Jay Kalyanbhai Savani

20009207

ASSIGNMENT 1

**Reinforcement**

**Graphical user interface, text

Description automatically generated**

Ans:

The order from lower to higher functions:

1. 1/n
2. 2100
3. log log n
4. sqrt(log n)
5. log2n
6. n0.01
7. sqrt(n), 3 n0.5
8. 2log n, 5 n
9. n log4n, 6 n log n
10. 2 n log2 n
11. 4 n3/2
12. 4log n
13. n2 log n
14. n3
15. 2n
16. 4n
17. 22n

Graphical user interface, text, application, email

Description automatically generated

Ans: Since the Find2D approach is a quadratic one rather than a linear one, its worst-case runtime complexity is O(n2). By analyzing the worst scenario, in which element x is the very last element to be inspected in the n\*n array. Find2D in this situation repeatedly uses arrayFind. Then, until the last element where x is located, arrayFind will have to search through all n items for every call. For each arrayFind call, n comparisons are made. Because of this, our running time is O(n2) for n\*n operations.

Graphical user interface, text, application, email

Description automatically generated

Ans: If there is a constant n0 >= 0 for any constant c > 0 and n >= n0, then we say that n is o(n log n), and n < c \* n log n for n >= n0.

Therefore, using the formula 1/c log n, we pick n0 = 21/c + 1. (When the log is the base of 2).

Graphical user interface, text, application, email

Description automatically generated

Ans: Let c > 0 be any constant; there is a constant n0 > 0 such that n2> cn. This one will imply that n2 is w(n). So, n > c. Yet another approach is n0 = c + 1.

Graphical user interface, text, application, email

Description automatically generated

Ans: Finding a constant c > 0 and a constant n0 >= 1 that causes n3 log n to exceed cn3 will allow us to demonstrate the expression above.

The options are c = 1 and n0 = 2. (Assume log is the base of 2).

Graphical user interface, text, application, email

Description automatically generated

Ans: Based on Chernoff Bounds,

µ = E(X) = n \* (1/n1/2)= n1/2.

Then for δ =2, the upper bound is

Pr[X>(1+ δ)u] < (eδ/(1+δ)(1+δ))u => Pr(X > 3µ) <

**Creativity**

Graphical user interface, text, application, email

Description automatically generated

Ans: t=time to change every single bit

let k=total bits.

The total work is:  
t \* (n/20 + n/21 + n/22... + n/2k) < t \* n \* 2 => O(n)

The total running time is O(n).

Graphical user interface, text, application, email

Description automatically generated

Ans: T(n): T(n) = 2 \* T(n-1) = 2 \* 2 \* T(n-2) = ... = 2 \* 2n-1 \* T(0) = 2n

Graphical user interface, text, application, email

Description automatically generated

Ans: Assume that the base to all log used is 2.

Upper Bound Summation

(log i) = log(1) + log(2) + .... + log(n)

log(n) + log(n) + ...... log(n)

n \* log(n)

Lower Bound Summation

(log i) = log (1) + ... + log(n/2) + ... + log(n)

log(n/2) + ... + log (n)

log(n/2) + ... + log(n/2)

n/2 \* log (n/2)

summation is O(n log(n)).

Graphical user interface, text, application, email

Description automatically generated

Ans: The size of the array is expanded from N to N + ⌈N1/2⌉

Based on the amortization, each insertion will cost (N+ N1/2)/ N1/2 = 1+ N1/2 cyber dollars ($).

The total insertion cost:

∑ 1+ 1+SQRT(N) = ∑ 2+SQRT(N)= 2n + ∑ SQRT(N) from N=1 to N=n

that we can get is no more than:  
(2/3)n3/2+ (1/2)n1/2- 1/6

but no less than  
(2/3)n3/2+ (1/2)n1/2+ 1/3 - (1/2)21/2.

The total cost of the array operation is θ(n3/2).

**Application**

Graphical user interface, text, application, email

Description automatically generated

Ans: Algorithm reversal (start, end, n, A)

Initialize an array with n elements by starting at 0 and ending at n-1 in step one.

The Second Step is Swap in a loop (arr[start], arr[end])

The third step is Start Start + 1 End End -1.

There will be a time complexity O(n)

The preceding method will run through Step 1 in constant time (1), Step 2 in constant time, swapping in constant time, and Step 3 again in constant time (1). Consequently, the algorithm's overall running time is O(n). Because it only accesses each element of the array once, this algorithm would execute in O(n) time. It simply requires two variables to store the pointers, therefore there is no need for additional storage. The space complexity is therefore O(1).

Graphical user interface, text, application, email

Description automatically generated

Ans:

Input: An array A of n-bits, indexed from 1 to n.

Output: The shortest subarray of A that contains k 1’s.

Count 0

k 0 //maximum found so far

for i 1 to n do

if A[i] = 0 then

count 0

else

count count + 1

k max(k, count)

return k

Run Time:

Count 0 1 time

k 0 1 time

for i 1 to n do n times

if A[i] = 0 then 1 time

count 0 1 time

else 1 time

count count + 1 1 time

k max(k, count) 1 time

return k 1 time

Total Run Time = 1 + 1 + n + 1 + 1 + 1 + 1 + 1 + 1

= n + 8

= O(n)

For loop will take n times to run whereas if and else statement will run in constant O(1) time. All others are variables that will take constant time O(1) to run.