# Binary Search

int binarySearch(int arr[],int low,int high,int key){

    if(low<=high){

        int mid=low+(high-low)/2;

        if(arr[mid]<key) return binarySearch(arr,mid+1,high,key);

        else if(arr[mid]>key) return binarySearch(arr,low,mid-1,key);

        else return mid;

    }

    return -1;

 }

## 1.

 vector<int> find\_pos(int arr[],int N,int key){

    int pos=binarySearch(arr,0,N-1,key);

    vector<int> v(2,-1);

    if (pos==-1) return v;

    int first=pos,last=pos;

    for(int i=pos;i>=0;i--){

        if(arr[i]!=key) break;

        first=i;

    }

    for(int i=pos;i<N;i++){

        if(arr[i]!=key) break;

        last=i;

    }

    v[0]=first;

    v[1]=last;

    return v;

 }

## 2. find actual index in sorted array given rotated sorted array

brute force solution toh yeah hai ki pura array traverse karo elem ka actual index le aa aur voh index jaha uske baad wala elem chota ho dono ko minus

int find\_rotated\_sorted(int arr[],int N,int key){

    int pos,rotate=0;

    for(int i=0;i<N;i++){

        if(arr[i]==key) pos= i;

        if(arr[i]>arr[i+1] && i+1<N) rotate=i+1;

    }

    int ans=pos-rotate;

    if(ans<0) return (N+ans);

    return ans;

}

## 3. find the index of minimun ele in rotated sorted array

int index\_min(int arr[],int N){

    // size array

    if(N==1) return 0;

    // to check rotated or not if not first ele is smallest

   if(arr[0]<arr[N-1]) return 0;

   //find mid

   int low=0,high=N-1;

   while(low<=high){

        int mid=low+(high-low)/2;

        //checking mid ke adjecent me ans toh nhi hai

        if(arr[mid]>arr[mid+1]) return mid+1;

        if (arr[mid]<arr[mid-1]) return mid;

        //mid is greater than first elem and first elem is greater than last toh first se

        // mid me sarey badey elements hi honge min will be in second half

        if(arr[low]<arr[mid]) low=mid+1;

        else high=mid-1;

   }

}

## 4. find actual index of the traget in rotated sorted array

//iska graph video se

int index\_target(int arr[],int N,int key){

   //find mid

   int low=0,high=N-1;

   while(low<=high){

        int mid=low+(high-low)/2;

        //checking mid ke adjecent me ans toh nhi hai

        if(arr[mid]==key) return mid;

        //mid is greater than first elem and first elem is greater than last toh first se

        // mid me sarey badey elements hi honge min will be in second half

        if(arr[mid]>=arr[low]){

            if(key>=arr[low]&&key<arr[mid]) high=mid-1;

            else low=mid+1;

        }

        else{

            if(key>arr[mid] && key<=arr[high]) low=mid +1;

            else high=mid-1;

        }

   }

}

## 5.ind peak element

int index\_peak(int arr[],int N){

    int low = 0, high = N-1;

    while(low<=high){

        int mid=low+(high-low)/2;

        //incresing curve

        if(arr[mid-1]<arr[mid]){

            if(arr[mid]>arr[mid+1]|| mid==N-1) return mid;

            low=mid+1;

        }

        // deacreasing curve

        if(arr[mid-1]>arr[mid]){

            if(arr[mid]>arr[mid-1]|| mid==0) return mid;

            high=mid-1;

        }

    }

}

// multiple peak index same code

## 6.sreach in 2d array

### Method 1- TC O(nlogm)

bool search\_2D(vector<vector<int>> vec,int target){

    for(int i=0;i<vec.size();i++){

        if(vec[i][0]<target ){

            int low=0,high=vec[i].size()-1;

            //binary search in that row

            while(low<=high){

                int mid=low+(high-low)/2;

                if(vec[i][mid]==target) return 1;

                if(vec[i][mid]>target) high=mid-1;

                if(vec[i][mid]<target) low=mid+1;

            }

        }

    }

    return 0;

}

### Method 2- TC O(log(nm))

//dekho last ele next line ke first se chota hai agar sarey elem ko ek 1D array me convert karu sorted array hoga!

// ab 1D array space lega therefore 2D ko hi 1D ki tarha assume kao

bool search\_2d(vector<vector<int>> vec,int key){

    int n= vec.size();

    int m=vec[0].size();

    int low=0,high=n\*m-1;

    while(low<=high){

        int mid = low+(high - low)/2;;

        int X\_cor=mid/m;

        int Y\_cor=mid%m;

        if(vec[X\_cor][Y\_cor]==key) return 1;

        if(vec[X\_cor][Y\_cor]>key) high=mid-1;

        if(vec[X\_cor][Y\_cor]<key) low=mid+1;

    }

    return 0;

}

## 7.first ocurance of ele

int first\_occur(int arr[],int N,int key){

    int pos = binarySearch(arr,0,N-1,key);

    if (pos==-1) return pos;

    int ans=pos;

    for(int i=pos;i<=0;i--){

        if(arr[i]==key) ans=i;

    }

    return ans;

}

## 8. find out the sqaure root of the given non negative value of x. round it off to nearest int nearest floor int find out using binary search

//linear search TC 0(root x)

int square\_root(int N){

    int ans=-1;

    for(int i=1;i<N;i++){

        if(i\*i<=N) ans = i;

        else break;

    }

    return ans;

}

//using BS TC O(log(n))

int sqrt(int N){

    int ans=-1,low =1,high=N;

    while(low<=high){

        int mid=low+(high-low)/2;

        if(mid\*mid==N) return mid;

        else if(mid\*mid>N) high=mid-1;

        else{

            ans=mid;

            low=mid+1;

        }

    }

    return ans;

}

## 9.

// distribution from video

bool can\_distribute\_choco(vector<int> arr,int mid,int s){

    int n=arr.size();

    int studNo=1;

    int currSum=0;

    for(int i=0;i<n;i++){

        if(currSum+arr[i]>mid){

            studNo++;

            currSum=arr[i];

            if(studNo>s) return false;

        }

        else{

            currSum+=arr[i];

        }

    }

    return true;

}

int distribution(vector<int> arr,int s){

    int n=arr.size();

    int high=0,low=INT\_MAX;

    for(int i=0;i<n;i++){

        high+=arr[i];

        if(low>arr[i]) low=arr[i];

    }

    int ans=-1;

    while(low<=high){

        int mid=low+(high-low)/2;

        if(can\_distribute\_choco(arr,mid,s)){

            ans=mid;

            high=mid-1;

        } else{

            low=mid+1;

        }

    }

    return ans;

}

## 10.

distribution from video

bool can\_mini\_possible(vector<int> arr,int mid,int s){

    int n=arr.size();

    int studentRqd=1;

    int lastPlaced=arr[0];

    for(int i=0;i<n;i++){

        if(arr[i]-lastPlaced>=mid){

            studentRqd++;

            lastPlaced=arr[i];

            if(studentRqd==s) return true;

        }

    }

    return false;

}

int min\_distance(vector<int> arr,int stud){

    int n=arr.size();

    int high=arr[n-1]-arr[0];

    int low=1;

    int ans=-1;

    while(low<=high){

        int mid=low+(high-low)/2;

        if(can\_mini\_possible(arr,mid,stud)){

            ans=mid;

            low=mid+1;

        }

        else high=mid-1;

    }

    return ans;

}

## Main function

 int main(){

    //ques1

    int arr[]={1,2,3,3,3,5,11};

    int n=sizeof(arr)/sizeof(arr[0]);

    // vector <int> v=find\_pos(arr,n,5);

    // for(int i=0;i<2;i++) cout<<v[i]<<" ";

    // index in rotated array using brute force

    int arr2[]={4,5,1,2,3};

    int n2=sizeof(arr2)/sizeof(arr2[0]);

    //cout<< find\_rotated\_sorted(arr2,n2,1);

    //ques2

    //cout<<index\_min(arr2,n2);

    //ques 3

    //cout<<index\_target(arr2,n2,3);

    // ques 4/5

    int arr3[]={1,2,1,2,6,10,3};

    int n3=sizeof(arr3)/sizeof(arr3[0]);

    //cout<<index\_peak(arr3,n3);

    // ques 6

    vector<vector<int>> vec={{1,3,5,7},{10,11,16,20},{23,30,34,60}};

    //cout<< search\_2d(vec,30);

    //ques 7

    int arr4[]={2,5,5,5,6,6,8,9,9,9};

    int n4=sizeof(arr4)/sizeof(arr4[0]);

    //cout<<first\_occur(arr4,n4,9);

    //ques 8

    square\_root(21);

    //cout<<sqrt(4);

    //ques 9

    int student\_number=2;

    vector <int> vec2={12,34,67,90};

    //cout<<distribution(vec2,2);

    //ques10

    vector<int> vec3={1,2,4,8,9};

    int stud=3;

    cout<<min\_distance(vec3,stud);

 }