서론

* 과제에서 묻는 문제는 n 개의 크기의 배열이 있을때 숫자원소들은 동일한 숫자원소가 2개 씩 나오고 단 한개의 숫자 원소만 한 개 나올때 해당 한개의 원소의 위치를 찾는 것이다. 이를 D/C를 이용해서 풀어야하는 문제이다.

접근 방법

* 우선 숫자원소중 한 개라는 것에 집중을 하였다. 배열의 길이는 무조건 홀수

(2k-1)인것이 자명하다. 왜냐하면 원소중 하나만 한개고 나머지는 다 2개니까. 이 성질을 잘 이용하고 D/C를 이용하면 위 문제를 해결할 수 있을 거라는 생각이 들었다.

해결방법

* 구체적인 알고리즘

예시를 들어 설명하자면 11개 길이의 배열이 있고 이때의 숫자원소를 각각 [1,1,2,2,3,3,4,4,5,6,6] 이라고 할 때, 이의 길이의 절반을 나눈다. 그러면 5.5가 나오는데 이때 나머지는 제거한다. 따라서 단지 5가 나온다. 이 때 index가 5가 딱 절반이다. Index 5를 가진 원소는 3이다. 이때 3과 같은 원소가 index를 기준으로 왼쪽에 있는지 오른쪽에 있는지에 따라 케이스를 나눈다.

1. 왼쪽에 있을 경우

위의예시처럼 index 5의 왼쪽에 있으면 index 4에 있는것이다.

이때도 케이스를 나눈다

1. Half(3)와 Half와 같은 수(3)를 제외한 왼쪽의 길이가 홀수일 경우

이 경우 왼쪽에 우리가 찾고자 하는 한 개로 구성된 숫자가 들어있는 것이 자명함으로 end를 half-2로 진행하여 범위를 좁힌 다음 다시 위 함수를 호출한다.

1. Half와 Half와 같은 수를 제외한 오른쪽의 길이가 홀수일 경우

이 경우 오른쪽이 우리가 찾고자 하는 한개의 숫자가 들어있는것이 자명하므로 start 를 half+1 로 해주어 범위를 좁힌다음 다시 위 함수를 호출한다.

1. 오른쪽에 있을 경우

위와 같이 index 5의 오른쪽에 있으면 index 6에 있는 것이니 이때 도 케이스를 나눈다.

1. Half와 Half와 같은수를 제외한 왼쪽의 길이가 홀 수 인경우

이 경우 왼쪽에 우리가 찾고자 하는 한개의 숫자가 들어있는것이 자명함으로 end를 half-1로 진행하여 범위를 좁힌 다음 다시 위 함수를 호출한다.

1. Half와 Half와 같은수를 제외한 오른쪽의 길이가 홀 수 인경우

이 경우 오른쪽에 우리가 찾고자 하는 한개의 숫자가 들어있는것이 자명함으로 start를 half+2로 진행하여 범위를 좁인다음 다음 다시 위 함수를 호출한다.

3. Half가 그 한개인 index일 경우

해당 함수를 다시 call 할 필요없이 바로 해당 index를 출력하고 종료한다.

이렇게 범위를 좁히다가 end와 start의 차이가 2거나 0인 경우를 trivial 하다고 가정한다. 0 같은경우는 start가 해당 index가 된다

End와 start의 차이가 2인 경우에는 half를 기준으로 왼쪽과 오른쪽을 비교 하여 다른 하나를 찾아 답을 찾으면된다.

* 수도코드

list\_of\_num = []

def findA(list\_of\_num,start,end):

length=len(list\_of\_num)

half = (start+end)//2

if end==start :

print(“index: ",start)

elif end-start ==2:

if list\_of\_num[half] == list\_of\_num[half-1]:

ans= list\_of\_num[half+1]

print("index : ",half+1)

elif list\_of\_num[half] == list\_of\_num[half+1]:

ans = list\_of\_num[half-1]

print("index : ",half-1)

else :

if list\_of\_num[half] == list\_of\_num[half-1]:

if len(list\_of\_num[0:half-1])%2 == 1:

end=half-2

findA(list\_of\_num,start,end)

else:

start=half+1

findA(list\_of\_num,start,end)

elif list\_of\_num[half] == list\_of\_num[half+1]:

if len(list\_of\_num[0:half])%2 == 1:

end=half-1

findA(list\_of\_num,start,end)

else:

start=half+2

findA(list\_of\_num,start,end)

else:

print("index : ", half)

test case

1. Length가 11인 [1,1,2,2,3,3,4,4,5,5,6] 경우 우선 half로 가른다. 따라서 (O(1)) 그 후에 왼쪽과 같은지 오른쪽과 같은지 비교 (O(1)) half와 half와 같은 수를 제외하고 이를 기준으로 왼쪽이 홀 수 인지 오른쪽이 홀수 인지 비교(O(1)) 그렇고 다시 함수 호출한다. 그러면 위에 알고리즘에 따라 범위가 index 6부터 index10 까지 줄었다. 다시 [4,4,5,5,6]에서 half인 8(숫자로는 5)를 뽑고 위 과정을 반복하면 start 와 end가 동일한 10이고 다시 함수를 호출한다 start와 end가 동일하므로 if문에 걸려 start가 출력된다. 따라서O(logn)에 끝나는 것을 알 수 있다.
2. Length가 3인경우 [1,1,2] 경우

위 경우는 end와 start의 차이가 2이므로(start= 0, end =2 이므로 )if 문에 걸려 양끝과 half를 비교한다음 다른 하나의 index를 print 한다.

1. 가운데가 다른 경우 [1,1,2,3,3]

위 경우는 마지막 else 문에 걸려 바로 index가 나온다.

결론

* 위 문제는 정렬된 배열에서 모든 숫자원소의 갯수가 2개씩나오고 오직 한개의 원소만이 갯수가 하나인 숫자원소가 나올때 이것의 index가 몇인지를 찾는 문제였고 이를 D/C를 이용하여 Time Complexity가 O(logN) 인 방법으로 해결 하였다.

위 문제의 해결방법을 단순히 생각해봤을 때 가장 쉬운 방법은 한개씩 비교해보는 방법일 것이다 예를 들어 flag를 놓고 한개의 숫자를 만나면 flag를 1로 하고 그 후에 같은 숫자가 나오면 flag를 0으로 놓고 만약 이때 숫자를 옮기는데 flag가 1인 상태가 2번이상 지속되면 해당 index가 우리가 찾고자 하는 index일 것이다. 위 방법은 모든 숫자를 다 비교해봐야 함으로 O(n)이 걸린다. 평소 recursive와 선택알고리즘의 mechanism의 정확한 방식이 헷갈렸는데 (반으로 나누고 나머지 반은 신경쓰지 않고 반절만 신경쓰는 방식)이를 직접 코딩하고 수도코드를 구현해 봄으로써 명확히 이해가 갔다. 위 알고리즘을 응용하여 짝이 필수적인 “에어팟”과 같은 무선이어폰의 불량 상태등을 확인할 수 있는 프로그램으로 응용시켜볼 수 있겠다.