Binary Search Question

Ques 1) Ceiling of a Number.

arr = [2, 3, 5, 9, 14, 16, 18] target 15.

- s=0, e=6
  m = 3 [9]
  9 < target [15]
- s = m+1 = 4 [14]
  e = 6 [18]
  $m = \frac{4+6}{2} = 5$ [16]
  16 > target [15]
- e = m-1 = 4 [14]
  s = 4
  $m = \frac{4+4}{2} = 4$ [14]
  14 < target [15]
- s = m+1 = 5 [16]
  e = 4 [14]
  As e < s
  loop break
  return s = 5 [16]

Approach :—

s = 0, end = length - 1

- Case 1 :- If mid == target
  return mid.
- Case 2 :-
  If mid < target.
  start = mid + 1.
- Case 3 :-
  If mid > target.
  end = mid - 1
- And ~~when~~ When
  end < start
  ~~ret~~ Loops Break
  So
  return start.

Date

Q2) Floor of number in array.

arr = [2, 4, 5, 9, 14, 16, 18] , target = 15

- s=0 e=6
  m = 3 [9]
  9 < target [15]
- s = m+1 = 4 [14]
  e = 6 [18]
  m = 5 [16]
  [16] m > target [15]
- e = m-1 = 4
  s = 4
  m = 4 [14]
  14 < target [15]
- s = m+1 = 5
  e = 4

As e < s
Loop Break

return e;

Approach :-

s=0 ; end = length -1

Case 1 :-
If mid == target
return mid.

Case 2 :- If mid < target
start = mid + 1.

Case 3 :-
If mid > target
end = mid - 1

Loop Breaks when
end < start.

So;
return end.

Page No.
Date

Ques:- Smallest letter Greater than target
Leet code - 744.

0 ["c" , "f" , "j"] target = "a"

0 1 2

s=0 ; e = 2
mid = 1 [f]

mid [f] > target [a]
end = mid - 1 = 0

end = 0 [c] & start = 0 [c]
mid = 0 [c]
mid [c] > target [a]
end = mid - 1 = -1
loop break
return start

Case 1:- element > target
end = mid - 1

Case 2:- element < target
start = mid + 1

loop breaks
end < start
return start

• When start > arr.length - 1
return [0].

We can use
start % arr.length.
to solve this.
as
if start = 4 ; arr.length = 4.
4 % 4 = 0
return 1st element.

0 1 2 3

0 ['x' ; 'x', y , Y] target = z

s=0 ; e = 3
mid = 1 [x]
start = 2 (mid + 1)
s = 2 e = 3
mid = 2 [y]
mid < target.
start = 3 ; end = 3
mid = 3
mid = target
start = 4
loop breaks → But 4 is out of bound
so return [0].

Page No.
Date

Q.8 Leet code - 34

Find first & last position of Element in Sorted Array.

0 1 2 3 4 5

- [5, 7, 7, 8, 8, 10] target = 8
  s e

mid = 2 [7]

mid < target

s = mid + 1

So; mid > target

end = mid - 1.

Now

s = 2+1 = 3

e = 5

mid = 4

mid = target

ans = 4

Starting

s = 3 end = 4-1 = 3

So; mid = 3

mid = target

ans = 3 [8]

mid = target

end = mid - 1

end = 2

s = 3

break.

ans = 3

ending ans = 4

s = 5 end = 5

mid = 5

mid > target

end = mid - 1

= 4

∴ Break

ans = 4

Approach

start = 0, end = arr.length - 1

- element < target
  start = mid + 1
- element > target
  end = mid - 1
- element = target
  ans = mid

for starting
end = mid - 1

for ending
start = mid + 1.

Page No.
Date

Q Searching in infinite array :-

Approach :-

- Take smaller possible box & double it every thin & search in box until we find the target

- So smallest box
  start = 0 & end = 1
  if end < target
  start = end + 1
  end = 2x (Box size) + end
  = 2x (end - start + 1) + end

- Apply Binary Search in every box
  - start & end will be found from above
    mid = start + (end - start)/2
  - mid < target
    start = mid + 1 ;
  - mid > target
    end = mid - 1 ;
  - mid = target
    return mid.

Page No
Date

Ques:- Peak Index in a Mountain array

start [0, 1, 0]

s=0 end = 2

mid = 1

mid > mid+1

end=mid

s=0 ; end = 1

mid = 0

mid < mid +1

start = mid +1

s=1 , end = 1

~~mid = 1~~ loop

break

Return start.

- ## Search in a Mountain Array.
- Find Peak from Mountain Array.

  - By Above Approach.

- [1, 2, 3, 4, 5, 3, 1]

  ascending peak descending

Apply Binarr Search from 0 to peak

So ascending

Apply Binary Search from peak+1 to end

So descending

So; Binary Search.

take start end target & array from above.

mid = start + (end - start)/2

target == mid.

return mid;

for ascending

target < mid

end = mid - 1

target > mid

start = mid + 1

for descending

target < mid

start = mid + 1

target > mid

~~start =~~

end = mid - 1.

Rotated Binary Search

arr = [2, 4, 5, 7, 8, 9, 10, 12]

1st rotation.

After 1 rotation :—

arr = [12, 2, 4, 5, 7, 8, 9, 10]

2nd rotation.

After 2 rotation

arr = [10, 12, 2, 4, 5, 7, 8, 9, 10]

Searching in Rotated Array :-

1) Find pivot in the array

2) Search in 1st half (0, piviot)

& then

Search in 2nd half (pivot+1, end)

- Search using Binary Search.

Find piviot

arr = [3, 4, 5, 6, 7, 0, 1, 2]
s m e

~~Case 1:~~ s=0 ; e=7

m=3

- m[6] < [m+1](7)
  not pivot ✗
- m[6] > [m-1](5)
  not pivot
- m[6] ≥ s[3]
  more right;
  i.e. s=m+1;

s=4, e=7

m=5

- m[0] < [m+1](1)
  not pivot ✗
- m[0] < [m-1][7]
  (m-1) is pivot.

Now apply binary search in two half

(1) s=0, e=4

(2) s=5, e=7

Approach

Case 1 = m > m+1
m = piviot

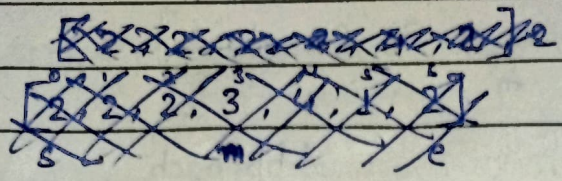
Case 2:- m < m-1
m-1 = piviot

case 3:- m < s
move left
end = mid - 1

Case 4:- m ≥ s
move right
s = m+1

Date

Ques:- Find pivot in ~~sorted~~ rotated array with duplicate values.



0	1	2	3	4	5
2	2	2	3	4	2
s		m			e

Case 2 (i) as s == m == e

check s is pivot if not then eliminate

```
{ s > s+1
    return s; }
start ++,

{ end-1 > end
    return end-1; }
end --;
```

Case 1 Check mid as a pivot

- (mid > mid+1)
  return mid;
- (mid < mid-1)
  return mid-1;

Case 2 (ii). start == mid && mid > end

So; left is same

then pivot is in right

start = mid + 1;

Case 3 (iii)

start < mid && mid == end

So; right side is same

then pivot is in left

end = mid - 1;

nothing found return -1;

Q Count the no of Rotation in Rotated array.

No of Rotation = Pivot Index + 1

Date

Ques:- Leet Code :- 410 Split Array largest Sum

[7, 2, 5, 10, 8]

Split into K = 1 (32)

• [7+2+5+10+8] = 32.

K = 2 (18)

[7] [2+5+10+8] = 25

[7+2] [5+10+8] = 23

[7+2+5] [10+8] = 18

[7+2+5+10] [8] = 24

K = 3

K = 4

K = 5 (10)

[7] [2] [5] [10] [8] = 10

So; K = 1 (32)

k = 2 (18)

k = 3 :

k = 4 .

k = 5 (10)

~~After minimize it becomes sorted in descending~~

[32, 18, ..., 10]

low = max of array

high = sum of all element of array

[10, ., ., 18, 32]

s = 10 e = 32

mid = 42/2 = 21

K = 2

No of array = 1

[7+2+5] 10 [10+8]

7 9 14 24 10 18

1 2

So 21 may be ans

then; end = mid

s = 10 e = 21 m = 15

[7+2+15]+10 [10]+8 [8]

1 14 2 3

No of Array > K

s = mid + 1;

b = 16 e = 21

m = 18

[7+2+15] +10 [10+8]
9 14 18

k = 2

e = 18

m = 17

[7+2+15] +10 [10] 8 [8]
14 10
1 2 3

No of Array > K

s = m+1

start = 18 end = 18

start = = end

So; loop

break

return start or end.

Approach :—

- Start = Max (Array)
- End = Sum (All Array elements)
- Apply Binary Search.

mid = (start + End) / 2

sum = 0 ; No of Array = 1

for

if [sum + num[i] > mid]

sum = num[i]

No of Array ++;

else {

sum = sum + num[i]

if No of Array > k

start = mid + 1

No of Array =< k

start end = mid.