Quick sort-

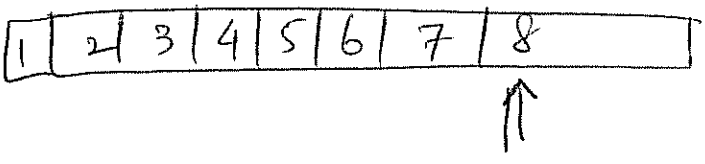
Best case: When the partition is balanced; eg, half way. The computational complexity is similar to mergesort.

$$T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n;$$
$$T(1) = c.$$

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

Worst case:

Partition always happens at the last element; there is no Right- array. In other words, The array was already sorted !!



$$T(1) = c.$$
$$T(n) = T(n-1) + c \cdot n$$
$$= T(n-2) + c(n-1) + cn$$
$$= T(n-2) + 2cn - c$$
$$= T(n-3) + 3cn - 3c$$
$$= T(n-k) + k \cdot cn - k\frac{(k-1)}{2} \cdot c \longrightarrow = T(n-4) + 4cn - 6c$$

at $k = n-1$;

$$T(1) + (n-1) \cdot c \cdot n - \frac{n(n-1)}{2} c$$
$$= O(n^2)$$

Average - Quick sort   Partition can happen at any index; equal probability

$$T(n) = n+1 + \frac{1}{n}\sum_{i=1}^{n-1}\left(T(i-1) + T(n-i)\right)$$

$$= (n+1) + \frac{1}{n}\left(2T(1) + 2T(2) + \cdots + 2T(n-1)\right)$$

$\Rightarrow$ ~~(n+1) T n~~ =

$$n\,T(n) = n(n+1) + 2\left(T(1) + T(2) + \cdots + T(n-1)\right)$$

$$\Rightarrow (n+1)\,T(n+1) = (n+1)(n+2) + 2\left(T(1) + T(2) + \cdots + T(n)\right)$$

Subtract -

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

$$= 2T(n) + (n+1)\cdot 2$$

$$(n+1)\,T(n+1) = (2+n)\,T(n) + (n+1)\cdot 2$$

or, $$\frac{T(n+1)}{(n+2)} = \frac{T(n)}{n+1} + \frac{2}{n+2}$$

Simplify, replace n+1 by n , $$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2}{n+1}$$

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

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

$\vdots$

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

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + \frac{2}{3} + \frac{2}{4} + \cdots$$

$$= \frac{T(1)}{2} + 2\sum_{k=3}^{n+1} \frac{1}{k}$$

$$= 2\sum_{k=3}^{n+1} \frac{1}{k}$$

$$= 2\log_e(n+1) - \log_e 2$$

$$\frac{T(n)}{n+1} \leq 2\log_e(n+1)$$

$$T(n) = 2(n+1)\log(n+1)$$

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

$$\int_1^{n+1} \frac{1}{x}\,dx = [\log x]_1^{n+1}$$

$$= \log(n+1) - \log 1$$

Time complexity — Merge sort

$$T(n) = \begin{cases} \text{if } c & ; \text{if } n=1 \\ 2T\left(\frac{n}{2}\right) + cn & \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$= 2\left(2T\left(\frac{n}{4}\right) + c\cdot\frac{n}{2}\right) + cn$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + 2\cdot cn$$

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + k\cdot cn .$$

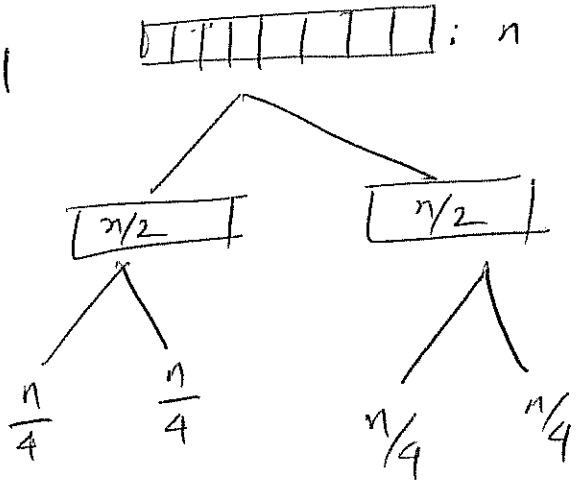
So for $k = \lg n$

$$= 2^{\log n} \underbrace{T(1)}_{c} + \lg n \cdot c \cdot n$$

$$= n\cdot c + c\cdot n\cdot \lg n$$

$$\leq O(n \lg n) .$$

Memory complexity



So for $k = \lg n$ we are at ~~root~~ leaf, (1-one item)

So there will be $\lg n$ levels and at each level we need n sized buffer,

So memory cost is $(n \cdot \lg n)$ ($\lg n$ levels)