2.6 Algorithm steps

- 1. Let X be the value we wish to search for.
- 2. We use 2 variables 'start' and rend to mark beginning index and ending index
- 3. Repeat the following steps while start L= end
 - (a) We calculate 2 mids, midl and midz mid1 = (end-Start)/3 + start mid2 = 2* (end-stort)/3 tstart

The array is now divided into 3-parts

- >[Start, midl]
- -> [mid], mid2] -> [midz, end]
- (b) If X is equal to arr[mid] ar arr[midz], return indiex of x and terminate the program.
- (c) if x lies in interval [start, midi] set end = mid-1
- (d) If x lies in the interval (md1 mid2) set start=mid +1 and end = mid2-1
- (e) Else X lies in the interval [mid2, end], set start = mid2 +1 4. If x is not present in array, return -1.

Time complexity

Let N be the size of array. After the the iteration issize of array = N/3k, N/3k = 1, the program will terminate.

N/3 = 1

N = 3k

K = log(N) since the worse case requires k iterations, hence the time complexity is o (logN)

Space Complexity

The space complexity for this algorithm is O(N)

```
2.7 Algorithm Steps

Let 'low' be starting index and 'high be ending index

[tunction Find Max]

if low = high then

return arr[low]

else.

set mid \( \) (low + high)/2
```

set mid \(\left(\left(\text{low} + \text{high})/z\)

(left \(\text{Recor FindMax}(A, \low, mid)\)

Fight \(\text{recor FindMax}(A, \text{mid} + \low, \text{mid})\)

Feturn max(left(\text{mox}, \text{right})

Time Complexity
Worst Case - O(N)
Base case - O(N)

Space Complexity

the space complexity is o (log N) for recursion tall Stack.

```
while i < m1 and j < m2 and K x high repeat
       if arr[i] < arr[i] then
              if arrEij < arrEk] then
                   set final[1] & arr[1]
                   Increase of and i by 1
              else
                   set final[1] < arr[k]
                   Increased and to by 1
               [end of lift]
        2/66
              if arrEi] < arrEK] thin
                   set final[l] < arr[J]
                   Increase land juyil
               else
                   Set final[f] & AMEK]
                    Increase of and to by 1
          [End of 'if']
 [End of 'while'].
while i'm, and i'm might repeat
      if arrEis 2 arrEst. then
          Set final[4] + arr[i]
           Increase I and i by 1
 Etnd of while ]
 While I am and that high repent
        The final [4] & arr (j)
        if antill + larr[k] then
 I not of white Frencase I and 3 by 1
        else
             Set final[l] = arr[k]
              Increased and k by 1
        [end of while]
```

Step 1 : [function merge]

get i + low, j & mj, K < mz, l < low

while I < m, and K < high bepeat if arreis carreks then Set final [1] & arrEly Increased and is by Set pinol Cas & arr [K] else Increase I and I to by 1 [End of 'while') while icm, report 197111 10 11miles Set pinal Edj & arreij Inverse of and i by lend of "while" while it is repeat the state of Set final [l] - UN[j] Increase e and v by 1 [End of mince'] while kichigh report Set final (e) & arr[k] Increase I and k by I [End of (while'] Step 2: Efunction Sort Lecursive J to soil about & product of the if high low < 2 then Set mi < lan + (high-law)/3 Set m2 & low + 2 * (high-low)/3 +1 Eall 1 Gort-recurrive for index low to my call "sort - new rand for index "my to my call sert recurring for index my to high Coll function manye to perform 3 way mange sort [End of function 'Sort-recursive')

Step 3: [funtion 'sort]

if n = 0 thun

return

for i = 0 to n-1 repeat

Set duplicate[i] < arrij

[all the tunction 'sort-recursive' from indexation n-1

for i = 0 to n-1 repeat

Sot arrive duplicate(i)

[tend of function 'sort)

[tend of function 'sort)

Algorithm thospyris

F(n) = 3T(n) + 0(n)

T(n) = 0(nlog 3n)

Space complexity