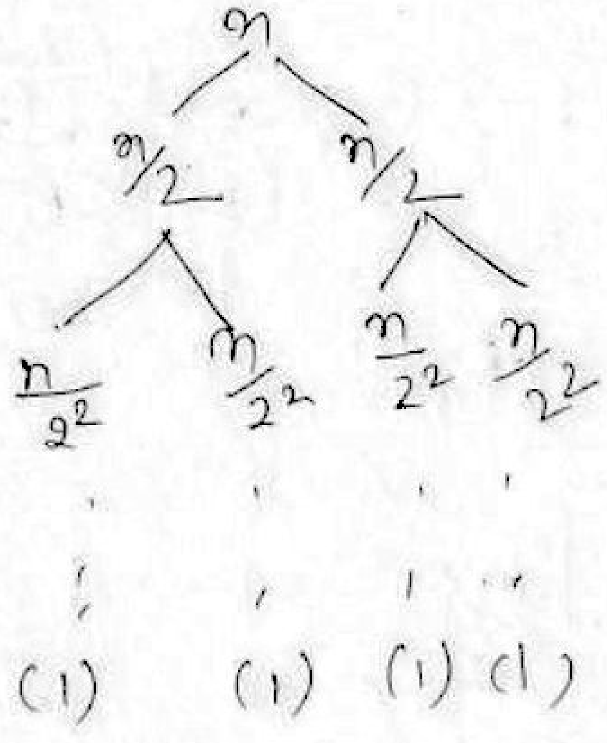
Quick sort [Tony Hoare]

```
Partition (A, p, q)
{   x = A[p]
    i = p;
    for (j = p+1, j ≤ q ; j++)
     {  if (A[j] ≤ x)
         {  i = i+1
            swap (A[i], A[j])
         }
     }
     swap( A[i], A[p])
 }
```

```
Quicksort (A, p, q)
{   if (p<q)
      {
O(n)     m = partition (A, p, q)
T(n1)    Quicksort (A, p, m-1)
T(n2)    Quicksort (A, m+1, q)
      }
}
```

$$T(n) = T(n_1) + T(n_2)$$



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

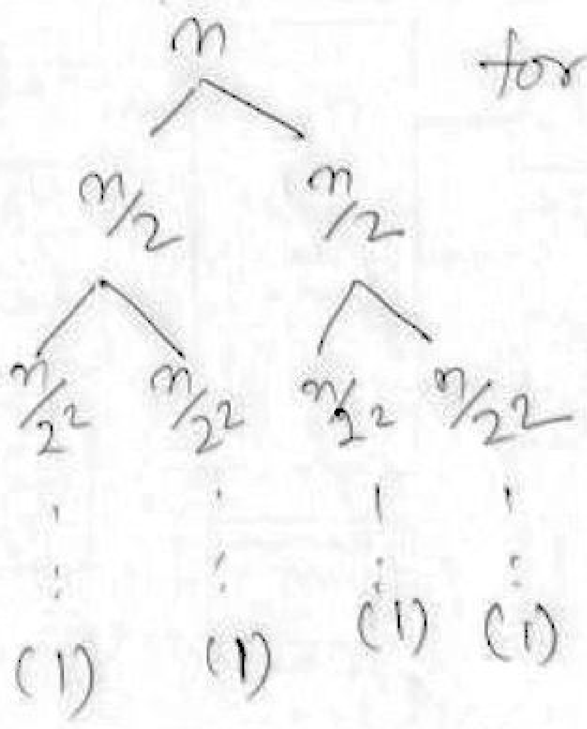
$$= 2T(n/2) + c \cdot n$$

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

QUICKSORT (Tony Hoare)

```
Quicksort (A, P, q)
{ if (p < q)
    $O(n)$  { m = partition (A, P, q)
    $T(n_1)$  Quicksort (A, P, m-1)
    $T(n_1)$  Quicksort (A, m+1, q)
    }
}
```

$$T(n) = T(n_1) + T(n_2) + O(n)$$

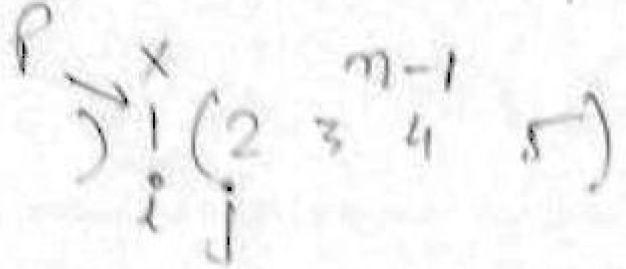


for best case

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

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

for worst : When Elements are already sorted



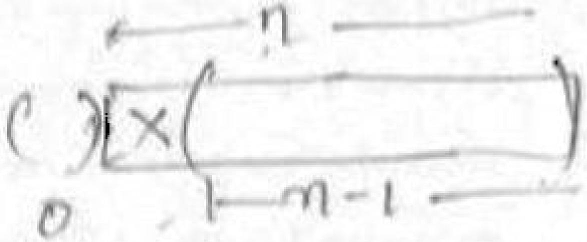
$x = 1$

$2 \leq x$

$3 \leq 1 x$

$4 \leq 1 x$

$5 \leq 1 x$

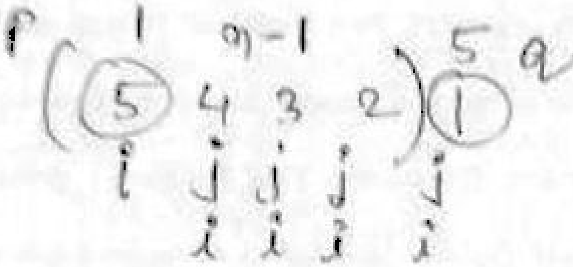


Recurrence Relation

$T(n) = T(0) + T(n-1) + c \cdot n$

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

$= O(n^2)$



$4 \leq 5$ ✓

$3 \leq 5$ ✓

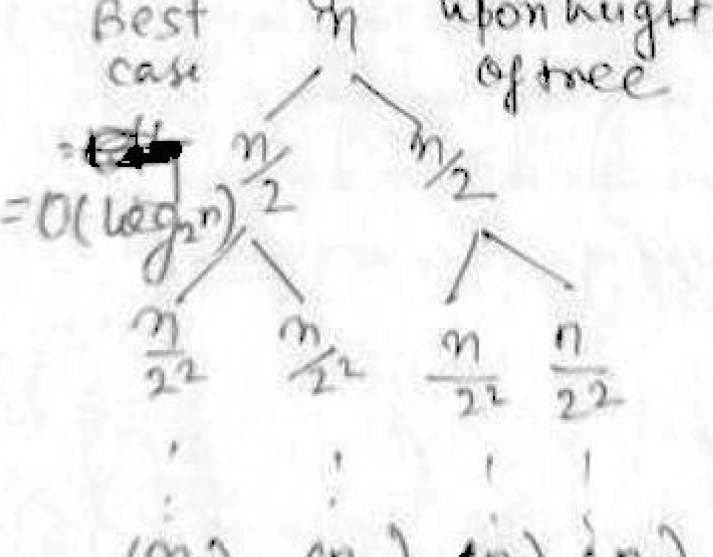
$2 \leq 5$ ✓

$1 \leq 5$ ✓

Space complexity : depend upon height of tree

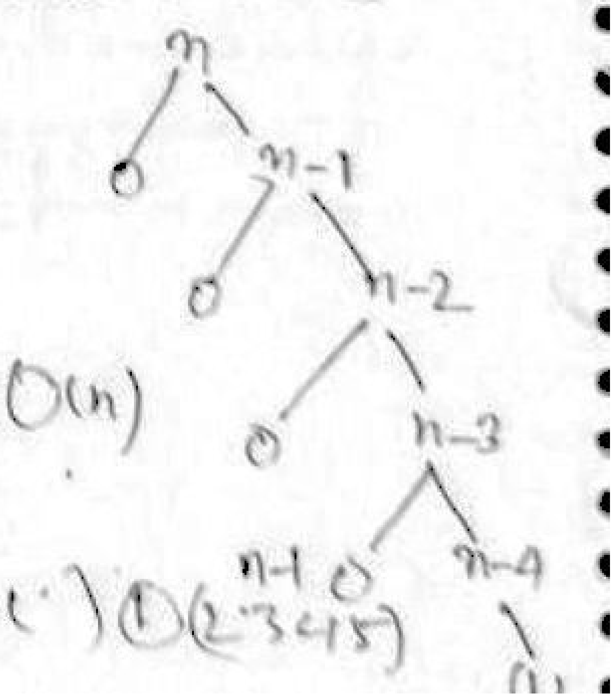
Best case

$= O(\log_2 n)$



worst case

$O(n)$



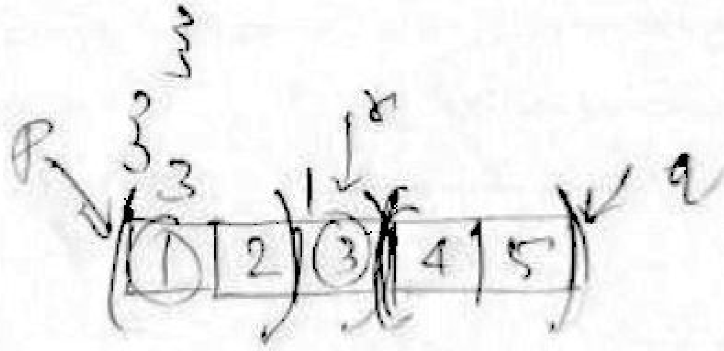
Randomized Quicksort

RGF
Random
Ganrator
function

$$(\ )\ \underset{x}{①}\ \overset{n-1}{(2\ 3\ 4\ 5)}$$

$T(n) = T(n-1) + cn \Rightarrow O(n^2)$

```
RQS ( A, P, q )
{ if (p<q)
    { r = RGF( P, q )
      swap ( A[P], A[r] )
    m = partition (A, P, q)
    RQS( A, P, m-1)
    RQS(A, m+1, q)
```



$x = 3$

$2 \leq 3$

$1 \leq 3$

$4 \leq 3$

$5 \leq 3$

$O(n \log n)$