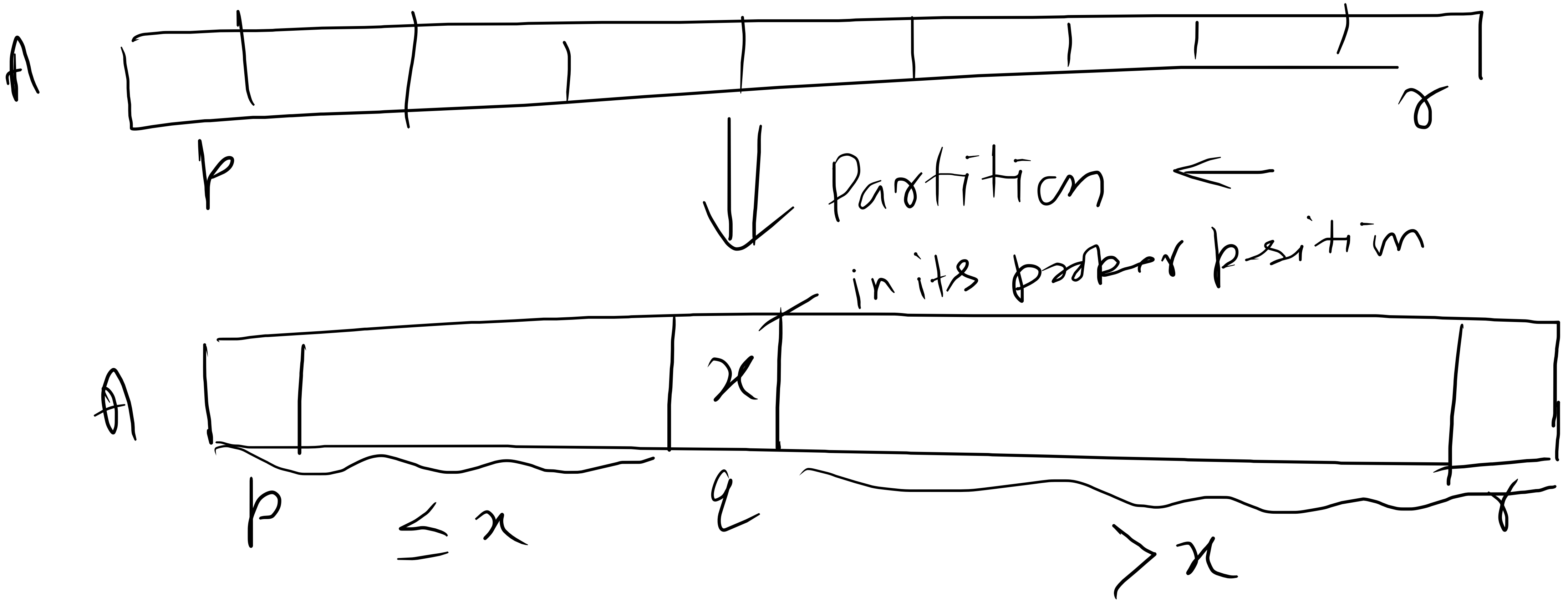
Quicksort

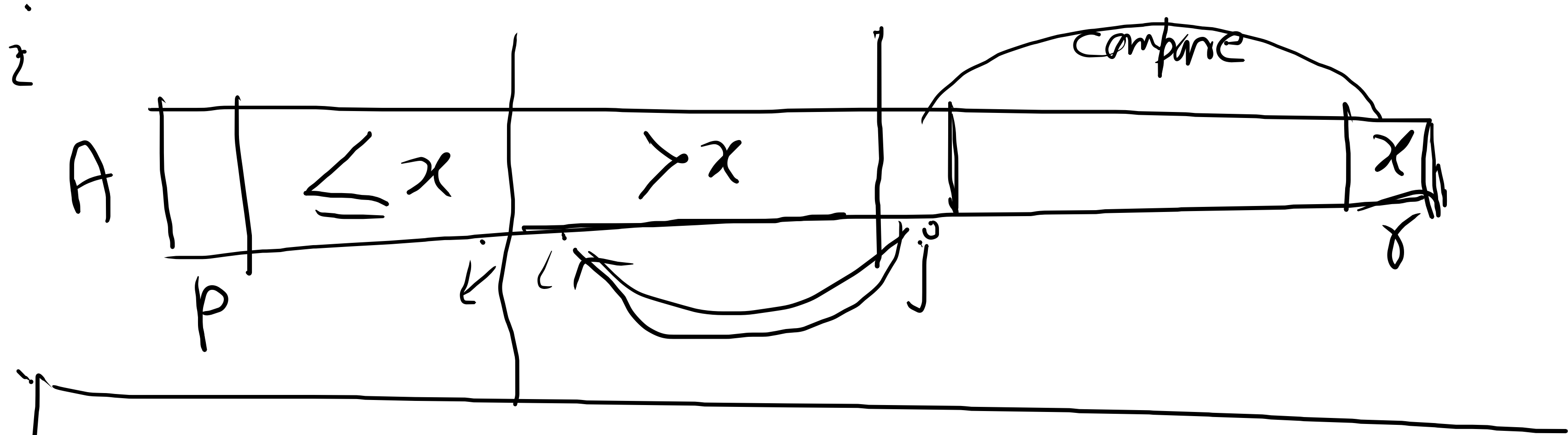
Helper function Partition.



```
Partition(A, p, r)
  x = A[r]
  i = p-1
  for j = p to r-1
        if A[j] <= x
              i = i+1
              swap A[i] and A[j]
  Swap A[i+1] and A[r]
  return i+1
```

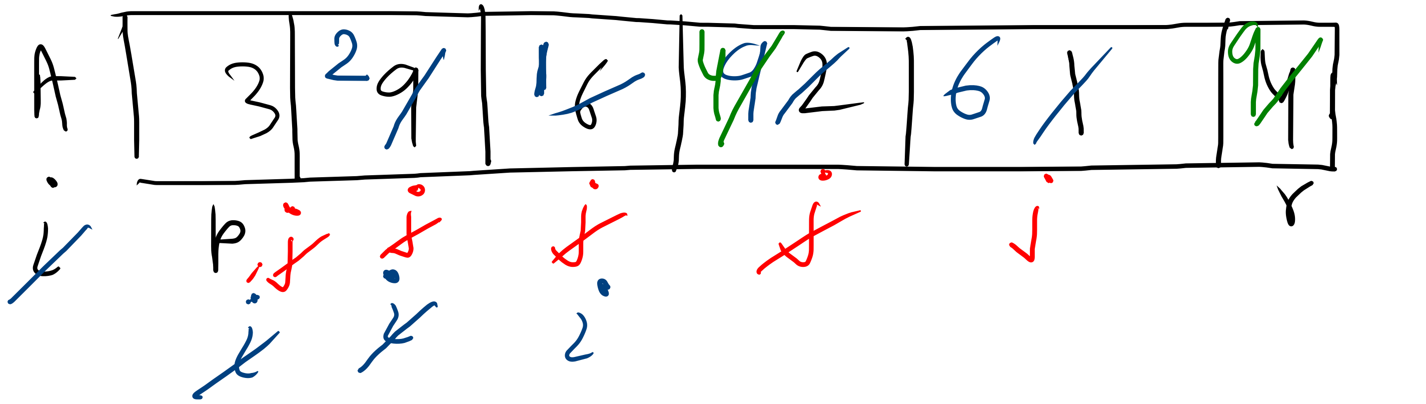
Running time

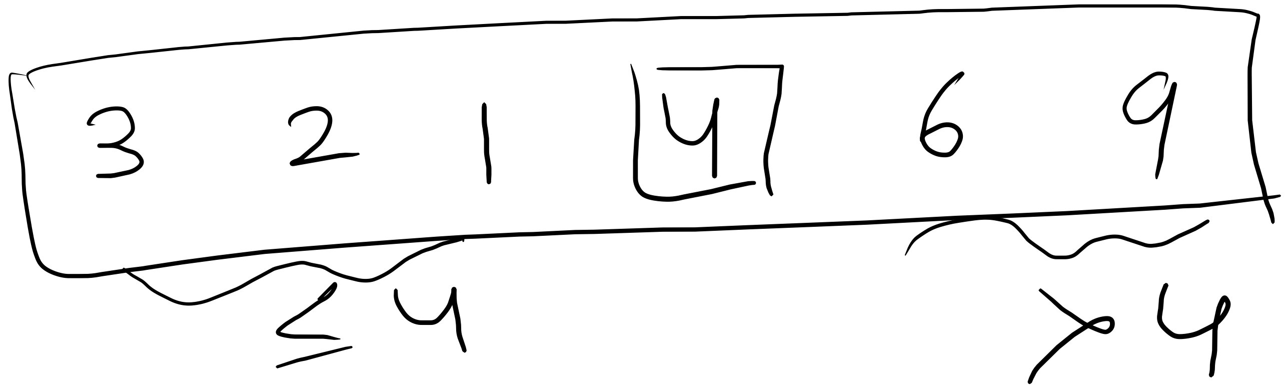
$O(n)$



3 9 6 2 1 4

$x = 4$





Quicksort (A, p, r) — T(n)

if p < r

  q = Partition (A, p, r) ←

  Quicksort (A, p, q−1)

  Quicksort (A, q+1, r)

Running time

x ← is called the pivot element.

efficient when the pivot element partitions the array into roughly two equal arrays.

$$\frac{1}{100} : \frac{99}{100}$$

$$T(n) = T\left(\frac{n}{100}\right) + T\left(\frac{99n}{100}\right) + O(n)$$

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

H.W

How?

non-efficient

constant : n - constant .

Recurrence

$$T(n) = T(\text{constant}) + T(n - \text{constant}) + O(n)$$

$$= O(n^2)$$

H.W How ?

Exm worst case happens

- All elements are same (roughly same)
- sorted sequence .

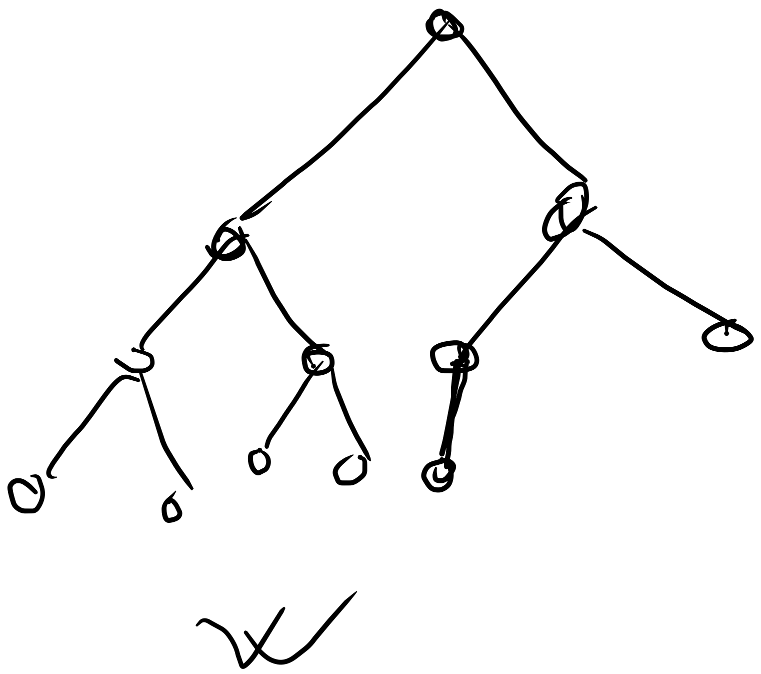
Randomized version

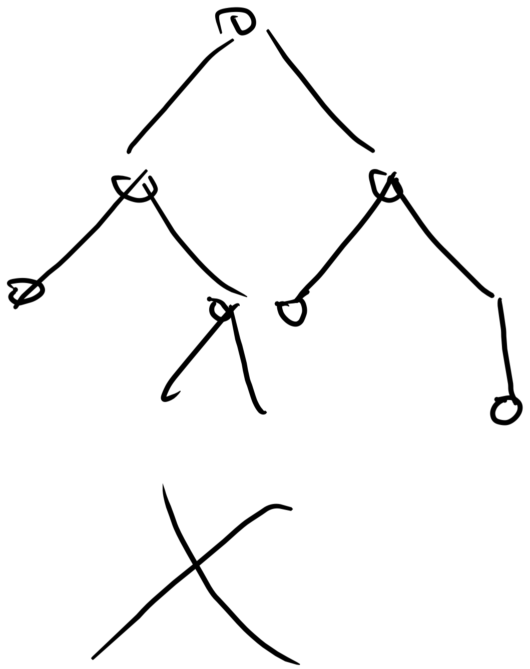
$O(n \log n)$ – expected running time.

Heap Sort

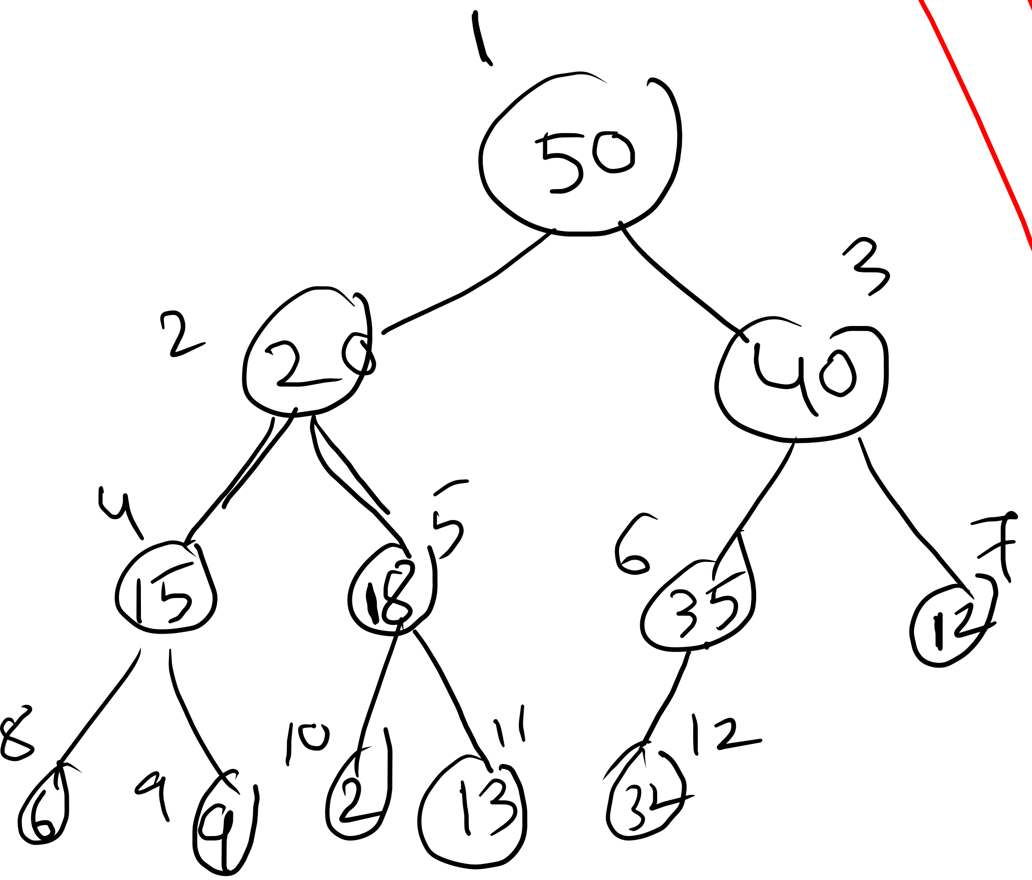
Heap data structure

- It is an array
- can be viewed as a nearly complete binary tree where the lowest level leaves are filled from left to right.





Max-heap property: The key of a node is $>$ the key of its children.



50	20	40	15	18	35	12	6	9	2	13	32
1	2	3	4	5	6	7	8	9	10	11	12

Min-heap property: The key of a node is $<$ the key of its children.