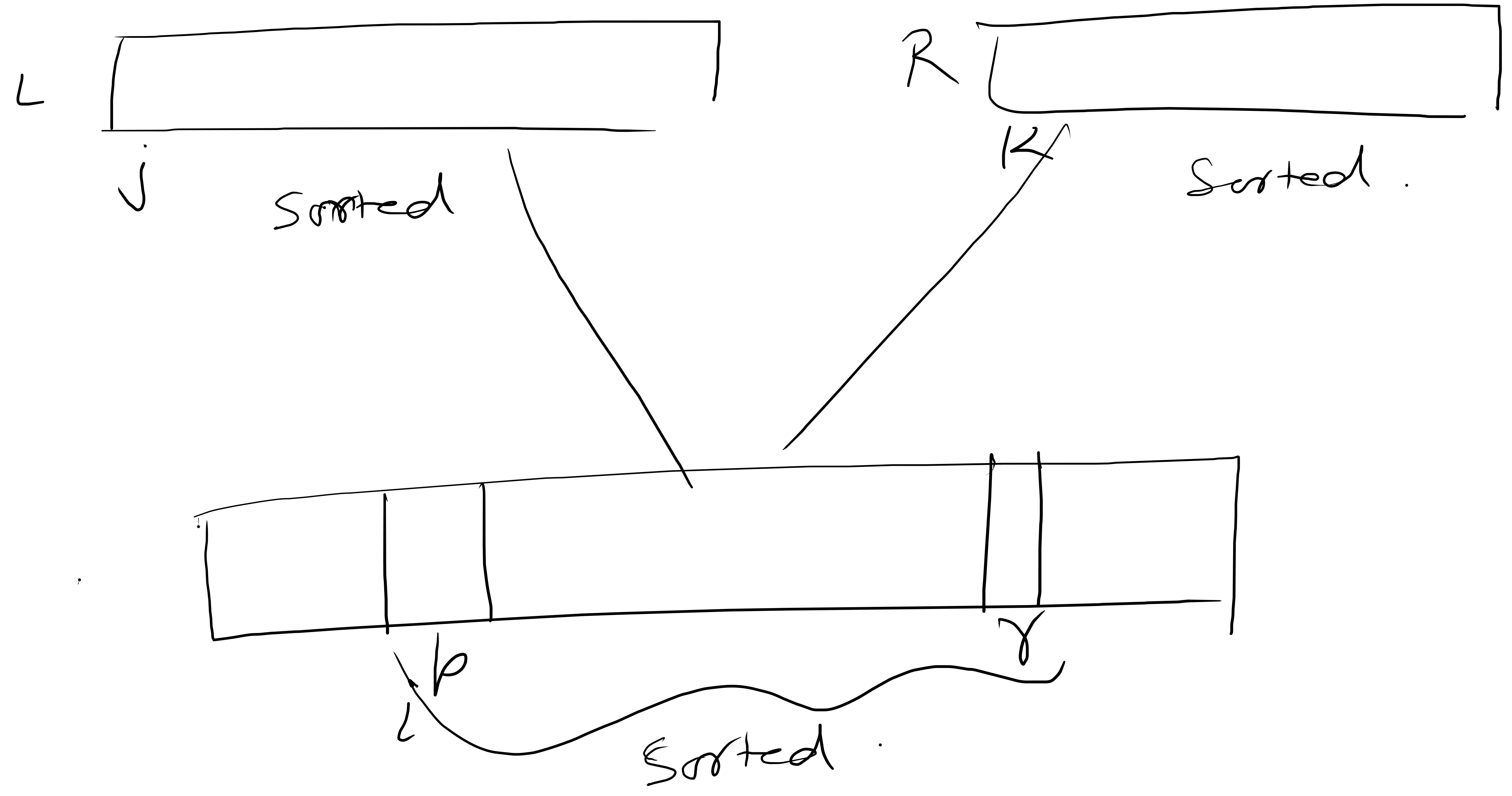
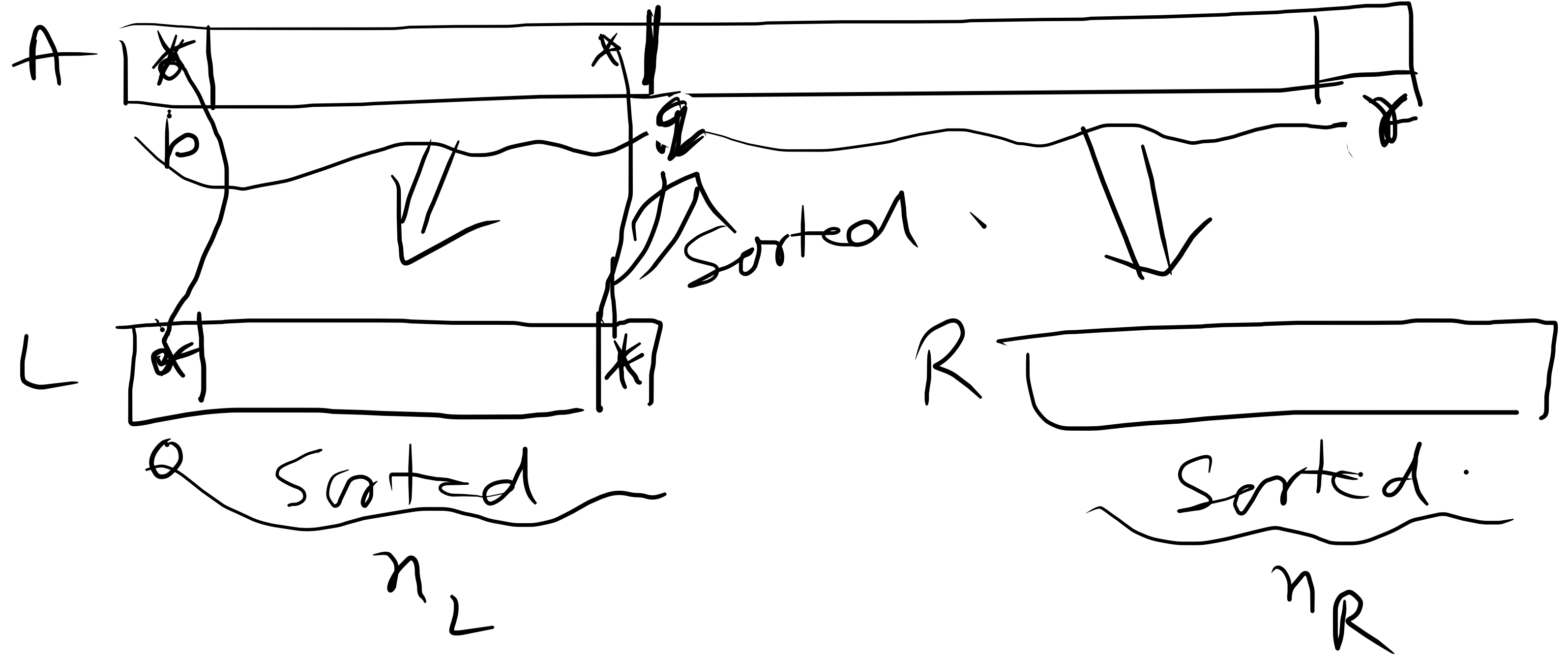
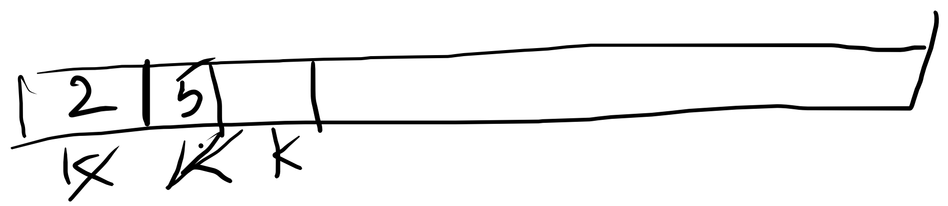
Merge Sort

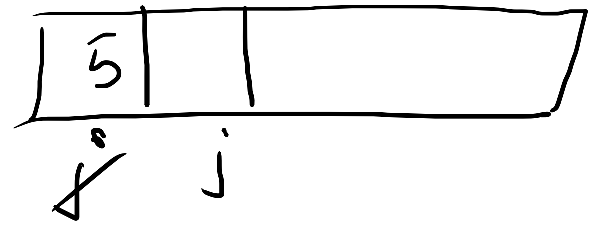
merge



```
merge (A, p, q, r)
  $n_L = q - p + 1$                          —— $\theta(1)$
  $n_R = r - q$                              —— $\theta(1)$
  create two arrays $L[0 .. n_L]$
  and $R[0 .. n_R]$                          —— $O(n)$
  for $i = 0$ to $n_L - 1$                   —— n time
      $L[i] \leftarrow A[p+i]$               —— $\theta(1)$
  for $i = 0$ to $n_R - 1$                   —— n time
      $R[i] \leftarrow A[q+i]$               —— $\theta(1)$
  $i = 0, j = 0, k = p$                      —— $\theta(1)$
  while $i < n_L$ and $j < n_R$              —— $O(n)$
      if $L[i] \le R[j]$
          $A[k] \leftarrow L[i]$
          $i = i + 1$                        } $\theta(1)$
          $k = k + 1$
      else
          $A[k] = R[j]$
          $j = j + 1$                        } $\theta(1)$
          $k = k + 1$
  while $i < n_L$                            —— ~~n~~ time
      $A[k] = L[i]$
      $i = i + 1$                            } $\theta(1)$
      $k = k + 1$
  while $j < n_R$                            —— n time
      $A[k] = R[j]$
      $j = j + 1$                            } $\theta(1)$
      $k = k + 1$
```

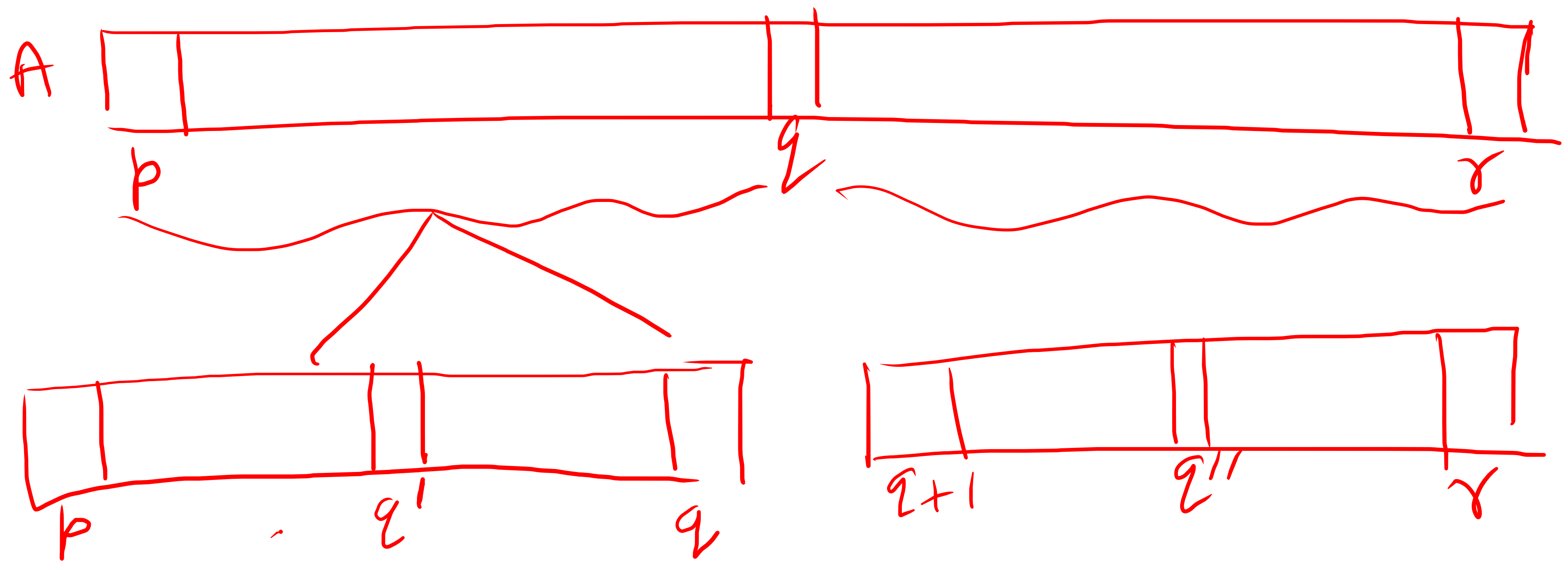


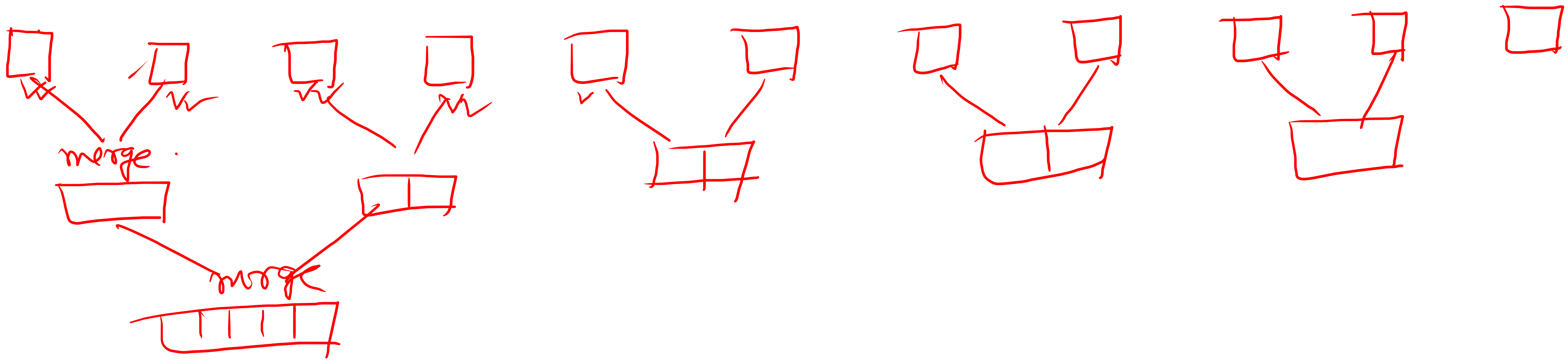




Running time:

$O(n)$


A
p
q
r
p
q'
q
q+1
q''
r


merge
merge

```
mergesort (A, p, r) —— $T(n)$
  if  $p \geq r$ —— $\theta(1)$
        return. —— $\theta(1)$
  $q = \lfloor \frac{p+r}{2} \rfloor$ —— $\theta(1)$
  w mergesort (A, p, q) —— $T(n/2)$
  w mergesort (A, q+1, r) —— $T(n/2)$
    merge (A, p, q, r) —— $\theta(n)$
```

<u>Running time</u>

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

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

H.W

Ex^m Run mergesort algorithm on this example.

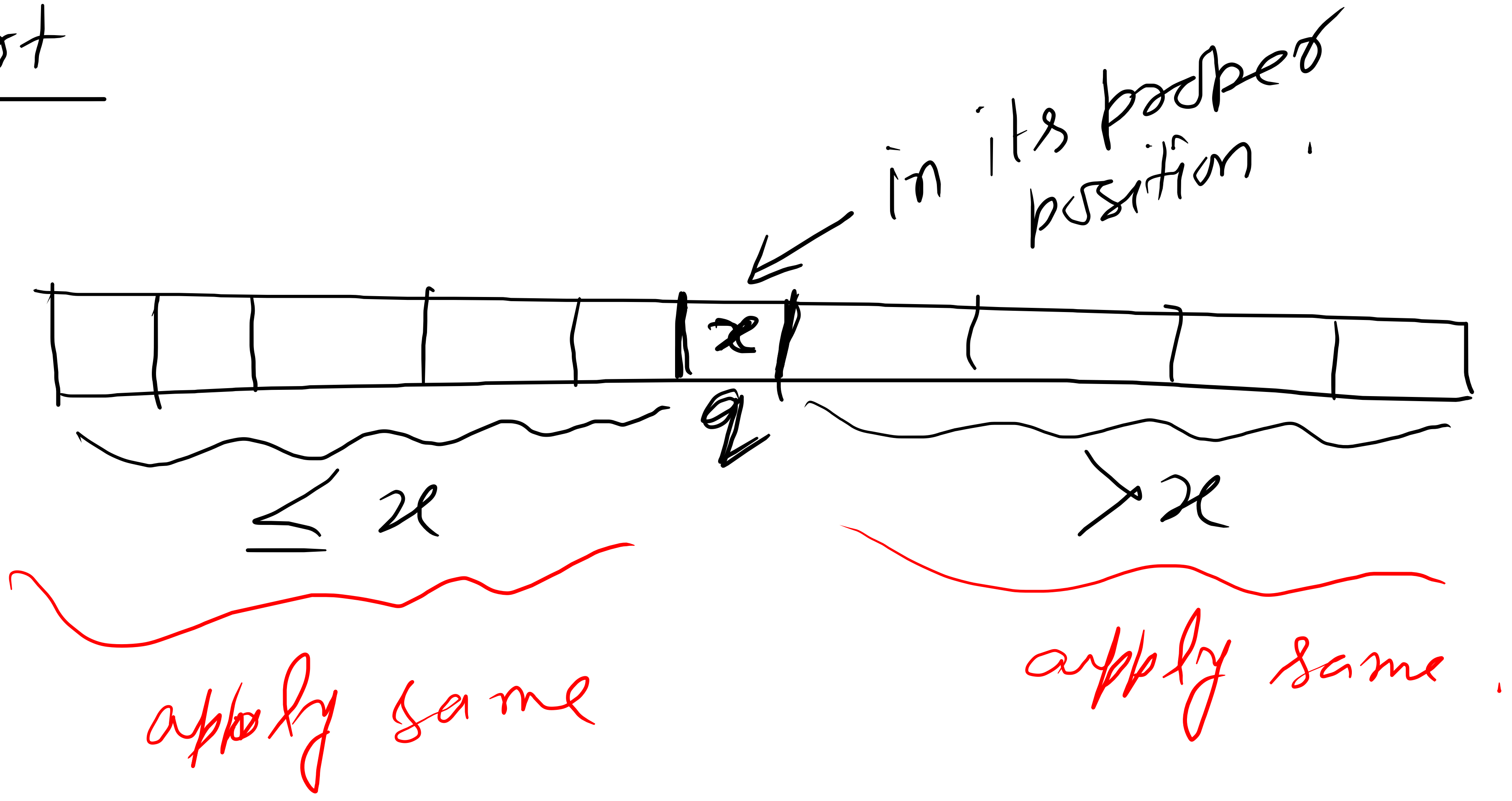
8 2 9 6 7 4 3 1

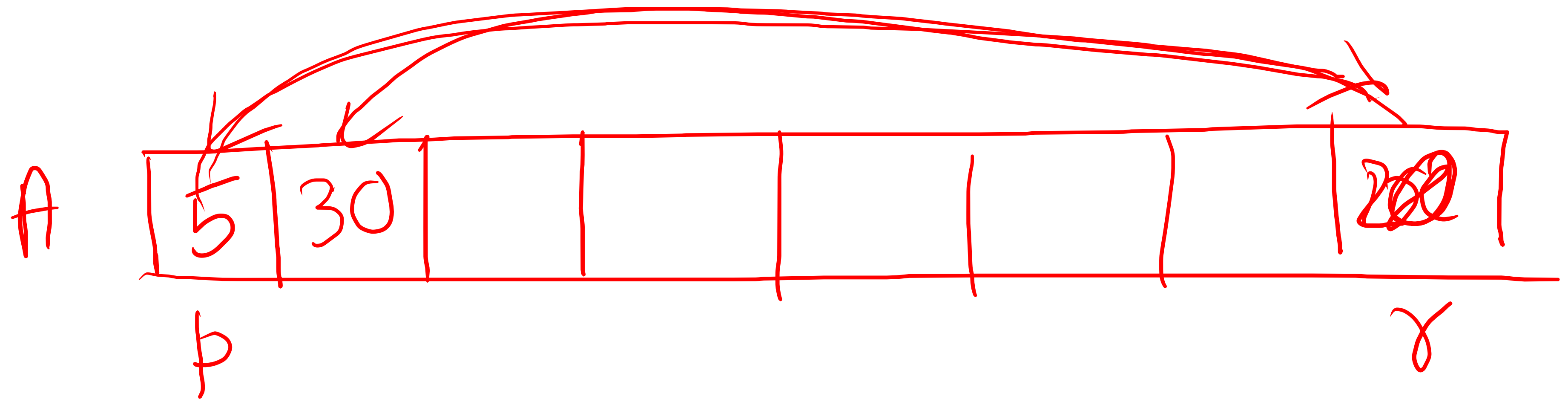
merge sort is not an inplace sorting.

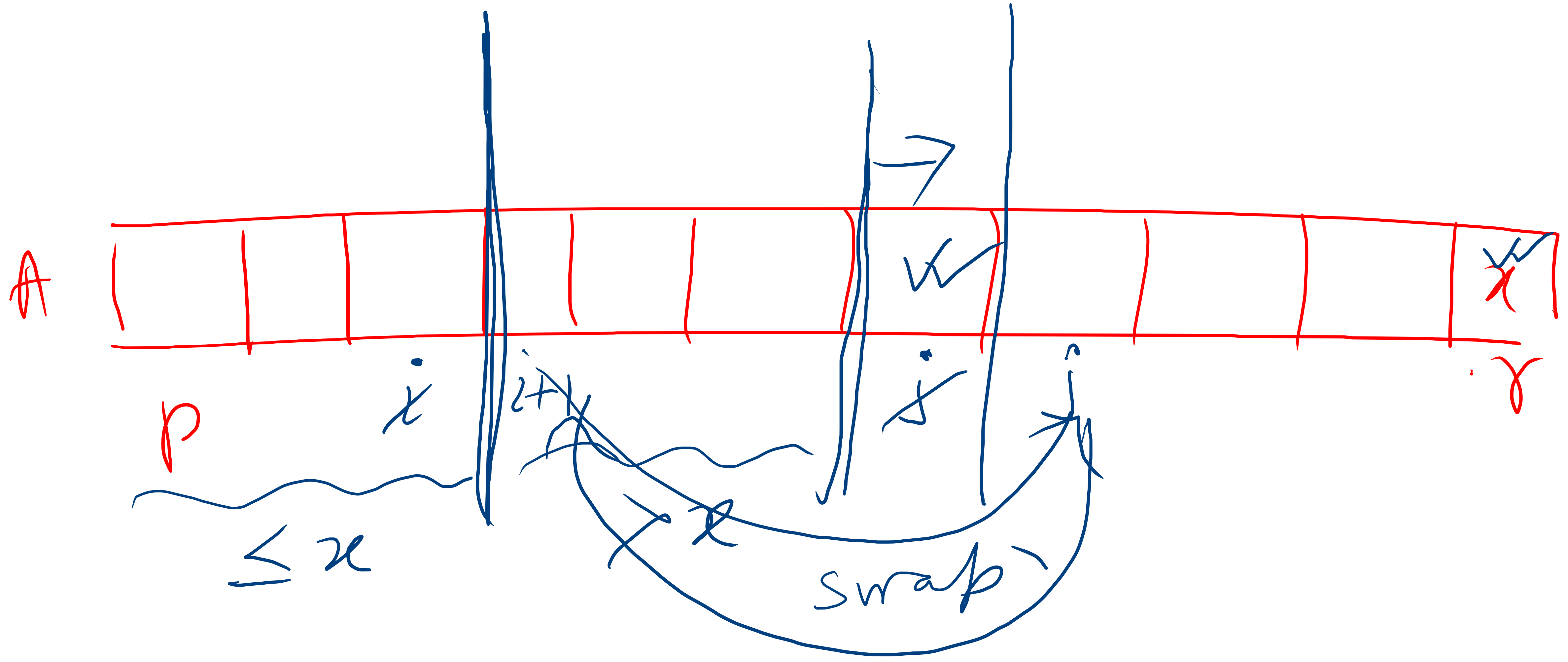
Additional $O(n)$ size array is required.

Quicksort

Partition




A
5
30
200
p
r


A
x
p
i
i+1
j
j
r
≤ x
> x
> x
swap