24/02/24

6. Searching

* List is doubly linked list.
* Iterator vector<int> :: iterator itr = a.begin();
  ↳ advance(ptr, x)

Binary Search TC: O(log n) SC: O(1) / O(log n)
(O(1): Iterative; O(log n): Recursive.)

Iterative

```
int fun(int a[], int n, int x){
    int l=0, h=n-1;
    while(l<=h){
        int m=(l+h)/2;
        if(a[m]==x)
            return m;
        if(a[m]<x)
            l=m+1;
        if(a[m]>x)
            h=m-1;
    }
    return -1;
}
```

Recursive

```
int fun(int a[], int l, int h, int x){
    if(l>h)
        return -1;
    int m=(l+h)/2;
    if(a[m]==x)
        return m;
    else if(a[m]>x)
        return fun(a,l,m-1,x);
    else if(a[m]<x)
        return fun(a,m+1,h,x);
}
```

1st occurence of a number

```
int firstocc(int a[], int n, int x){
    int l=0, h=n-1;
    while(l<=h){
        int m=(l+h)/2;
        if(a[m]>x)
            h=m-1;
```

Priyansh

TC: O(log n)
SC: O(1)

```
else if (a[m] < x)
        l = m+1;
else {
    if(m==0 or a[m] != a[m-1])
        return m;
    else
        h = m-1;
}
}
```

Last occurence of a number

```
int lastocc (int a[], int n, int x){
    int l=0, h=n-1;
    while( l<=h ){
        int m = (l+h)/2;
        if( a[m] >x)
            h = m-1;
        else if ( a[m] <x)
            l = m+1;
        else {
            if ( m==n-1 or a[m] != a[m+1])
                return m;
            else
                l = m+1;
        }
    }
```

TC: O(log n)
SC: O(1)

Square root of given number.

```
int squareroot (int n) {
    int l=0, h=n;
    int ans;
    while (l<=h) {
        int m = (l+h)/2;
        if ((m*m) == n)
            return m;
        else if ((m*m) < n) {
            l = m+1;
            ans = m;
        }
        else
            h = m-1;
    }
    return ans;
}
```

TC: O(log n)
SC: O(1)

Search in Infinite array.

```
int fun (int x, int a[], ) {
    if (a[0] == x)
        return 0;
    int i=1;
    while (a[i]<x)
        i *= 2;
    if (a[i] == x)
        return i;
    return binarysearch (a, i/2+1, i, x);
}
```

TC: O(log n)
SC: O(1)

Priyansh

```
int binarySearch ( int a[], int l, int h, int x) {
    while ( l <= h) {
        int m = (l+h) /2 ;
        if ( a[m] == x)
            return m;
        if ( a[m] > x)
            h = m-1 ;
        if ( a[m] < x)
            l = m+1 ;
    }
    return -1 ;
}
```

Search in a sorted Rotated array.

a[] = 3 4 5 1 2
x = 4

```
int fun ( int a[], int n, int x ){
    int l=0, h = n-1 ;
    while ( l <= h) {
        int m = (l+h)/2 ;
        if ( a[m] == x)
            return m ;
        if ( a[m] >= a[l] ) {
            // sorted increasing part
            if( a[l] <= x and a[m] >= x)
                h = m-1 ;
            else
                l = m+1 ;
        }
        else {
```

TC: O(log n)
SC: O(1)

Priyansh

```
        else {   // rotated part here
            if( a[h] >= x and a[m] <= x)
                l = m+1;
            else
                h = m-1;
        }
    }
    return -1;
}
```

find Peak element in a given array.

↳ Peak element: $\wedge$ element on left & right < elem

a = [1, 2, 4, 5, 3, 6]
ans = 5 or 6

```
int fun(int a[], int n){
    int l=0, h=n-1;
    if( a[0] > a[1] )
        return 0;
    if( a[n-1] > a[n-2])
        return n-2;
    while(l<h){
        int m = (l+h)/2;
        if( (m==0 or a[m] > a[m-1]) and
            (m==n-1 or a[m] > a[m+1]))
            return m;
        else if ( m!=0 and a[m] < a[m-1])
            h = m-1;
        else
            l = m+1;
    }
}
```

TC: O(logn)
SC: O(1)

Priyansh

find pair with given sum 's' in unsorted array

```
bool fun ( int a[], int n, int s ){
    map < int, int > hash ;
    for ( int i=0 ; i<n ; i++ ) {
        if ( hash [ s - a[i] ] != 0 )
            return true ;
        hash [a[i]] ++ ;
    }
    return false ;
}
```

TC: O(n)
SC: O(n)

find pair with given sum 's' in sorted array

```
bool fun ( int a[], int n, int s) {
    int l=0, h=n-1 ;
    while ( l<h) {
        if ( a[l]+ a[h] == s )
            return true ;
        if ( a[l] + a[h] < s )
            l++ ;
        if ( a[l] + a[h] > s)
            h-- ;
    }
    return false ;
}
```

TC: O(n)
SC: O(1)

~~find triplet with given sum 's' in sorted array~~ ...

Priyansh

25/02/24

Find triplet with given Sum 's' in a sorted array

```
bool triplet (int a[], int n, int s) {
    for(int i=0; i<n; i++) {
        if( pair(a, i+1, n-1, s-a[i]) )
            return true;
    }
    return false;
}

bool pair( int a[], int l, int h, int s) {
    while( l<h) {
        if( a[l] + a[h] == s)
            return true;
        else if( a[l] + a[h] < s)
            l++;
        else
            h--;
    }
    return false;
}
```

TC: $O(n^2)$
SC: $O(1)$

** If unsorted array is given, we can first sort array in $O(n \log n)$ then apply above method. → TC: $O(n^2)$ SC: $O(1)$.

Count no of pairs with a given sum.

① Unsorted array
TC: $O(n)$ SC: $O(n)$

```
int fun (int a[], int n, int s) {
    int ans = 0;
    map<int,int> hash;
```

② Sorted array
TC: $O(n)$ SC: $O(1)$

```
int fun(int a[], int n, int s) {
    int l=0, h=n-1;
    int ans=0;
```

Priyansh

```
for(int i=0; i<n; i++){
    if(hash[s-a[i]] != 0)
        ans++;
    hash[a[i]]++;
}
return ans;
}
```

```
while(l<h){
    if(a[l]+a[h]==s){
        ans++;
        l++; h--;
    }
    else if(a[l]+a[h]<s){
        l++;
    }else if(a[l]+a[h]>s){
        h--;
    }
}
return ans;
}
```

Count triplet with given Sum 's'

① Sorted array
TC: $O(n^2)$ SC: $O(1)$

```
int func(int a[], int n, int s){
    int ans=0;
    for(int i=0; i<n; i++){
        if(pair(a, i+1, n-1, s-a[i]))
            ans++;
    }
    return ans;
}
bool pair(int a[], int l, int h, int s){
    while(l<h){
        if(a[l]+a[h]==s)
            return true;
        else if(a[l]+a[h]>s)
            h--;
        else
```

② Unsorted array
TC: $O(n^2)$ SC: $O(1)$
→ Sort array → $O(n \log n)$
do same like this

Priyansh

```
        l++;
    }
    return false;
}
```

find triplet a, b, c such that $a^2 + b^2 = c^2$ [Pythagorean theorem]

* If unsorted, sort the array first.

```
bool fun(int a[], int n) {
    for(int i=0; i<n; i++) {
        a[i] *= a[i];
    sort(a, a+n);
    for(int i=n-1; i>=0; i--) {
        if(ispair(a, 0, i-1, a[i]))
            return true;
    }
    return false;
}
bool ispair(int a[], int l, int h, int s) {
    while(l<h) {
        if(a[l]+a[h]==s)
            return true;
        else if(a[l]+a[h]<s)
            l++;
        else
            h--;
    }
    return false;
}
```

TC: $O(n^2)$
SC: $O(1)$

Priyansh

video

Median of 2 sorted array

here $a = \{ \ldots \} \rightarrow n_1$

$b = \{ \ldots \} \rightarrow n_2$

$n_1 <= n_2$

```
int l = 0, h = n1;
while (l <= h) {
    int i1 = (l+h)/2;
    int i2 = (n1+n2+1)/2 - i1;

    int max1 = (i1==0) ? INT_MIN, a[i1-1];
    int min1 = (i1==n1) ? INT_MAX, a[i1];
    int max2 = (i2==0) ? INT_MIN, b[i2-1];
    int min2 = (i2==n2) ? INT_MAX, b[i2];
    if (max1 <= min2 and max2 <= min1) {
        if ((n1+n2)%2)
            return max(max1, max2);
        else
            return [max(max1, max2) + min(min1, min2)]/2;
    }
    else if (max1 > min2) {
        h = i1 - 1;
    } else if (max2 > min1) {
        l = i1 + 1;
    }
}
```

max_1	min_1
max_2	min_2

first point is

$m = (l+h)/2$

second point is.

$m = \frac{(n_1+n_2+1) - m}{2}$

TC : O(logn)

SC : O(1)

Priyansh

Repeating element → (video)

```
int sol(vector<int> a, int n) {
    int slow = a[0]+1, fast = a[0]+1;
    do {
        slow = a[slow] + 1;
        fast = a[a[fast]+1] + 1;
    } while (slow != fast);
    slow = a[0] + 1;
    while (slow != fast) {
        slow = a[slow] + 1;
        fast = a[fast] + 1;
    }
    return slow-1;
}
```

TC: O(n)
SC: O(1)

Allocate minimum pages

TC: O(n log(sum))
SC: O(1)

① Naive Recursive Approach.

```
int fmin(int a[], int n, int k){
    if(n==1)
        return a[0];
    if(k==1)
        return sum(a,0,n-1);
    int res = INT_MAX;
    for(int i=1; i<n; i++){
        res = min(res, max(fmin(a, i, k-1), sum(a,i,n-1)));
    }
    return res;
}
```

Priyansh

classmate
Date
Page

② Efficient approach

ans should lie b/w [maxm element, sum of all element] then apply binary search

if mid element is feasible ⇒ check for value < mid if there is any feasible value.

else find feasible in right half {value > mid}.

```
int findmin ( int a[], int n, int k ) {
        // here n → size of array, k → no. of people to divide
        int maxm = INT_MIN;
        int tot  = 0;
        for(int i=0; i<n; i++){
            tot += a[i];
            maxm = max(maxm, a[i]);
        }
        int l = maxm, h = tot, res;
        while(l<=h){
            int m = (l+h)/2;
            if( isfeasible (a, n, k, m)) {
                h = m-1;
                res = m;
            } else {
                l = m+1;
            }
        }
        return res;
}
```

Priyansh

```
bool isfeasible (int a[], int n, int k, int m) {
    int count = 1, cur = 0;
    for (int i = 0; i < n; i++) {
        if (cur + a[i] > m) {
            count++;
            cur = a[i];
        } else {
            cur += a[i];
        }
    }
    return (cur <= k);
}
```

binary_search (start_ptr, end_ptr, element) → true/false.

upper_bound (start_ptr, end_ptr, element) → ptr to element just greater than element.

lower_bound (start_ptr, end_ptr, element) → ptr to element if present / element just small than element.

Minimum number in a sorted rotated array.

```
int fun (int a[], int l, int h) {
    while (l <= h) {
        int m = (l+h)/2;
        if (a[m] == a[h])
            h--;
        if (a[m] > a[h])
            l = m+1;
        else
            h = m;
    }
    return a[h];
}
```

TC: O(logn)
SC: O(1)

Priyansh

classmate
Date
Page

two repeated element

```
int fun( int a[], int n){
    vector<int> ans;
    for(int i=0; i<N+2; i++){
        a[abs(a[i])%-1] *= -1;
    }
    for( int i=0; i<N; i++){
        if(a[i]>0){
            ans.push_back(i+1);
        }
    }
    return ans;
}
```

Maxm water b/w 2 building.

↓

delete (n-2) building → Watter trapped = (gap b/w building) * min(a[l], a[h])

```
int fun( int a[], int n){
    if(n<=2)
        return 0;
    int ans=0, maxm=0;
    int l=0, h=n-1;
    while(l<=h){
        ans = max(ans, (h-l-1)*min(a[l],a[h]));
        maxm = max(maxm, ans);
        if(a[l]<a[h])
            l++;
        else
            h--;
    }
```

TC: O(n)
SC: O(1)

Priyansh

```
        return maxm;
    }
```

Smallest positive missing number.

```
int missingno. (int a[], int n) {
    int p = 0;
    for (int i=0; i<n; i++) {
        if (a[i] == 1)
            p++;
    }
    if (p == 0)
        return 1;
    for (int i=0; i<n; i++) {
        if (a[i] <= 0 or a[i] > n)
            a[i] = 1;
    }
    for (int i=0; i<n; i++) {
        if (a[i] <= (n+1))
            return i+1;
    }
    return n+1;
}
```

TC: O(n)
SC: O(1)

Count only repeated.

```
int fun (int a[], int n) {
    int l = 0, h = n-1;
    while (l <= h) {
        int m = (l+h)/2;
        if (a[m] >= a[0] + m)
            l = m+1;
        else
            h = m;
    }
}
```

Priyansh

```
        if( n-a[n-1] + a[0] == 1 )
                return {-1,-1};
        else.
                return { a[l], n-a[n-1]+a[0] };
    }
```

Count more than n/k occurences

```
int fun( int a[], int n, int k) {
    map<int,int> m;
    for(int i=0; i<n; i++) {
        m[a[i]]++;
    }
    int p = n/k;  int count = 0;
    for( auto x: m) {
        if( x.second > p )
            count++;
    }
    return count;
}
```

TC: O(n)
SC: O(n)

Allocate minimum number of pages

```
int find pages ( int a[], int n, int k) {
    int l, h, ans = -1;
    l = INT_MIN;
    h = 0;
    for( int i=0; i<n; i++){
        if( l > a[i])
            l = a[i];
        h += a[i];
    }
    while ( l<=h ) {
        m = (l+h)/2;
```

Priyansh

```
        if (isvalid(a, n, k, m)) {
            ans = m;
            h = m-1;
        } else {
            l = m+1;
        }
    }
    return ans;
}

bool isvalid(int a[], int n, int k, int m) {
    int cur = 1, pages = 0;
    for(int i=0; i<n; i++) {
        if(a[i] > m)
            return false;
        if(pages + a[i] > m) {
            cur++;
            pages = a[i];
        } else
            pages += a[i];
    }
    return (cur <= k);
}
```

TC: O(n log n)
SC: O(1)

Priyansh