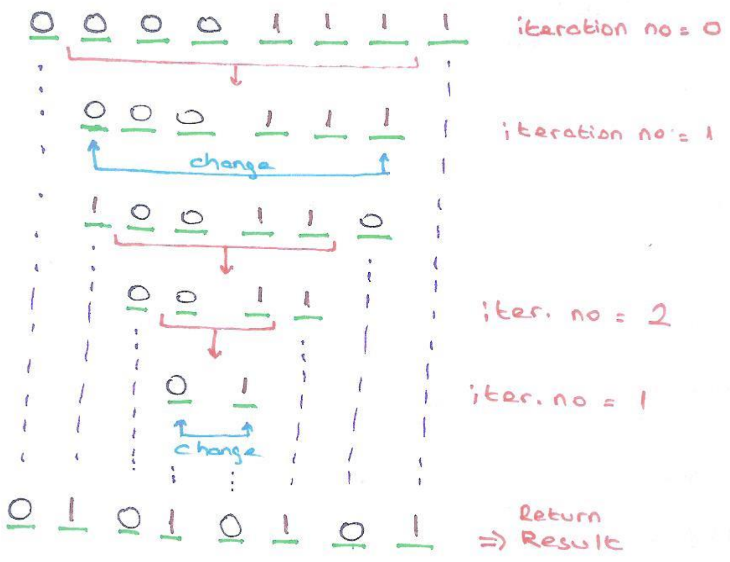
Yağız DÖNER
141044062

CSE 321 - HW3 - REPORT

PART 1



0's are black boxes.
1's are white boxes.

⇒ I count the number of iteration which start with 0. If iteration is an even number, I call the function recursively without the first and last boxes. If Iteration is an odd number, I change the first and last boxes of the sub-array and then call the function recursively without first and last boxes. In every recursive call, iteration number is increased. At the end, I combine the results.

Best Case : If there are 2 boxes, Best case is occur.
$B(n) = O(1)$ = constant time.

Worst - Average Cases : In every iteration, I decrease the subarray with 2.

$T(n) = T(n-2) + 1$

$T(n-2) = T(n-4) + 1 + 1$

$T(n-4) = T(n-6) + 1 + 1 + 1$

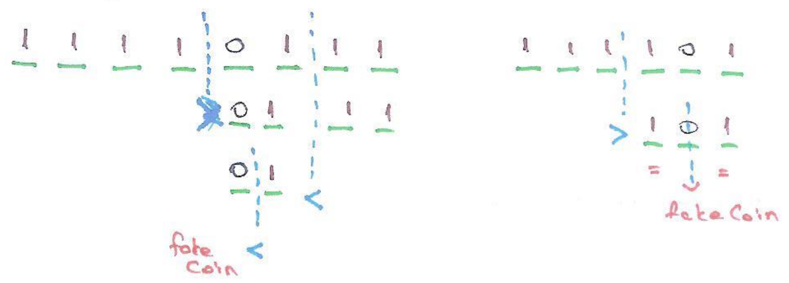
$\vdots$

$T(n) = T(n-2k) + k \Rightarrow T(n) = T(1) + \frac{n+1}{2}$

$k = \frac{n+1}{2} \quad T(1) = 1$

$T(n) = O(n)$

PART 2



0 is fake coin.

⇒ Take one coin from the coins, If a half of the remaining coins equal to other part of the remaining coins, the fake coin is the coin that taken at the beginning.
If two parts not equall together, leighter part is calling recursively.

Best Case ; the number of coins are smaller than 3.

$B(n) = O(1)$ = constant

Worst - Average Cases ;

$a = 1$, $b = 2$, $d = 0$

$b^d = a$

$T(n) = T(n/2) + 1$ From M.T ⇒ $n^{d}\lg n$ ⇒ $O(\lg n)$

PART 3

⇒ I add the swap-counter in the both insertion-sort and quick-sort. Number of swap operation results are shown by the python code.

Average case of QuickSort ⇒ $T(n) = T(i) + T(n-i) + cn$

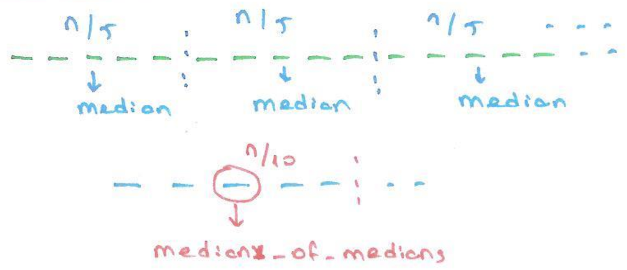
$T(n) = O(n\lg n)$

Average case of InsertionSort ⇒ 2 loops , ⇒ $O(n^2)$

→ In my results, when the number of elements increase, the swap operation of the insertion sort increase rapidly.

→ In theoritical, n^2 is bigger than $n\lg n$.

PART 4



⇒ For this part, I use the median of medians algorithm. This function take the median of medians as a pivot and then divide the list according to pivot.

→ there are two elements smaller than the medians of medians, and also there are two elements smaller than these two elements.

$\searrow \frac{3n}{10} \to$ remaining $\frac{7n}{10}$

$$T(n) = T(n/5) + T(7n/10) + O(n)$$

$$T(n) \leq 10 \cdot c \cdot n \in \underline{O(n)}$$

PART 5

! NOTE = In PDF, $sum(B) \boxed{\leq} (max(A) + min(A)) \cdot \frac{n}{4}$

But, according to the examples in the PDF, i change it like
$sum(B) \boxed{>} \dots$

⇒ In every iteration, function is called recursively with decreasing 1.

$$T(n) = T(n-1) + 1$$
$$T(n-1) = T(n-2) + 1 + 1$$
$$T(n-2) = T(n-3) + 1 + 1 + 1$$

$$T(n) = T(n-k) + k$$
$$k = n-1, \quad T(1) = 1$$
$$T(n) = T(1) + n - 1$$
$$\underline{T(n) \in O(n)}$$

⇒ Firstly, I order the list descending order.
If, Summation of the elements are smaller than formula, I take another element. If the summation is bigger, I take another element which is summation is smaller than other summation but bigger than formula.