Ans. to the Q. No - 1

~~Ternary~~ Ternary search divides the problem (array) of size n in 3 halves.

We can write $T(n) = \frac{n}{3} + \frac{n}{3} + \frac{n}{3}$.

As we consider only one portion, therefore, we write $T(n) = \frac{n}{3}$.

Inside the part, there are constant operations.

Thus, $T(n) = \frac{n}{3} + 1$.

The portion keeps breaking down in 3 pieces by calling the same ternary search.

So, this method T is invoked on $n/3$.

$\therefore$ We can write $\Rightarrow T(n) = T(n/3) + 1$.

(Ans.) $T(\frac{n}{3}) + 1$.

Ans to the Q.No-2

Step-1: $T(n) \ldots\ldots 1$

$\downarrow$

$T(n/3) \ldots\ldots 1$

$\downarrow$

$T(n/9) \ldots\ldots 1$

$\downarrow$

$T(n/27) \ldots\ldots 1$

$\vdots$

1

It took k steps for the searching to end and work done at each step was constant. We need to find k in terms of n. Each problem size can be written as, $\frac{n}{3^{\wedge}(\text{step no})}$.

The last line can be written as, $1 = \frac{n}{3^{\wedge}k}$

If we solve it, we will find that $k = \log_3 n$.

Therefore, the time complexity is $\log_3 n \times 1$, which is eventually $O(\log_3 n)$.

Ans to Q.No-3

Time complexity will be constant or O(1) if the value is found on the first loop.

```
while(l<=r) {  -> O(1)
  mid1 = l+(r-l)/3 -> O(1)
  mid2 = r-(r-l)/3 -> O(1)
  if(A[mid1]==item || A[mid2]==item) {  -> O(1)
    return true;  -> O(1)
  }
```

$\therefore O(1+1+1+1) = O(4) = O(1)$