30 Days Leetcode challenge - Day 16

public class leetday16

{

class Solution1 {

public int findMin(int[] arr)

{

int low = 0;

int high = arr.length-1;

int ans = Integer.MAX\_VALUE;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[low]<=arr[high])

{

ans = Math.min(ans,arr[low]);

break;

}

if(arr[low]<=arr[mid])

{

ans = Math.min(ans,arr[low]);

low = mid + 1;

}

else if(arr[mid]<= arr[high])

{

ans = Math.min(ans,arr[mid]);

high = mid - 1;

}

}

return ans;

}

}

class Solution2 {

public int singleNonDuplicate(int[] arr)

{

int n = arr.length;

if(n==1)

{

return arr[0];

}

if(arr[0]!=arr[1])

{

return arr[0];

}

if(arr[n-1]!= arr[n-2])

{

return arr[n-1];

}

int low = 1;

int high = n-2;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[mid]!=arr[mid+1] && arr[mid]!=arr[mid-1])

{

return arr[mid];

}

if((mid%2==0 && arr[mid]==arr[mid+1])||(mid%2==1 && arr[mid]== arr[mid-1]))

{

low = mid+1;

}

else

{

high = mid-1;

}

}

return -1;

}

}

}