Peseudocode of recursive solution

Algorithm Distinct (arr,n)

1. Heap-Sort(arr,n) -->O(n)

2. variable <- arr[0] -->c

3. c <-- 1 -->c

4. i <-- 1 -->c

5. return Count_distinct(arr,i,variable,c,n) --->O(n)

ANALYSIS

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)

complexity

best case O(n lgn)+c= O(n lgn)

worst case O(n lgn)+O(n)= max(n lgn, n)= O(n lgn)

avg case O(n lgn)+O(n)= max(n lgn, n)= O(n lgn)

Algorithm Heap-Sort (arr,n)

1.Build-max-heap(arr,n) -->O(n)

2.for i <-- n-1 downto 1 -->O(n)

3.do temp <-- arr[0] -->c

4.arr[0]=arr[i] -->c

5.arr[i]=temp -->c

6.Heapify(arr,i,0) --> O(log n)

ANALYSIS

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)

Complexity

best case O(n)+O(n)+O(1)+O(logn)= O(n)

worst case O(n)+O(nlgn)= O(nlgn)

avg case O(n lgn)+O(n)= max(n lgn, n)= O(n lgn)

Algorithm Build-max-heap(arr,n)

1.for i <-- n/2-1 downto 0 -->O(n)

2.do heapify(arr,n,i) -->O(log n)

ANALYSIS

Σ i*i

The tight is O(n)

Complexity

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

Worst = best = avg = O(n)

Algorithm Heapify (arr,n,i)

1. Mx <-- i -->c

2. Left <-- 2*i+1 -->c

2. Right <-- 2*i+2 -->c

4. If Left <n and arr[Left]>arr[mx] -->c

5. Then Mx <-- Left -->c

6. If Right <n and arr[r]>arr[mx] -->c

7. Then mx <-- Right -->c

8. If mx not equal i -->c

9. Then excharge arr[i] <--> arr[mx] -->c

10. heapify(arr,n,mx) -->O(logn)

ANALYSIS

Check by using if condition it take constant

And call heapify resursivly

The total take O(log n)

complexity

<u>best case =worst = avg = O(log n)</u> ---

Algorithm Count_distinct (arr,i,variable,c,n)

1. If i > n-1 -->c

2. Then return c   -->c

3. else   -->c

4. do   -->c

5. If variable not equal arr[i] -->c

6. then c++   -->c

7. Variable <-- arr[i]   -->c

8. count_distinct(arr,i+1,variable,c,n)   --> t(n+1)

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

ANALYSIS

T(n)=t(n+1)+c   →k=1

T(n+1)=t(n+2)+c →t(n)=t(n+2)+2c →k=2

T(n+2)=t(n+3)+c → t(n)=t(n+3)+3c →k=3

General form is t(n)=t(n+k)+k.c

N+k=1 so k=1-n

t(n)=t(n+1-n)+(1-n).c

t(n)=t(1)+c-c.n

so

t(n)=O(n)

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

complexity

So Time complexity of Count_distinct is O(n)