Best case time Complexity of Quicksort

```
Algorithm quicksort (low, high)
{ if (low < high)
    { j = Partition (low, high+1)
      quicksort (low, j-1)
      quicksort (j+1, high)
    }
}
```

Time taken by Quicksort in general can be written as

$$T(n) = T(j) + T(n-j-1) + \Theta(n).$$

Best case

Best case occurs when partition algorithm picks the middle element as pivot always

$$\therefore T(n) = T(n/2) + T(n/2) + \Theta(n)$$

$$T(n) = 2T(n/2) + c \cdot n.$$

$$= 2T\left[2T\left(\frac{n/2}{2}\right) + c \cdot n/2\right] + c \cdot n.$$

$$= 2^2 T\left(n/2^2\right) + 2 \cdot c \cdot n.$$

$$= 2^2\left[2T\left(\frac{n/2^2}{2}\right) + c \cdot \left(n/2^2\right)\right] + 2 \cdot c \cdot n.$$

$$= 2^3 T\left(n/2^3\right) + 3 \cdot c \cdot n.$$

let $n = 2^k$ $\quad k = \log n.$

$$\vdots$$

$$= 2^k T\left(n/2^k\right) + k \cdot c \cdot n.$$

$$= n \cdot T(1) + \log n \cdot c \cdot n$$

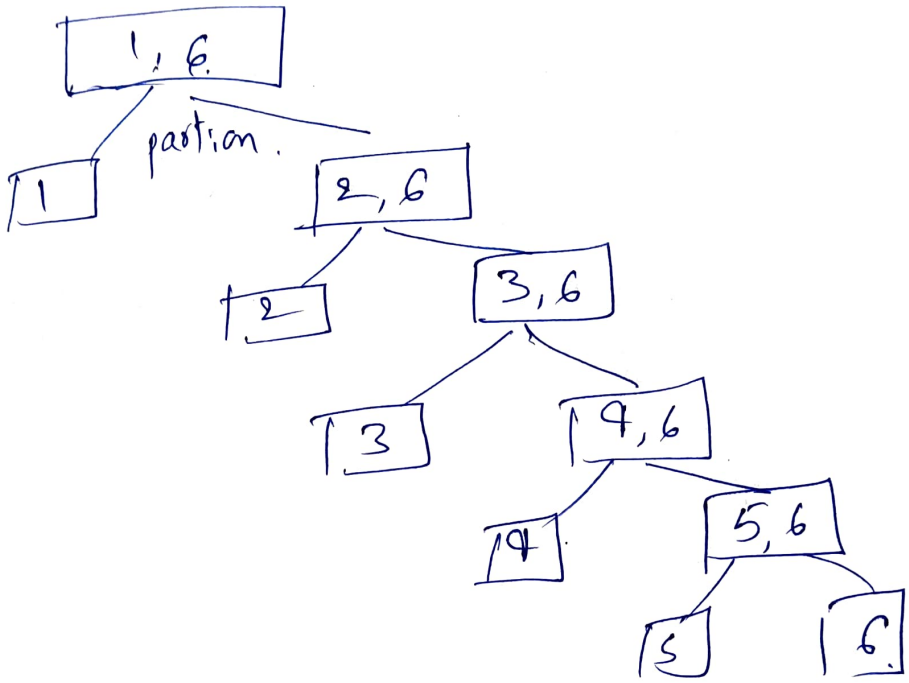
$$= O(n \log n).$$

Worst Case :

Worst case occurs when the partition always picks smallest or greatest as pivot.

eg:

1	2	3	4	5	6
2	4	8	10	12	14



$\therefore T(n) = T(0) + T(n-1) + \Theta n.$ ——— (1)

$T(n) = T(n-1) + c \cdot n.$

$= [T(n-2) + c \cdot (n-1)] + c \cdot n.$

$= T(n-2) + c[n + (n-1)]$

$= [T(n-3) + c \cdot (n-2)] + c[n + (n-1)]$

$$= T(n-3) + c \cdot [n + (n-1) + (n-2)]$$

$$\vdots$$

$$= T(n-n) + c \cdot [n + (n-1) + (n-2) \ldots (n-n)]$$

$$= 0 + c[0 + 1 + 2 + 3 \ldots n]$$

$$= 0 + c \cdot \frac{n(n+1)}{2}$$

$$= 0 + c \cdot \frac{n^2 - n}{2}$$

$$= O(n^2) \; //$$