Peseudocode of recursive solution

```
Algorithm  Distinct (arr,n
        1. Heap-Sort(arr,n)
        2. variable <- arr[0]
        3. c <--  1
        4. i <-- 1
        5.  return Count_distinct(arr,i,variable,c,n)
```

Time complexity of Heap-Sort function is O(n log n)

And Time complexity of Count_distinct is O(n)

So Time complexity of Distinct is Max(O(n log n), O(n))

--> Time complexity of this function is O(n log n)

```
Algorithm  Heap-Sort (arr,n)
    1.  Build-max-heap(arr,n)
    2.  for i <-- n-1 downto 1
    3.          do temp <-- arr[0]
    4.          arr[0]=arr[i]
    5.          arr[i]=temp
    6.          Heapify(arr,i,0)
```

Time complexity of Build-max-heap is O(N)

Time complexity of Heapify is O(log n)

So Time complexity of this function is O(n log n)

Algorithm Build-max-heap(arr,n)

```
1.  for i <-- n/2-1 downto 0
2.          do heapify(arr,n,i)
```

Time complexity of Build-max-heap is O(N)

Algorithm Heapify (arr,n,i)

```
1.  Mx <-- i
2.  Left <-- 2*i+1
2.  Right <-- 2*i+2
4. If Left <n and arr[Left]>arr[mx]
5.      Then Mx <-- Left
6. If Right <n and arr[r]>arr[mx]
7.      Then mx <-- Right
8. If mx not equal i
9.      Then excharge arr[i] <--> arr[mx]
10.     heapify(arr,n,mx)
```

Time complexity of Heapify is O(log n)

```
Algorithm Count_distinct (arr,i,variable,c,n)
1. If i > n-1
2.      Then  return c
3. else
4.      do
5.      If variable not equal arr[i]
6.           then c++
7.           Variable <-- arr[i]
8.           count_distinct(arr,i+1,variable,c,n)
```

Time complexity of Count_distinct is $t(n)= t(n+1) +c$

So Time complexity of Count_distinct is $O(n)$