist (in the same array).

— This algorithm is not suitable for large data sets as its average and worst time complexity are $O(n^2)$, where (n) is the number of items.

How Insertion sort Work?

We take an unsorted array for our example.

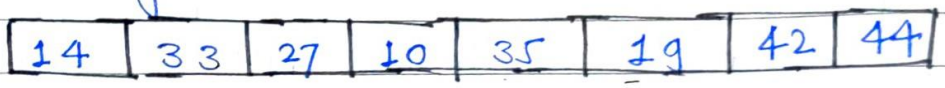


Insertion sort compares first two elements

rudraksh.karpe.cs@ghrcem.raisoni.net

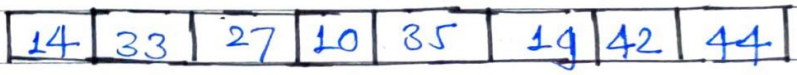
Page :
Date :

It finds that both 14 and 33 are already in ascending order. For now, 14 is sorted sublist.



Insertion sort moves ahead and Compares 33 with 27.

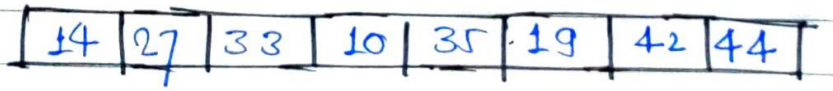
And finds that 33 is not in the Correct position.



It swaps 33 with 27. It also checks with the elements of sorted sublist.

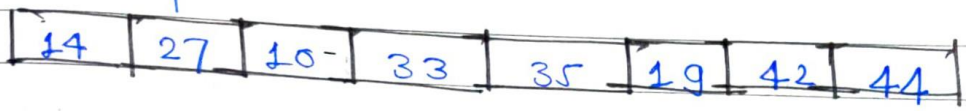
By now we have 14 and 27 in the sorted sub list

These values are not in a sorted order

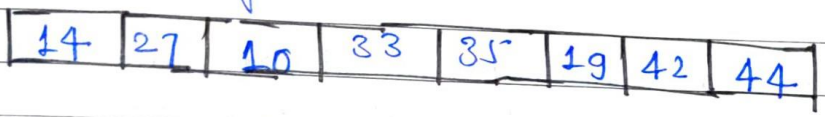


rudraksh.karpe.cs@ghrcem.raisoni.net

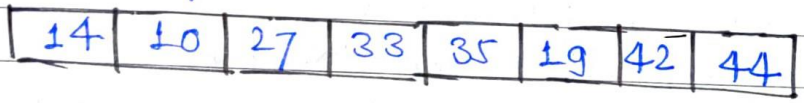
So we swap them



However, swapping makes 27 and 10 unsorted



Here, we swap them too,



We swap them again. By the end of third iteration, we have sorted sublist of 4 items.

This process goes on until all the unsorted values are covered in a sorted sub list.

rudraksh.karpe.cs@ghrcem.raisoni.net

Quick sort

- Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays.
- A large array is partitioned into two arrays one of which holds value of smaller than the specified value.
- Quicksort partition an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large sized data sets as its average and worst-case complexity are $O(n^2)$, respectively.

Quick sort: Algorithm → (Pivot)

Based on our understanding of partitioning in quick sort, we will now try to write an algorithm for it, which is as follows -

Step (1) → Choose the highest index value has pivot.
Step (2) → Take two variables to point left and right of the list excluding pivot.
Step (3) → Left points to low index.
Step (4) → Right points to the high.
Step (5) → While value is less than pivot move right
↓
at left

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

step (6) → While value at right is greater than the (pivot) pivot move left.

step (7) → If both step (5) and step (6) does not match swap left and right.

step (8) → If left ≥ right, the point where they met is new pivot.

Quick sort Algorithm →

step 1 – Make the right-most index value pivot
step 2 – partition the array using pivot value
step 3 – quicksort left partition recursively
step 4 – quicksort right partition recursively.

rudraksh.karpe.cs@ghrcem.raisoni.net

Binary search.

- Binary search is a fast search Algorithm with run time Complexity of $O(\log n)$.
- This search algorithm works on the principal of divide and Conquer.
- Data collection should be in sorted form, for ~~this~~ to make this algorithm work.
- Binary search looks for a particular item by Comparing the middle item of the collection. If a match occurs then the match occurs, then the index of the item is returned. If the middle is greater than the item is searched in the sub-array to left of the middle item. Otherwise the item is searched for the sub array to the right of the middle item.
- This process continues in sub arrays as we until the size of the sub array reduces to zero.

Working of Binary search.

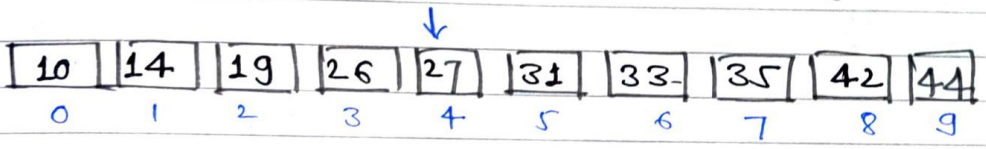
10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

rudraksh.karpe.cs@ghrcem.raisoni.net

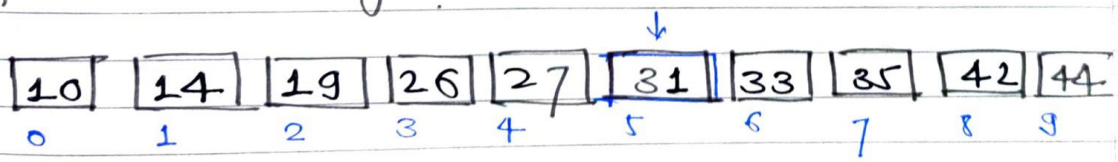
First, we'll determine half of the array by using the formula below.

Mid = low + (high - low)/2.

Hence, (0 + (9 - 0))/2 = 4 (integer value of 4.5)
So, 4 is the mid of the array



Now we compare the value sorted at location 4, with the value being searched, i.e. 31. We find that value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

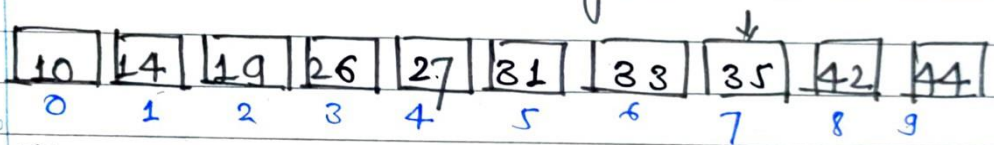


We change our low to mid + 1 and find the new mid val.

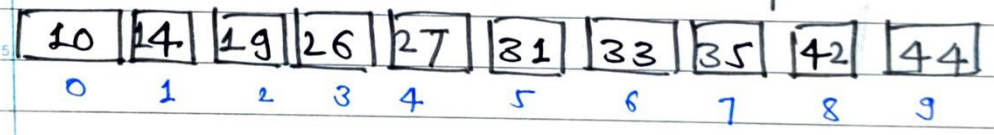
low = mid + 1.

mid = low + (high - low) / 2

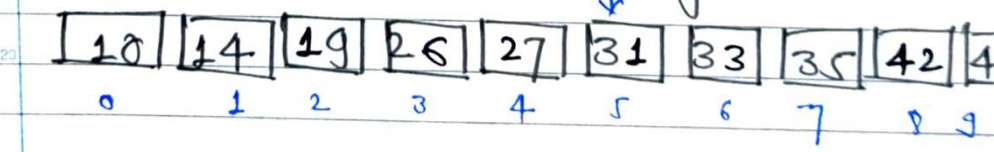
Our new mid is (7) now. We compare the value stored at location 7 with our target val. 31.



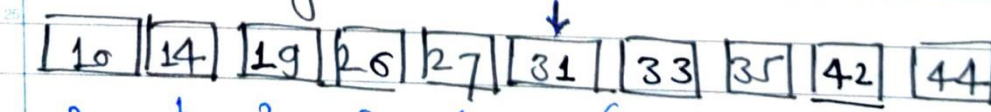
The value stored at location 7 not a match, rather it is more than what we are looking for. So, the value must be in the lower part location.



Hence, we calculate the mid again. This time it is (5)



We compare the value stored at location 5 with our target value. We find that it is a match.



We conclude that the target val. is stored at 5

rudraksh.karpe.cs@ghrcem.raisoni.net

Program code:

```
#include <iostream>
#include <string.h>
using namespace std;
typedef struct student
{
   int roll_num;
   char name[20];
   float sgpa;
} stud;
void create(stud s[20], int n);
void display(stud s[20], int n);
void bubble_sort(stud s[20], int n);
void insertionSort(stud s[20], int n);
void quick_sort(stud s[20], int, int);
int partition(stud s[20], int, int);
void search(stud s[20], int n, float key);
int bsearch(stud s[20], char x[20], int low, int high);

int main()
{
   stud s[20];
   int ch, n, result;
   float key;
```

```
char x[20];
do
{
   cout << "\n 1) Create Student Database ";
   cout << "\n 2) Display Student Records ";
   cout << "\n 3) Bubble Sort ";
   cout << "\n 4) Insertion Sort ";
   cout << "\n 5) Quick Sort ";
   cout << "\n 6) Linear search ";
   cout << "\n 7) Binary search  ";
   cout << "\n 8) Exit ";
   cout << "\n Enetr Your Choice:=";
   cin >> ch;
   switch (ch)
   {
   case 1:
      cout << "\n Enter The Number Of Records:=";
      cin >> n;
      create(s, n);
      break;
   case 2:
      display(s, n);
      break;
   case 3:
      bubble_sort(s, n);
```

```
            break;
        case 4:
            insertionSort(s, n);
            break;
        case 5:
            quick_sort(s, 0, n - 1);
            cout << "\n"
                 << "\t"
                 << "Roll No"
                 << "\t"
                 << " Name"
                 << "\t"
                 << "sgpa";
            for (int i = n - 1; i >= n - 10; i--)
            {
                cout << "\n";
                cout << "\t " << s[i].roll_num << "\t " << s[i].name << "\t " <<
s[i].sgpa;
            }
            break;
        case 6:
            cout << "\n Enter the sgpa which u want to search:=";
            cin >> key;
            search(s, n, key);
            break;
```

```
        case 7:
            cout << "\n Enter the name of student  which u want to search:=";
            cin >> x;
            insertionSort(s, n);
            result = bsearch(s, x, 0, (n - 1));
            if (result == -1)
            {
                cout << " \n Student name you want to search for is not present
! \n";
            }
            else
            {
                cout << " \n The student  is present :\t" << s[result].name;
            }
            break;
        case 8:
            return 0;
        default:
            cout << "\n Invalid choice !! Please enter your choice again." <<
endl;
        }
    } while (ch != 8);
}


void create(stud s[20], int n)
{
```

```
    int i;
    for (i = 0; i < n; i++)
    {
        cout << "\n Enter the roll number:=";
        cin >> s[i].roll_num;
        cout << "\n Enter the Name:=";
        cin >> s[i].name;
        cout << "\n Enter the sgpa:=";
        cin >> s[i].sgpa;
    }
}
void display(stud s[20], int n)
{
    int i;
    cout << "\n"
         << "\t"
         << "Roll No"
         << "\t"
         << " Name"
         << "\t"
         << "sgpa";
    for (i = 0; i < n; i++)
    {
        cout << "\n";
```

```
        cout << "\t " << s[i].roll_num << "\t " << s[i].name << "\t " <<
s[i].sgpa;
    }
}
//bubble sort to sort in ascending order on roll number


void bubble_sort(stud s[20], int n)
{
    int i, j;
    stud temp;
    for (i = 1; i < n; i++)
    {
        for (j = 0; j < n - i; j++)
        {
            if (s[j].roll_num > s[j + 1].roll_num)
            {
                temp = s[j];
                s[j] = s[j + 1];
                s[j + 1] = temp;
            }
        }
    }
}
// insertion sort to sort on names in ascending order
void insertionSort(stud s[20], int n)
```

```
{
   int i, j;
   stud key;
   for (i = 1; i < n; i++)
   {
      key = s[i];
      j = i - 1;
      /* Move elements of arr[0..i-1], that are
      greater than key, to one position ahead
      of their current position */
      while (j >= 0 && strcmp(s[j].name, key.name) > 0)
      {
         s[j + 1] = s[j];
         j = j - 1;
      }
      s[j + 1] = key;
   }
}

//Quick sort to sort on sgpa
void quick_sort(stud s[20], int l, int u)
{
   int j;
   if (l < u)
   {
```

```
        j = partition(s, l, u);
        quick_sort(s, l, j - 1);
        quick_sort(s, j + 1, u);
    }
}
int partition(stud s[20], int l, int u)
{
    int i, j;
    stud temp, v;
    v = s[l];
    i = l;
    j = u + 1;
    do
    {
        do
            i++;

        while (s[i].sgpa < v.sgpa && i <= u);

        do
            j--;
        while (v.sgpa < s[j].sgpa);

        if (i < j)
        {
```

```
            temp = s[i];
            s[i] = s[j];
            s[j] = temp;
        }
    } while (i < j);

    s[l] = s[j];
    s[j] = v;

    return (j);
}

// linear search for sgpa if more than one student having same sgpa print all of them
void search(stud s[20], int n, float key)
{
    int i;
    bool found = false;
    cout << "\n"
        << "\t"
        << "Roll No"
        << "\t"
        << " Name"
        << "\t"
        << "sgpa";
```

```
    for (i = 0; i < n; i++)
    {
        if (key == s[i].sgpa)
        {
            cout << "\n\t " << s[i].roll_num << "\t " << s[i].name << "\t " <<
s[i].sgpa;
            found = true;
            break;
        }
        else
        {
        }
    }
    if (found == false)
    {
        cout << "No record found for this sgpa value";
    }
}

int bsearch(stud s[20], char x[20], int low, int high)
{
    int mid;
    while (low <= high)
    {
        mid = (low + high) / 2;
```

```
        if (strcmp(x, s[mid].name) == 0)
        {
            return mid;
        }
        else if (strcmp(x, s[mid].name) < 0)
        {
            high = mid - 1;
        }
        else
        {
            low = mid + 1;
        }
    }

    return -1;
}
```

Output:

The Database of students:

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=1

Enter The Number Of Records:=10

Enter the roll number:=86

Enter the Name:=Rudraksh

Enter the sgpa:=9.9

Enter the roll number:=56

Enter the Name:=Mihir

Enter the sgpa:=7.9

Enter the roll number:=69

Enter the Name:=Devang

Enter the sgpa:=8.2
```

Student Records:

```
Enter the roll number:=78

Enter the Name:=Varun

Enter the sgpa:=9.3

Enter the roll number:=99

Enter the Name:=Prathmesh

Enter the sgpa:=3.5

Enter the roll number:=33

Enter the Name:=Eshan

Enter the sgpa:=9.3

Enter the roll number:=88

Enter the Name:=Devesh

Enter the sgpa:=8.3

Enter the roll number:=77

Enter the Name:=Vaishnav

Enter the sgpa:=8.2
```

rudraksh.karpe.cs@ghrcem.raisoni.net

```
Enter the roll number:=66

Enter the Name:=Adnan

Enter the sgpa:=7.3

Enter the roll number:=22

Enter the Name:=Krishna

Enter the sgpa:=8.3
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Students Records:

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=2

        Roll No  Name    sgpa
         86      Rudraksh        9.9
         56      Mihir   7.9
         69      Devang  8.2
         78      Varun   9.3
         99      Prathmesh       3.5
         33      Eshan   9.3
         88      Devesh  8.3
         77      Vaishnav        8.2
         66      Adnan   7.3
         22      Krishna         8.3
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Bubble Sort:

```
Enetr Your Choice:=3

1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=2

        Roll No  Name   sgpa
         22      Krishna        8.3
         33      Eshan   9.3
         56      Mihir   7.9
         66      Adnan   7.3
         69      Devang  8.2
         77      Vaishnav       8.2
         78      Varun   9.3
         86      Rudraksh       9.9
         88      Devesh  8.3
         99      Prathmesh      3.5
```

Insertion Sort:

```
Enetr Your Choice:=4

1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=2

        Roll No  Name    sgpa
         66      Adnan    7.3
         69      Devang   8.2
         88      Devesh   8.3
         33      Eshan    9.3
         22      Krishna          8.3
         56      Mihir    7.9
         99      Prathmesh        3.5
         86      Rudraksh         9.9
         77      Vaishnav         8.2
         78      Varun    9.3
```

Quick Sort:

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=5

        Roll No  Name   sgpa
         86      Rudraksh        9.9
         78      Varun   9.3
         33      Eshan   9.3
         22      Krishna         8.3
         88      Devesh  8.3
         77      Vaishnav        8.2
         69      Devang  8.2
         56      Mihir   7.9
         66      Adnan   7.3
         99      Prathmesh       3.5
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Linear Search:

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=6

Enter the sgpa which u want to search:=9.9

        Roll No  Name    sgpa
         86      Rudraksh        9.9
```

rudraksh.karpe.cs@ghrcem.raisoni.net

Binary Search:

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=7

Enter the name of student  which u want to search:=Mihir

The student  is present :      Mihir
```

Exit:

```
 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=8
```

rudraksh.karpe.cs@ghrcem.raisoni.net