(DATA STRUCTURES AND ALOGARITHMS)

(SORTING AND BINARY SEARCH)

(ALI AKBER BSCS 3RD SS1)

SORTING ARRAY.

```
#include <iostream>

using namespace std;

const int MAX=7;

class OrderArray{

        private:

        int arr[MAX];

        int count;

        public:

        OrderArray(){

                count=0;

        }

        int size(){

                return count;

        }

                void insertion(int val){

        int i,j;

        for(i=0;i<count;i++){

                if(arr[i]>val)

                break;

        }

        for(j=count;j>i;j--){

                arr[j]=arr[j-1];
        }
          arr[i]=val;
                count++;
        }
        int operator[](int index) {
                if (index >= 0 && index < count) {
                        return arr[index];
                } else {
                        return -1;
                }
        }
};

int main(){
        OrderArray oa;

        oa.insertion(3);
          oa.insertion(4);
            oa.insertion(2);

                for (int i = 0; i < oa.size(); i++) {
                cout << oa[i] << " ";
        }
        cout << endl;
      return 0;
}
```

HOW THIS CODE WORKS?

For example we have an array of SIZE 03[3,4,2] and we have to sort them.

(1) First we will insert value 3. oa.insertion will call the function insertion and pass 3 as an argument.

- Variables i and j are initialized.
- "i" loop will be checked whether the condition is satisfied or not inside the loop. for(i=0;i<count;i++) as i is 0,count is also zero so i will also be zero . as (i<count => 0<0) is false so if condition will not be executed and loop (i) breaks.
- Now "j" loop.for(j=count;j>i;j--) as count is 0,j will be zero as well.as (j<i=> 0<0) this condition is also false and the loop will break again.
- arr[i]=val; count++; The value "3" will be inserted in arr[i] and count value will be updated to "1".

(2) Thent we will insert value 4. oa.insertion will call the function insertion and pass 4 as an argument.

- Variables i and j are initialized.
- "i" loop will be checked whether the condition is satisfied or not inside the loop. for(i=0;i<count;i++) as i is 0,count is one (i<count=>0<1) which is true then if statement executes as (arr[i]<val=3>4) is false . condition i s false.loop breaks and i will be incremented to "1".
- Now "j" loop.for(j=count;j>i;j--) as count is 1,j will be one as well.as (j<i=> 1<1) this condition is also false and the loop will break again.
- arr[i]=val; count++; The value "4" will be inserted in arr[i] (i.e IST INDEX) and count value will be updated to "2".

(3) Thent we will insert value 2. oa.insertion will call the function insertion and pass 2 as an argument.

- Variables i and j are initialized.
- "i" loop will be checked whether the condition is satisfied or not inside the loop. for(i=0;i<count;i++) as i is 0,count is two (i<count=>0<2) which is true then if statement executes as (arr[i]<val=3>2) is true then if statement breaks and loop terminates.
- Now "j" loop.for(j=count;j>i;j--) as count is 2,j will be two as well.as (j<i=> 2<1) this condition is true then (arr[j]=arr[j-1]); will execute and on 2nd index value of index 1 will be pushed "4" and then value of j will be decrement to 1 (j--) and value "3" will be stored in index 1.
- arr[i]=val; count++; The value "2" will be inserted in arr[i] (i.e 0TH INDEX) and count value will be updated to "3".

- **SAME PROCESS WILL BE REPEATED ON OTHER INSERTIONS.**

BINARY SEARCH.

```
#include <iostream>

using namespace std;

const int MAX=7;

class OrderArray{

    private:

    int arr[MAX]={1,2,3,4,5};

    int count;

    public:

    OrderArray(){

        count=MAX;

    }

    int size(){

        return count;

    }



     int search(int sval){

     int lower=0;

     int upper=count-1;

     int loc;

     while(true){

        loc=(lower+upper)/2;

        if (arr[loc]==sval)
```

```
                return loc;

                else if(lower>upper)

                return count;

                else {

                        if(arr[loc]<sval)

                lower=loc+1;

                else

                upper=loc-1;

                }

          }

      }

};


int main(){

        OrderArray oa;



              if(oa.search(63)!=oa.size())

                cout << "SEARCH SUCCESSFUL! NUMBER FOUND"<<endl;

                else

                        cout<<"SEARCH UNSUCCESSFUL! NUMBER NOT FOUND";

      return 0;

}
```

HOW THIS CODE WORKS?

=> For Binary Searching.

- for example we have 1,2,3,4,5 in an array. if the number we want to search is 3 then

if (arr[loc]==sval) will be executed (3 = 3). loc is the middle value of the array.

- If the lowest value (lower) is greater than higher number (higher) then its mean number is not present because in binary search array should be sorted.
- If the number we want to search is "4"   then if(arr[loc]<sval) will be executed (3<4) . that's means our searched value is present at right side of loc.
- If the number we want to search is "2"   then if(arr[loc]>sval) will be executed (3>2) . that's means our searched value is present at left side of loc.

DELETING A VALUE.

```
#include <iostream>

using namespace std;

const int MAX=5;

class OrderArray{

    private:

    int arr[MAX]={1,2,3,4,5};

    int count;

    public:

    OrderArray(){

        count=MAX;

    }

    int size(){

        return count;

    }



     int search(int sval){

     int lower=0;

     int upper=count-1;
}
    int loc;
    while(true){
        loc=(lower+upper)/2;
        if (arr[loc]==sval)
        return loc;
        else if(lower>upper)
        return count;
        else {
            if(arr[loc]<sval)
        lower=loc+1;
        else
        upper=loc-1;
        }
    }
}
bool deletion(int val){
  int i=search(val);
  if(i==count){
  return false;
} else{
  for(int j=i;j<count-1;j++){
        arr[j]=arr[j+1];
  }
    count--;
        return true;
}
```

```
};

};


int main(){

        OrderArray oa;



              if(oa.search(5)!=oa.size())

                cout << "SEARCH SUCCESSFUL! NUMBER FOUND"<<endl;

                else

                        cout<<"SEARCH UNSUCCESSFUL! NUMBER NOT FOUND";



                        cout<<endl;



                        if(oa.deletion(3))

                        cout<<"VALUE DELETED SUCCESSFULLY"<<endl;

                        else

                          cout<<"VALUE DELETE UNSUCCESSFULL"<<endl;

      return 0;

}
```

HOW THIS CODE WORKS?

=> For Deletion.

- for example we have 1,2,3,4,5 in an array. if the number we want to delete is 3 then
- bool function will tell whether the condition is true or false and the value wil be deleted after that condition is checked. bool function has a parameter called val.
- int i=search(val); search function will search whether the value to be delete is present in the array or not.
- search function will store the index of that value at variable "i". if(i==count) { return false;}

If the value of both i and count are same then it will return false. It means the value is not present.

for(int j=i;j<count-1;j++){ arr[j]=arr[j+1]; value of j will be same . (2=2;2<4;j++) . value of third index "4" will be stored in second index . and the value at second index which was "3" will be replaced by the succeding index.(got deleted). and loop breaks count value will be decremented to 3. It will return true and loop continues until j<count condition becomes false .

(SORTING AND BINARY SEARCHING COMBINED)

```
#include <iostream>

using namespace std;

const int MAX=7;

class OrderArray{

    private:

    int arr[MAX];

    int count;

    public:

    OrderArray(){

        count=0;

    }

    int size(){

        return count;

    }

        void insertion(int val){

    int i,j;

    for(i=0;i<count;i++){

        if(arr[i]>val)

        break;

    }
```

```
for(j=count;j>i;j--){
        arr[j]=arr[j-1];
}
  arr[i]=val;
        count++;
}
int operator[](int index) {
        if (index >= 0 && index < count) {
                return arr[index];
        } else {
                return -1;
        }
}


  int search(int sval){
  int lower;
  int upper;
  int loc;
  while(true){
        loc=(lower+upper)/2;
        if (arr[loc]==sval)
        return loc;
        else if(lower>upper)
        return count;
        else {
                if(arr[loc]<sval)
                lower=loc+1;

                else

                upper=loc-1;

                }

          }

      }

};


int main(){

        OrderArray oa;


        oa.insertion(3);

          oa.insertion(4);

            oa.insertion(2);


                for (int i = 0; i < oa.size(); i++) {

                cout << oa[i] << " ";

        }

        cout << endl;

              if(oa.search(30)!=oa.size())

                cout << "SEARCH SUCCESSFUL! NUMBER FOUND"<<endl;

                else

                        cout<<"SEARCH UNSUCCESSFUL! NUMBER NOT FOUND";

      return 0;

}
```