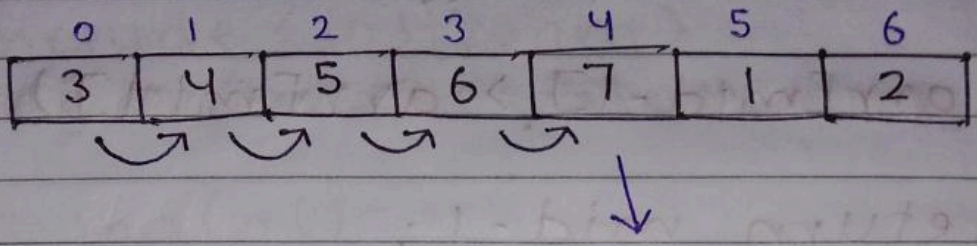
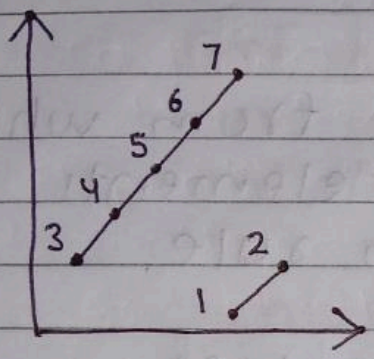
18-02-2023

Date.....

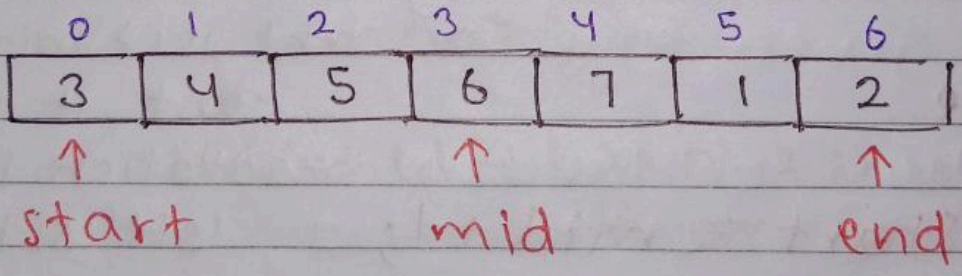
Ques: Find pivot element in an array.





after this element order is break, it means 7 is the pivot element.

If we apply Binary search on above array, we have 2 different elements which break the rule of Binary Search, i.e., 7 & 1.



Let suppose, if our mid comes of 7, then the condition would be,

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

Date.....

If our mid comes on 1, then our condition would be,

```
if(arr[mid-1] > arr[mid])
{
    return mid-1;
}
```

Above are the 2 conditions, from which we can handle the 2 odd elements which break Binary Search rule.

Now, we have to handle the left array & the right array.

```
if(arr[start] > arr[mid])
{
    end = mid-1;
}
else
{
    start = mid+1;
}
```

if start element is > mid element, it means rotation is at left side.

left array

right array

if start element < mid element, it means rotation is at right side.

code :-

```
#include <iostream>
using namespace std;

int main()
{
    int arr[] = {3, 4, 5, 6, 7, 1, 2};
    int n = 7;
    int start = 0, end = n-1;
    int mid;
    while(start <= end)
    {
        mid = (start + end) / 2;
        if((mid+1) < n && arr[mid]
                > arr[mid+1])
        {
            cout << arr[mid];
            break;
        }
        if((mid-1) >= 0 && arr[mid]
                < arr[mid-1])
        {
            cout << arr[mid]-1;
            break;
        }
        if(arr[start] >= arr[mid])
        {
            end = mid-1;
        }
        else {
            start = mid+1;
        }
    }
}
```

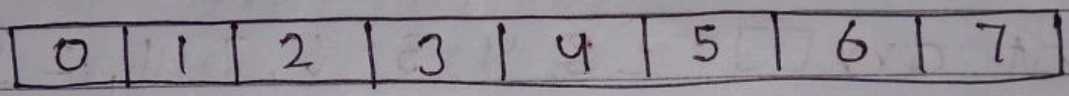
Spiral

Teacher's Sign .....

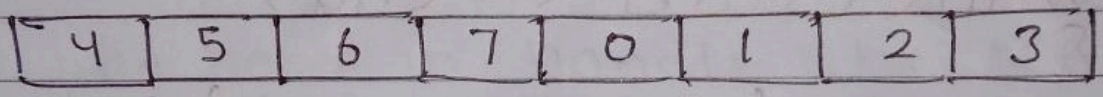
Date.....

Ques. Search in rotated & sorted array.

In this problem, we have a sorted array,



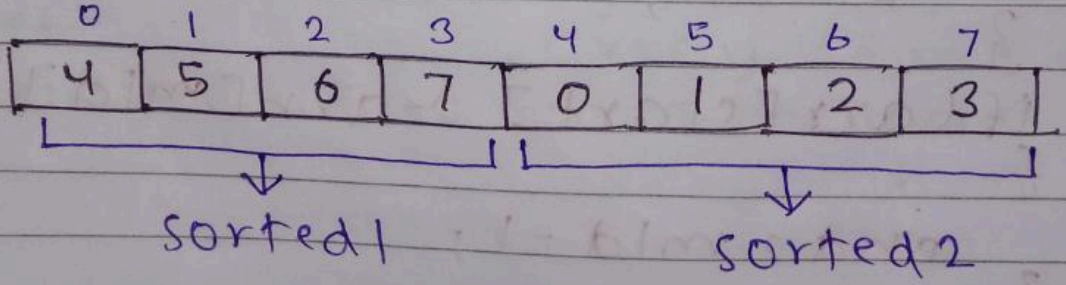
Now this array is rotated,



↳ this rotated sorted array.

In the problem, we have given a rotated sorted array & we have to find the given Key in the array.

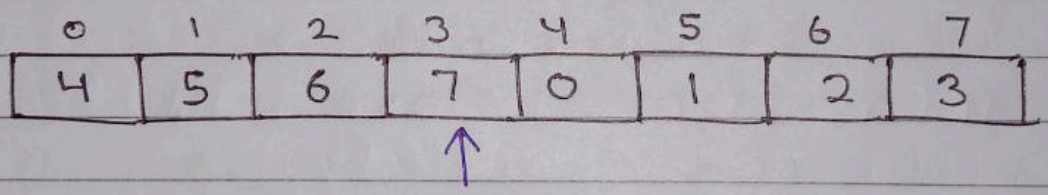
The condition of Binary search is that it applies on sorted array only, In our above array we are getting 2 sorted arrays, i.e.,



Date.....

we've find out our pivot element in the previous problem.
we can use that approach in this problem as well.

Approach :- If we find our pivot element using the previous approach then we can check that, if our given key is present in the sorted1 array, then apply Binary search on sorted 1 array.
If our given key is present in sorted 2 array, then apply Binary search on sorted 2 array.



pivot index, i.e., 3

```
if ( key >= arr[0] && key <= arr[piv])
```

↳ If this is true, it means key is present in sorted 1 array, then we apply Binary search on sorted 1 array & search the key.

Date.....

if (key >= arr[piv + 1] && key <= arr[n-1])

↳ if this condition is true, it means key is present in sorted 2 array, then we apply Binary Search on sorted 2 array & search the key.

Date.....

code :-

```
#include <iostream>
using namespace std;

int binarysearch(int arr[], int n,
    int target, int start, int end)
{
    int mid;
    while (start <= end) {
        mid = (start + end) / 2;
        if (target == arr[mid]) {
            return mid;
        }
        else if (target < arr[mid]){
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
    return -1;
}
```

↓

Binary search is applied on the parts of array to search the given target element.

```
int findPivot(int arr[], int n)
{
    int start = 0, end = n-1, mid;
    while(start < end){
        mid = (start + end) / 2;
        if((mid + 1) < n && arr[mid] >
                                arr[mid + 1]){
            return mid;
            break;
        }
        if((mid - 1) >= 0 && arr[mid] <
                                arr[mid - 1]){
            return mid - 1;
            break;
        }
        if(arr[start] > arr[mid]){
            end = mid - 1;
        }
        else {
            start = mid;
        }
    }
    return start;
}
```

↓

this function will find the index of pivot element & return that index to search() function.

Date.....

```
int search(int arr[], int n, int key)
{
    int pivotElement = findPivot(arr, n);
    if(target >= arr[0] && target <=
                    arr[pivotElement]){
        //search in 1 sorted array
        int ans = binarySearch(arr, n,
                target, 0, pivotEleme
                                -nt);
        return ans;
    }
    if(pivotElement + 1 < n && target >=
        arr[pivotElement + 1] && target
                    <= arr[n-1]){
        // search in 2 sorted array
        int ans = binarySearch(arr, n,
                target, pivotElement + 1,
                                n-1);
        return ans;
    }
    return -1;
}
```

→ search() will first find the index of our pivot element, the according to the pivot element we check that if our target lies between 0th element to pivot element then we apply Binary search on left part else apply Binary search on right part.

Teacher's Sign.....

Date.....

```
int main()
{
    int arr[] = {5, 6, 7, 8, 9, 10, 1, 2, 3};
    int n = 9;
    int target = 3;

    int targetIndex = search(arr, n,
                             target);

    if(targetIndex == -1){
        cout << "not exist";
    }
    else {
        cout << target << " is at " <<
                targetIndex << " index";
    }

    return 0;
}
```

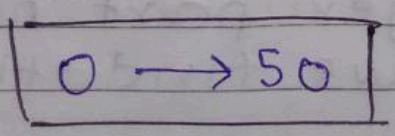
Here, we call search() to find index of our target.

Date.....

Ques. Find the square root of a number.

If we want to find the square root of 100 then it means it lies between 0 to 100.
Similarly, if n = 50, then it means square root of 50 lies between 0 to 50.

Approach:- If a square root to 50 lies between 0 to 50 then we say that our start = 0 & end = 50.
we simply calculate mid, i.e., (0 + 50) / 2 = 25.
Now, we check the square of mid, means 25 x 25 = 625.
No, we check if square of mid is equal to n, then return mid.
If square of mid is less than n, then we store the answer first, i.e., store the mid & check at the right part of the array, that is there any nearest number present to n.
But if square is greater than n then we simply check at left part, i.e., end = mid - 1.



Date.....

code:-

```
int squareRoot(int n)
{
    int start = 0, end = n, mid;
    int ans;
    while(start <= end){
        mid = (start + end)/2;
        int square = mid * mid;
        if(square == n)
        {
            return mid;
        }
        else if(square < n)
        {
            ans = mid;
            start = mid + 1;
        }
        else
        {
            end = mid - 1;
        }
    }
    return ans;
}
```

From the above code we just only find out the integer part of the square root. Now, we find the floating part also.

code :- (For Floating numbers also)

```
double squareRoot(int n)
{
    int start = 0, end = n, mid;
    int ans;
    while(start <= end)
    {
        mid = (start + end) / 2;
        int square = mid * mid;
        if(square == n){
            return mid;
        }
        else if(square < n){
            ans = mid;
            start = mid + 1;
        }
        else{
            end = mid - 1;
        }
    }
    int precision;
    cout << "enter precision: ";
    cin >> precision;
    double step = 0.1;
    double finalAns = ans;
    for(int i=0; i<precision; i++){
        for(double j = finalAns;
                j*j <= n; j += step){
            finalAns = j;
        }
        step /= 10;
    }
    return finalAns; }
```

Spiral

Teacher's Sign .....

Date.....

Ques Apply binary search on 2D matrix.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

rows (n) = 5
column (m) = 4

In linear array, start = 0 & end = n-1.
Same in Matrix case, start is at 0 & end is at last address.
In 2D array, last address is (n x m) - 1

start →

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

mid →

← end

* start = 0
end = (5 x 4) - 1
end = 20 - 1
* end = 19

mid = (start + end) / 2
mid = (0 + 19) / 2
mid = 19 / 2
* mid = 9

start = 0
end = 19
mid = 9

Date.....

In 1D array, we directly print the mid element like arr[mid].

In 2D array, how do we print mid element, in 2D array we need to give row no. & column no. to print that element, but we have only mid. So, how do we print mid element ?
we have a formula that will find that at which index our element is stored in the memory, when we have row no. (i) & column no. (j), the formula to find the memory location is,

```
c * i + j
```

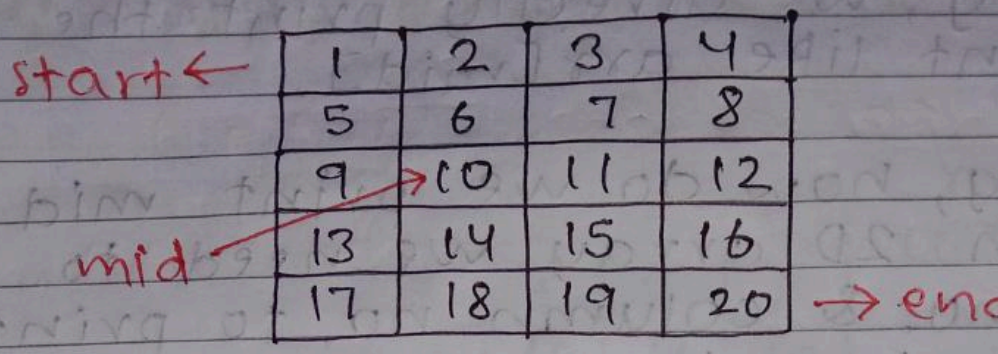
this is the formula to find memory location using row & column.

But to find row index & column index from a given memory location is,

```
row index = mid / columns
```

```
column index = mid % columns
```

Date.....



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

How to know row no. & column no. of mid element?

* mid = $\frac{0+19}{2}$ = 9

* row no. = $\frac{\text{mid}}{\text{columns}}$ = $\frac{9}{4}$ = 2

* column no. = mid % columns
= 9 % 4
= 1

So, row no. & column no. of middle element is 2, 1

code:-

```
#include<iostream>
using namespace std;

bool binarysearch(int arr[][4],
        int rows, int columns,
                    int target)
{
    int start = 0;
    int end = (rows * columns)-1;
    int mid;
    while(start <= end){
        mid = (start + end) / 2;
        int rowIndex = mid / columns;
        int columnIndex = mid %
                        columns;
        if(target == arr[rowIndex]
                    [columnIndex])
        {
            cout << "element found at"
             << rowIndex << "," <<
                columnIndex;
            return true;
        }
        else
        {
            if(target < arr[rowIndex]
                    [columnIndex])
            {
                end = mid-1;
            }
```

```
            else
            {
                start = mid + 1;
            }
        }
    }
    return false;
}

int main()
{
    int arr[5][4] = {
                        {10, 15, 20, 25},
                        {30, 35, 40, 45},
                        {50, 55, 60, 65},
                        {70, 75, 80, 85},
                        {90, 92, 96, 98}
                    };
    int rows = 5;
    int columns = 4;
    int target;
    cout << "enter target : ";
    cin >> target;

    bool ans = binarysearch(arr,
                    rows, columns, target);

    if(ans){
        cout << "found";
    }
    else {
        cout << "not found";
    }
}
```


Teacher's Sign .....