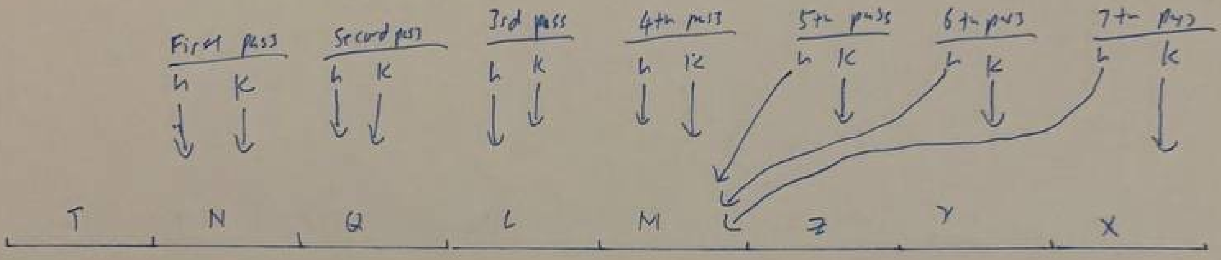
Goh Zu Hong U2021167E EE35

2×1×1×6×7= 84

(e) T is chosen as pivot; k is for loop of array ; h is the partition



Once for loop ends, T will swap with M (pointed by h)

M N Q L T Z Y X // After partition

(f) After first partition: (from e)

M N Q L T Z Y X

choose M and Z as ~~subarr~~ pivots for sub arrays

L M Q N X Y Z

choose Q and X as pivot

L N Q X Y

combining all sub arrays

L M N Q T X Y Z

```
(g) sort-by-odd-even (arr) {
      # Assume index-0 base
      h = 0 ;
      for (k = 1 to arr.len()-1) {
          if (arr[k] % 2 = 1) {
              h = h+1 ;
              swap(arr, h, k)
          }
      swap (arr, 0, h)
      }
```

Similar to partition algorithm.
ignore first element. At end of function,
swap it with the last odd element.

```
(h) smallest_in_max_heap (~~heap~~ arr) {
      # only ~~sort~~ find smallest value in leaf nodes
      start = ⌊arr.len/2⌋ + 1 ;
      end = arr.len()
      partition (arr, start, end)
      }

  partition (arr, start, end) {
    if (start = end) {
      return arr [start] }
    val = arr[start] ; h = start
    for ( i = start to end) {
        if ( arr[i] < val) {
            h = h+1
            swap (~~A~~ arr, h, ~~k~~i)
            }
        }
    swap (arr, start, h)
    partition (arr, start, h-1)
  }
```

!! Assume index-1 base.