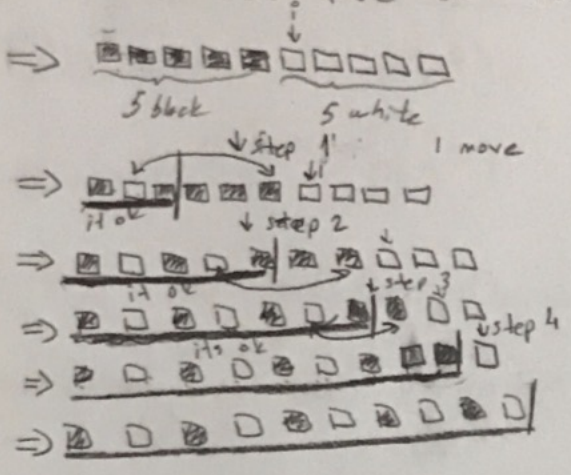
① Assume that there are 10 boxes



there are $n/2$ compare for each insider loop.

so $\frac{n}{2} + \frac{n-1}{2} + \frac{n-2}{2} \cdots = n$

so insider loop is $O(n)$

then there $n/2$ iterator loop so

for outer loop it is $O(n)$ again

So we can say that the Time Complexity is $n \times n = O(n^2)$

For best case: Best case means for this algorithm the pattern already sorted like black-white-black-white so only outer loop would work so, Best Case = $O(n)$

For average case and worst Case: They are equal for this algorithm, I have already explained it is $O(n^2)$

---

② There are n coins and for my solution the fake one is lighter. there are 2 types way in algorith belong to n is odd or even. If it is odd, checks the left side and right side of middle elements heavyly equal. If they are fake is middle coin, else continue with lighter side. So in each operation the n devides by 2 so we can say its $O(\log n)$.

For best case: if the fake coin is at middle $O(1)$

for average case: if the fake coin is at somewhere in list it is $O(\log n)$

For worst case: Same with average case it is $O(\log n)$

(3)

Algorithm	Worst Case	Best Case	Average Case	Count for my test
Quick Sort	$O(n^2)$	$O(n \log n)$	$O(n \log n)$	8
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$	19

For my solution Quick Sort is better. But if array is already sorted Insertion Sort is better.

Insertion Sort is suitable for for small files, but again it is an $O(n^2)$ algorithm, but with a small constant. Also it work best when the array is already sorted.

Quick Sort looks better. It is an $O(n * \log(n))$ algorithm on average case and an $O(n^2)$ algorith in the worst case screnario If the array was already sorter worst case occurs for Quick Sort

~ Insertion Sort Avarge Case ~

$$P(T_i = j) = \begin{cases} \frac{1}{i+1} & \text{if } 1 \le j \le i-1 \\ \frac{2}{i+1} & \text{if } j = i \end{cases}$$

$$\frac{i(i+3)}{2(i+1)} = \frac{1}{2} + 1 - \frac{1}{i+1}$$

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

$$= \frac{n \cdot (n-1)}{4} + n - 1 - \sum_{i=1}^{n-1} \frac{1}{i+1}$$

$\sum_{i=1}^{n-1} \frac{1}{i+1} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{n} = H_{n-1}$

$$\frac{n(n-1)}{4} + n - 1 - (H_n - 1) = \frac{n(n-1)}{4} + n - H_n \quad H_n \in \Theta(\log n)$$

$\Downarrow$

$\in \Theta(n^2)$ Average case

~ Quick Sort Avarge Case ~

Asuming that the partition split can happen in each position i with the same prop $\frac{1}{n}$, we get the following recurrence relation.

$$C_{avg}(n) = \sum_{j=0}^{n-1} C_{avg}(j) + C_{avg}(n-j-1) + (n+1)$$

$C_{avg}(0) = 0$
$C_{avg}(1) = 0$

so is too complex

$C_{avg}(n) \approx \ldots 1{,}39 \cdot \log(n)$

$c \in \Theta(n \log n)$

(4) For this question;

Firstly sort the array then (I do insertion sort)
if n is even number, I calcute the avorage of the middle 2 elements, then return it
if n is odd number, return midde element.

So for sorting $o(n^2)$

$+$ middle element $O(1)$ $= o(n^2)$

So finding median of unsorted array's time Complexity is changeable belongs to sorting algorith. I prefer insertion sort algoritm for implement so For my solution time Complexity is $O(n^2)$

(5) First I create all sub arrays of given array. Then I find out the sub-arrays that satisfy the condition from all sub arrays by "brute force" method. While looking satisfy arrays, on the other hand I check if it is optimal array for condition, at the end return the optimal sub array.

Finding all sub array is (2^n) then we have an array consist of sub-arrays with 2^n elements, So the Exausted search will work 2^n times recursivly, Then $2^n + 2^n = O(2^n)$

The time complexity of my algorithm is $O(2^n)$