Quicksort (I)

\# (10), 80, 90, 30, 20 → 10 is sorted position

\# 6, 3, 1, 2, 5, (9) → 9 is sorted pos.

\# 4, 6, 7, (10) 16, 12, 14 → 10 is sorted pos.

because → all the elements before the element are smaller and All the elements are greater than the circled element.

So, quick sort works on the idea that, an element is in the sorted position if all the elements left hand side should be smaller than that element and all the elements after that element (righ hand side) should be greater than the element.

A →

0	1	2	3	4	5	6	7	8	9
(10)	16	8	12	15	6	3	9	5	∞/large value.

i → which are greater than 10 or pivot

j ← wich are smaller than 10 or pivot

Pivot = 10,

i → will increase   j = decrease.

(10)	16	8	12	15	6	3	9	5	∞
	i							j	

condition 16 > 10 & 5 < 10

so, swap (16, 5)

(10)	5	8	12	15	6	3	9	16	∞

i = 8

8 > 10 → False.

so, i will increment further,

i = 12 and condition true.

J = 9, 9 < 10 condition true

so swap, (12, 9)

(10)	5	8	9	15	6	3	12	16	∞

→ i = 15   J = 3

swap (15, 3)

swap, (15, 3)

(10)	5	8	9	3	6	15	12	16	∞
					j	i			

~~[illegible]~~e.

in this case i > J → So we ~~[illegible]~~ need not go further or exe
↳ i greater J.

Now. Swap J with pivot = 10,

6	5	8	9	3	10	15	12	16	∞

Not sorted (6 5 8 9 3) — sorted or partiontion position (10) — Not sorted (15 12 16 ∞)

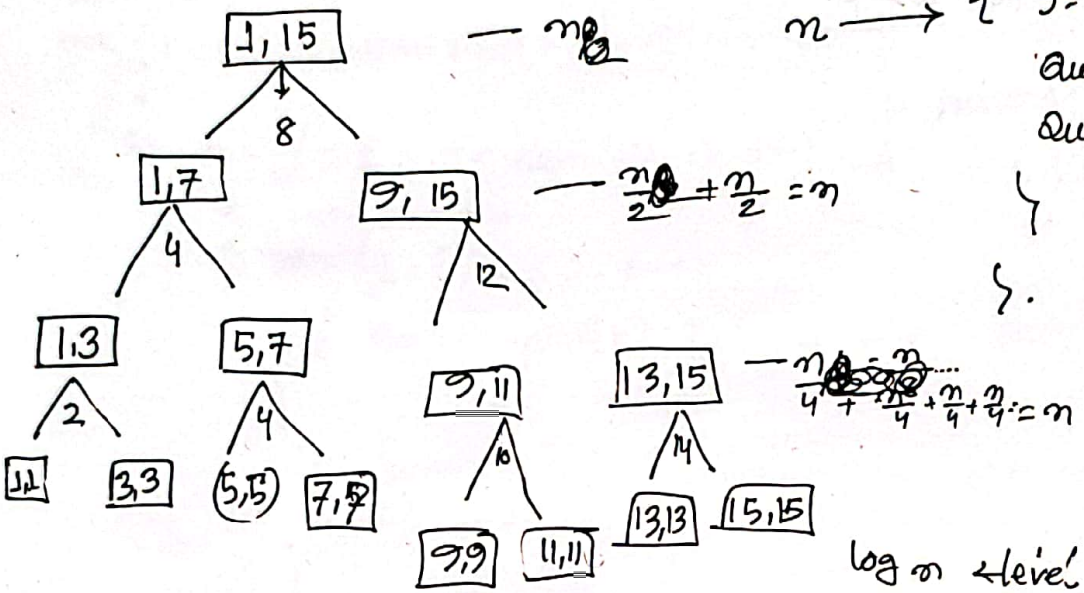
```
partion{ l, h)
{ pivot = A[l]
  i=l, j=h,
  while (i<J)   ← (i less than J)
  { do{ i++,
     } while (A[i] ≤ pivot)      যদি কোন বড় element পায়
    do, { J--, }
     while (A[J] > pivot)        যদি কোন ছোট element পায়.
    if (i<J)
       swap (A[i], A[J]);
  }
  swap ( A[l], A[J] }
  return j,
}
```

```
Quick sort
{ if ( l < h)
    J = partion (l, h)
    Quicksort( l, J (J-1) )      3 বার আসবে
                 ↳ J second iterations.
    Quicksort (J+1, h)   . এ ও একই কাজ করে.
}
```

Best case :-

we have 1-15 elements.

i.e every time partition n/2



$n \rightarrow$ — n

$\frac{n}{2} + \frac{n}{2} = n$

$\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4} = n$

log n level

```
Quick sort (l, h)
{ if (l < h)
  { j = partition (l, h);
    Quicksort ( l, j);
    Quick sort (j+1, h);
  }
}.
```

~~Best case~~ → level = $\log n$.

$\frac{n}{2}, \frac{\frac{n}{2}}{2}, \frac{\frac{\frac{n}{2}}{2}}{2}, \ldots = 1$

$\therefore n = 2^k$

$k = \log_2 n$

Best case: $(n \log n)$

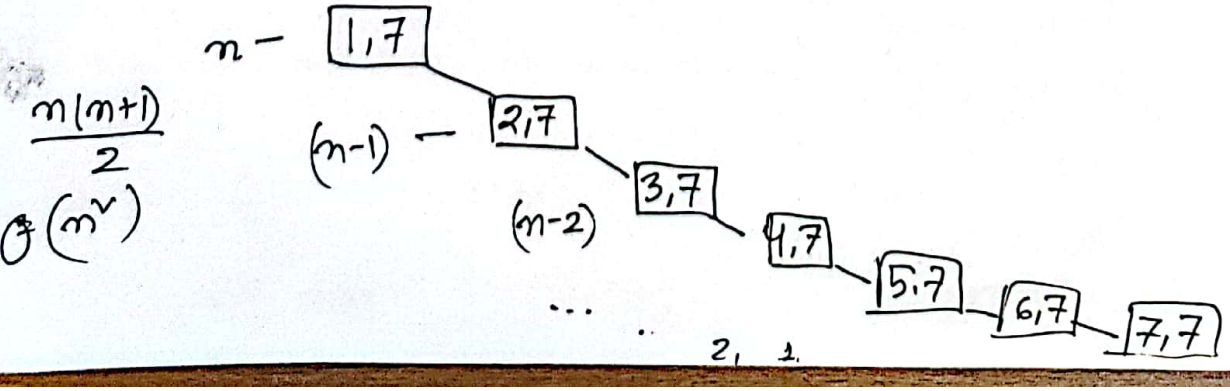
it portion will occour middle position

worst case :-

(2) 4, 8, 10, 16, 18, 17

i → ← j

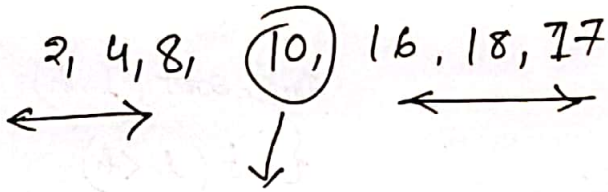
Always, partioning will happen in the begging of a list.



$\frac{n(n+1)}{2}$

$O(n^2)$

→ if



if we select 10 as pivot that means median value of the list