Recursive Sorting Algorithms

- A4 due tonight
- Submit to "A4 SmokeTest"
- Slip days

Last time: quadratic-time sorting
- insertion sort
- selection sort

+ lower bound proof: sorting takes atleast:

A: $O(n)$
B: $\Omega(n)$
C: $O(n \lg n)$
D: $\Omega(n \lg n)$
E: $\Omega(n^2)$

O: upper bound
Ω: lower bound

Are there actual $O(n \lg n)$ algorithms?

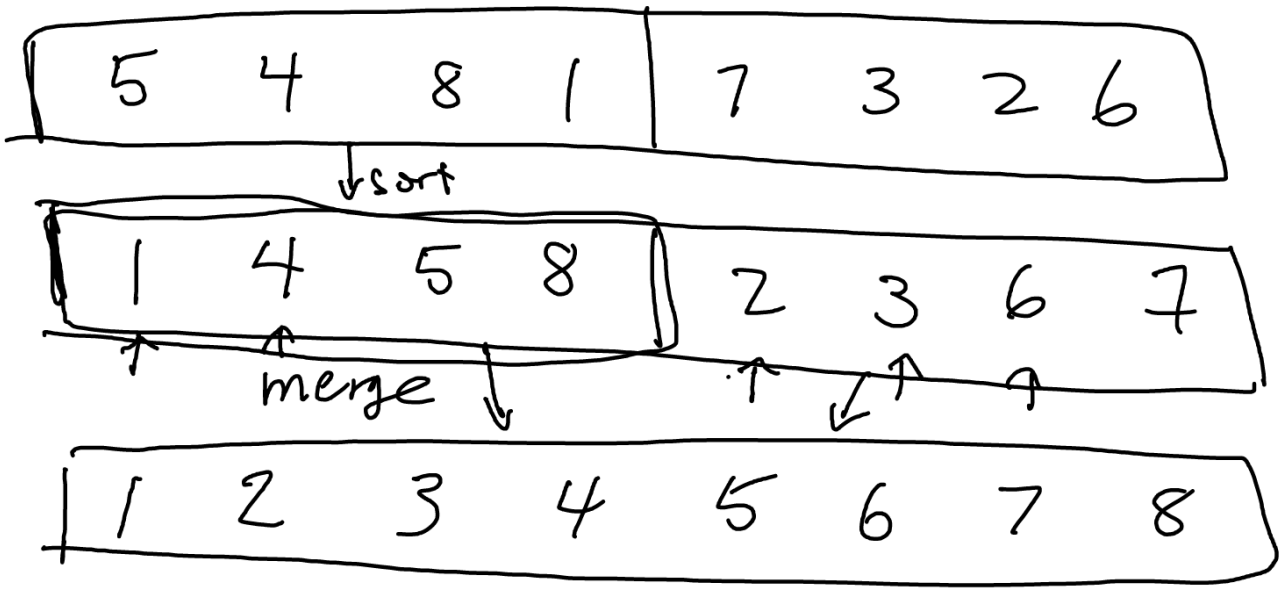
Yes!

Mergesort

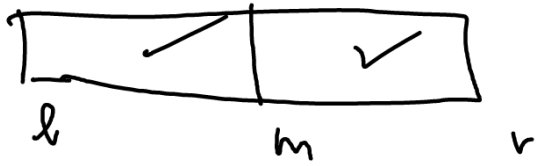
recursive : "divide and conquer"

To sort 2 arrays:

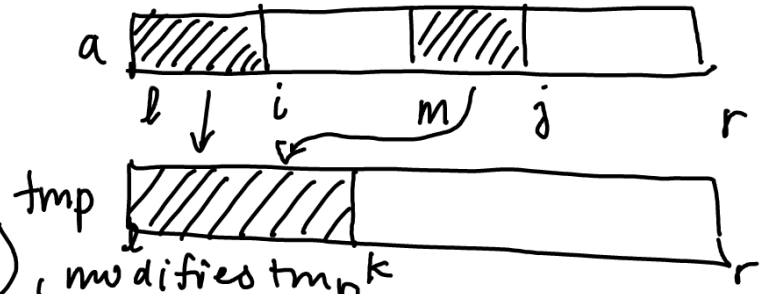
1. Divide it into 2 halves
2. Sort subarrays recursively
3. Merge results



```
/** Effect: sort a[l..r), modifies tmp.
 * Requires: l<r. tmp.length ≥ a.length */
void sort(int[] a, int l, int r, int[] tmp) {
    if (r-l == 1) return;
    int m = (l+r)/2;   // note: l<m<r
    sort(a, l, m, tmp);
    sort(a, m, r, tmp);
    merge(a, l, m, r, tmp);
}
```



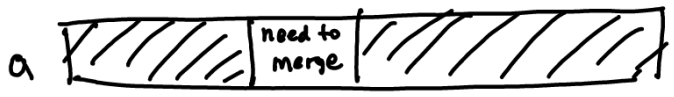
Real work: merge.



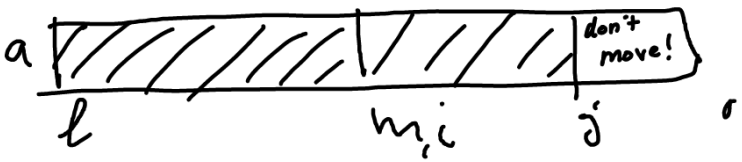
** ~~Effect~~: sort a[l..r), modifies tmp
Requires: a[l..m), a[m..r) sorted, l<m<r

```
void merge(int[] a, int l, int m, int r, int[] tmp) {
    int i=l, j=m, k=l;
    while (i<m && j<r) {
        if (a[i] < a[j]) {
            tmp[k] = a[i]; i++; k++;
        }else {
            tmp[k]= a[j]; j++; k++;
        }
    }
    System.arraycopy(a, i, tmp, k, m-i);
    System.arraycopy(tmp, l, a, l, j-l);
}
```

loop post cond.

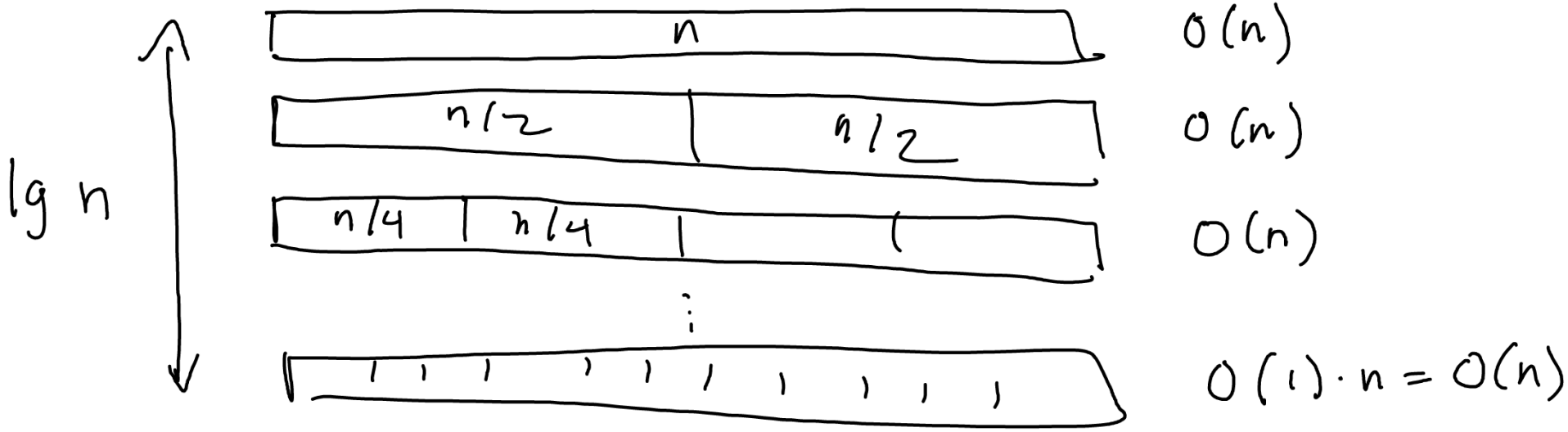


OR



Performance

work = merging = $O(r-\ell)$



Total work: $O(n \lg n)$

\+ stable sort
\+ reliable worst case
− uses double space

Quicksort

mergesort : split → recurse → merge
O(n)

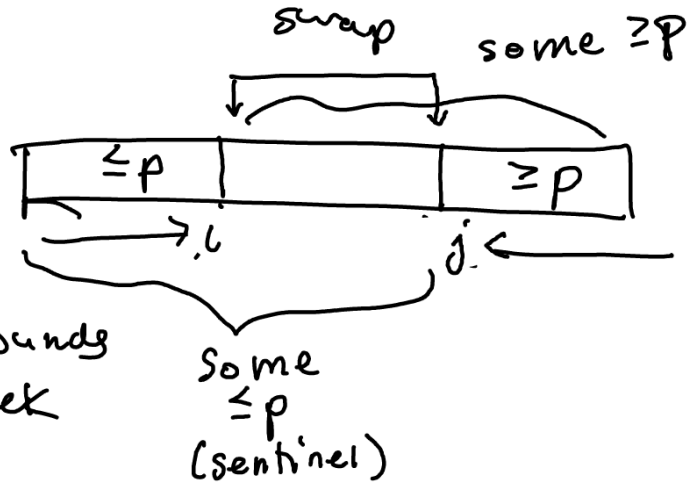
Quicksort : partition → split → recurse
O(n)

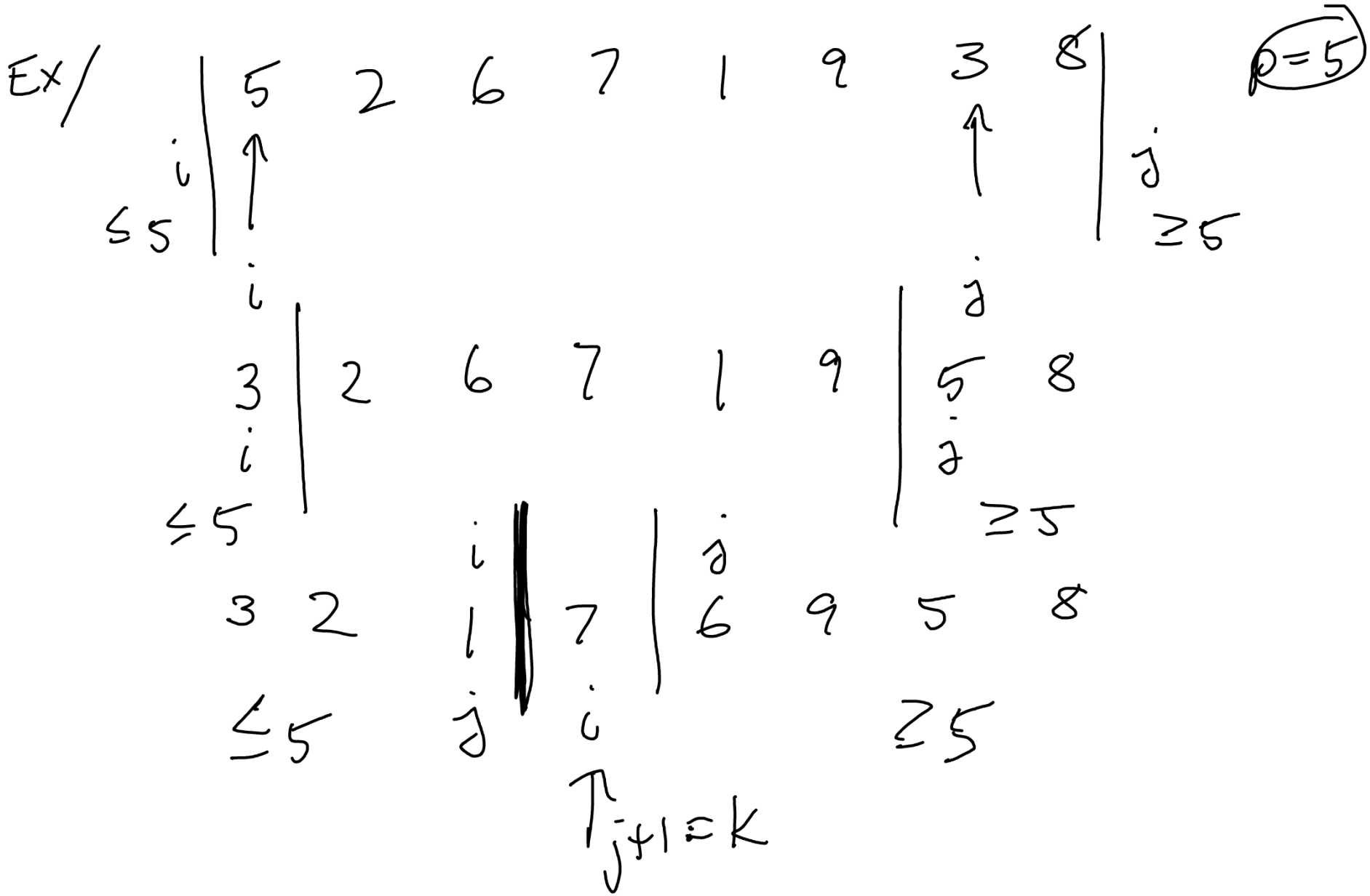
Partition does work. Goal:

for some pivot value p

```
/** Effect: sort a[l..r). */
void quicksort(int[] a, int l, int r) {
    if (r-l ==1) return;
    int k = partition(a,l,r);
    quicksort(a, l, k);
}   quicksort(a, k, r);
```

```
/** Returns k where l<=k<r. Effect: permute a so a[l..k) <= p
                                                 a[k..r) >= p */
int partition(int[] a, int l, int r)
    int p = a[l]; // better: random
    int i=l-1, j=r;
    while (true) { // for(;;)
        i++; while (a[i]<p) i++;
        j--; while (a[j]>p) j--;        ] no bounds check
        if (i>=j) break;
        swap a[i], a[j]
    }
    return j+1;
}
```




Ex/
p=5
5 2 6 7 1 9 3 8
i
j
≤5
≥5
3 2 6 7 1 9 5 8
i
j
≤5
≥5
3 2 1 7 6 9 5 8
j
i
≥5
≤5
i+1=k