Fatih Selim YAKAR
161044054

- **Question 1 Complexity Analysis – Black-White Boxes**

There are 2n boxes. There are 2 start point in the code, calls low and middle. Algorithm starts the low(first index) and middle(index of length of array/2) then decrease the domain set by 2 recursively up to middle equal to length of boxList.

So the general complexity is: $T(n) = T(n-2) + 1$

$T(n-2)$ is came decrease section.
+1 is came swap section.

$Assume\ T(0) = 0$
$T(2) = T(0) + 1$
$T(4) = T(2) + 1$
$T(6) = T(4) + 1 \rightarrow T(6) = T(0) + 3$

In generalize the recurrence relation came up: $T(n) = T(0) + n/2 \rightarrow T(n) = n/2$

The complexity is $\boldsymbol{T(n)} \in \boldsymbol{O(n)}$

There are no possibility in this situation because of that the worst, best and average case is same $\boldsymbol{O(n)}$

- **Question 2 Complexity Analysis – Fake Coin Problem**

I used binary search based algorithm in this section. I measure the weight by dividing two equal coins If one side of the weightbridge is lighter, it enters recursive until it finds the fake coin.

Average Case:

The probability of successfull search is p $0 \leq P \leq 1$

Assume that this recurrence relation is tree. So there are n internal node and n+1 leaf node. Probability that coin can be found in any position is $\frac{P}{n}$.

Internal Path Length(IPL) : The sum of the length of the paths from the root to the interval nodes.
Leaf Path Length(LPL) : The sum of the length of the paths from the root to the leaf node.

Probability of finding = $\frac{P}{n}(IPL(T) + n) \rightarrow$ Average number of comparisons for internal paths

Probability of not finding = $\frac{1-P}{n+1}LPL(T)$ →Average number of comparisons for leaf paths

$A(n) = \Sigma T(I) * P(I) = \frac{P}{n}(IPL(T)+n) + \frac{1-P}{n+1}LPL(T)$ → Total average case

For a 2 tree T, $IPL(T) = LPL(T) - 2I$, where I is the number of internal nodes for any binary tree T, $LPL(T) \geq L * \lfloor \log_2 L \rfloor + 2(L - 2^{\lfloor \log_2 L \rfloor})$ when L is the number of the leaf nodes

$$A(n) = \frac{P}{n}(IPL(T)+n) + \frac{1-P}{n+1}LPL(T)$$

$$= \frac{P}{n}(LPL(T)-n) + \frac{1-P}{n+1}LPL(T)$$

$$= \left(\frac{P}{n} + \frac{1-P}{n+1}\right)LPL(T) - P$$

$$\geq \left(\frac{P}{n} + \frac{1-P}{n+1}\right)\left[(n+1) * \lfloor \log_2(n+1) + 2\left((n+1) - 2^{\lfloor \log_2(n+1) \rfloor}\right)\right] - P$$

$$\rightarrow \left(\frac{P}{n} + \frac{1-P}{n+1}\right) = \frac{1}{n}$$

$$\rightarrow \left[(n+1) * \lfloor \log_2(n+1) + 2\left((n+1) - 2^{\lfloor \log_2(n+1) \rfloor}\right)\right] = \mathrm{n}\log_2(n)$$

$$\boldsymbol{A(n) = \frac{1}{n}\mathbf{n}\log_2(n) \in \vartheta(logn)}$$

Best Case:

If n is odd and fake coin is last element: $\boldsymbol{B(n) \in \vartheta(1)}$ except this
If n is even coins: $\boldsymbol{B(n) \in \vartheta(logn)}$

Worst Case:
Suppose that there are n element and $n = 2^k - 1$
There are k comparisons in the worst case complexity and $k = \log_2(n+1)$

$\boldsymbol{W(n) \in \vartheta(logn)}$

Fatih Selim YAKAR
161044054

- **Question 3 Complexity Analysis – Insertion and Quick Sort**

Insertion Sort's Average Case:

Let Ti be the number of basic operations at step i, where $1 \leq i \leq n-1$

$T = T1 + T2 + \cdots \ldots. + Tn - 1 = \sum Ti\ from\ 1\ up\ to\ n-1$

$A(n) = E(T) = E(\sum_{i=1}^{n-1} Ti)$ (E is Expected)

$$E[Ti] = \sum_{j=1}^{i} j\,Prob(Ti = j)$$

$$Prob(Ti = j) = \begin{cases} \dfrac{1}{1+i}\ if\ 1 \leq j \leq i-1 \\ \dfrac{2}{i+1}\ if\ j = i \end{cases}$$

$$E[Ti] = \sum_{j=1}^{i-1}(j * \frac{1}{i+1}) + i * \frac{2}{i+1} = \frac{i}{2} + 1 - \frac{1}{i+1}$$

$$A(n) = E[T] = \sum_{i=1}^{n-1}(\frac{i}{2} + 1 - \frac{1}{i+1}) = \frac{n(n+1)}{4} + (n-1) - \sum_{i=1}^{n-1}\frac{1}{i+1}$$

$A(n) = \frac{n(n+1)}{4} + n - H(n) \in \boldsymbol{\vartheta}(\boldsymbol{n^2})$

Quicksort's Average Case:

$T = T1 + T2$

Ts are random variables
$T1$ is number of operations in rearrange
$T2$ is number of operations in recursive calls

$A(n) = E[T] = E[T1] + E[T2]$

$T1\ is\ fixed: n + 1$
$T2\ depends\ on\ where\ the\ pivot\ has\ been\ placed$

Fatih Selim YAKAR
161044054

E[T]=(n+1) +$\sum_{i=1}^{n} E[T2 \mid x = i] * P(x = i)$

$$= (n+1) + \sum_{i=1}^{n} (A[i-1] + A[n-i]) * \frac{1}{n}$$

$$= (n+1) + [A(0) + A(n-1) + \cdots + A(n-1) + A(0)] * \frac{1}{n}$$

$$= (n+1) + \frac{2}{n}[A(0) + \cdots + A(n-1)]$$

$$n.A(n) - (n-1)A(n-1) = nA(n) - A(n-1)(n-1+2) = 2n$$

$$\frac{A(n)}{n+1} - \frac{A(n-1)}{n} = \frac{2}{n+1}$$

$$Change\ the\ variable\ t(n) = \frac{A(n)}{n+1}$$

$$so\ the\ general\ recurrence\ is \rightarrow t(n) = t(n-1) + \frac{2}{n+1}$$

$The\ recurrence's\ solution$:

$$t(0) = \frac{A(0)}{1} = 0$$
$$t(n) = t(n-1) + \frac{2}{n+1}$$
$$t(n) = t(n-2) + \frac{2}{n+1} + \frac{2}{n}$$
.
.
.
$$t(n) = \sum_{i=1}^{n} \frac{2}{i+1}$$
$$t(n) = 2H(n+1) - 3$$
$$A(n) = H(n)(n+1)$$
$$\boldsymbol{A(n) = 2(n+1)H(n) - 3(n+1) \in \vartheta(n\log n)\ from\ harmonic\ series}.$$

Number of swaps:

Sort type	Arr[10,9,8,7,6]	Arr[10,9,8,7,6,5]	Arr[10,9,8,7,6,5,…,1]
Quicksort	8	11	29
Insertion Sort	10	15	45

It seems clearly that the quicksort insertion is increasing much slower than the insertion sorting. This fits the complexity.

- **Question 4 Complexity Analysis – Find Median**

Worst Case:
This algorithm based on insertion sort and the remaining parts are constant time so we can this algorithm's worst case is insertion sort's algorithm.

The worst case (maximum number of basic operation) occurs if the list is already sorted in reverse order.

$$W(n) = \sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2} \in \vartheta(n^2)$$

- **Question 5 Complexity Analysis – Find Optimal Sub Array**

Traverse recursively the all of combinations 1 to n of n. First control the sum of that combination.If greater than spesific calculated value, checks if it is the smallest. If it is small it replaces it with template "Min" variables. Thus, it finds the optimal subarray.

The worst case is based on maximum number of basic operation. So all loops within the function should work as many times as possible.

- Outer for loop runs **n** times (n is number of element in list)
- The combination function's complexity is: $\theta[r\ (n\ choose\ r)]$
- Recursion is calls $\sum_{i=1}^{n}\binom{n}{i} = \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \dots + \binom{n}{n}$ times equal to $2^n - 1$
- Inner in recursion there are sum and multiply function: i+i complexity
- So total complexity is $[\sum_{i=1}^{n}\binom{n}{i} * 2i] + n * \theta[r\ (n\ choose\ r)]$

$$\rightarrow [[\sum_{i=1}^{n}\binom{n}{i} * \cancel{2}i]] \leq [[\sum_{i=1}^{n}\binom{n}{i} * n]]$$

$$\rightarrow So\ [[\sum_{i=1}^{n}\binom{n}{i} * i]] \in O(n(2^n \cancel{-1})$$

$$\rightarrow O(n2^n) + n * \theta[r\ (n\ choose\ r)]] \in \boldsymbol{O(n2^n)}$$