Order Statistics

- Maximum, Minimum $n-1$ comparisons

7	3	1	9	4	8	2	5	0	6

- MinMax
  - $2(n-1)$ comparisons
  - $3n/2$ comparisons
- Max and 2ndMax
  - $(n-1) + (n-2)$ comparisons
  - ???

k-Selection; Median

- Select the k-th smallest item in list
- Naïve Solution
  - Sort;
  - pick the k-th smallest item in sorted list.

    O(n log n) time complexity
- Randomized solution: Average case O(n)
- Improved Solution: worst case O(n)

QuickSort

```
QUICKSORT(array A, int p, int r)
if (p < r)
    then q ← PARTITION(A, p, r)
         QUICKSORT(A, p, q − 1)
         QUICKSORT(A, q + 1, r)
```

To sort array call QUICKSORT($A, 1, length[A]$).

```
PARTITION(array A, int p, int r)
x ← A[r]                              ▷ Choose pivot
i ← p − 1
for j ← p to r − 1
    do if (A[j] ≤ x)
          then i ← i + 1
               exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```

Page 146, CLRS

RandomizedPartition

- RandomizedPartition picks the pivot uniformly at random from among the elements in the list to be partitioned.

Homework

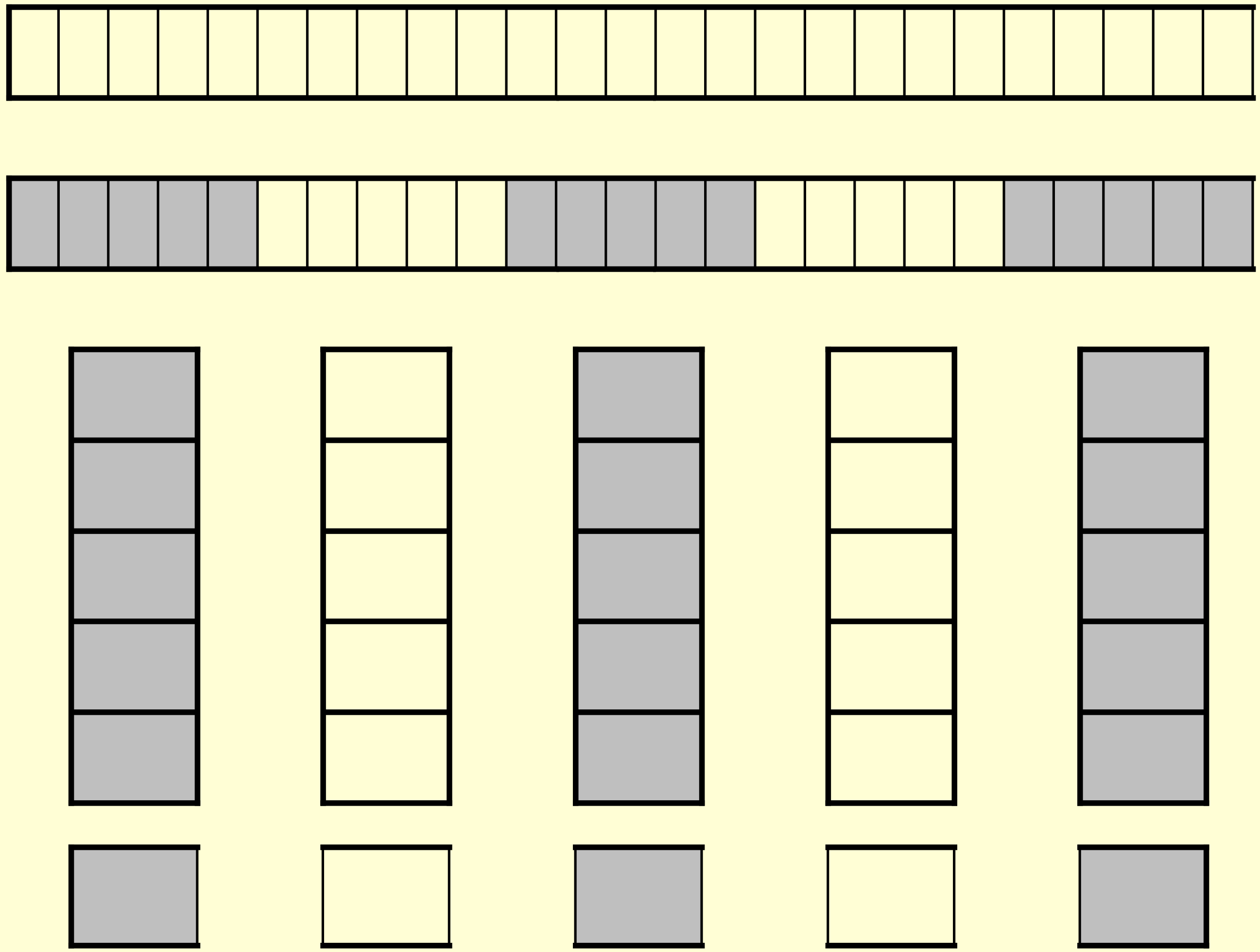
- **Statement of Collaboration**
  - Take it **seriously.**
  - **Reproduce** the statement faithfully and sign it by hand.
  - For each problem, explain **separately** the sources and your collaborations with other people.
  - Your homework **will not be graded** without the signed statement.
- **Extra Credit Problem**
  - You can turn it in any time until second last class day (Dec 4th).
  - You may retry a problem, but don't waste my time.
  - You will not get partial credit on an extra credit problem.
  - Put it on a separate sheet of paper and label it appropriately.

QuickSelect: a variant of QuickSort

```
QUICKSELECT(array A, int k, int p, int r)
    ▷ Select k-th largest in subarray A[p..r]
    if (p = r)
        then return A[p]
    q ← PARTITION(A, p, r)
    i ← q − p + 1        ▷ Compute rank of pivot
    if (i = k)
        then return A[q]
    if (i > k)
        then return QUICKSELECT(A, k, p, q)
        else return QUICKSELECT(A, k − i, q + 1, r)
```

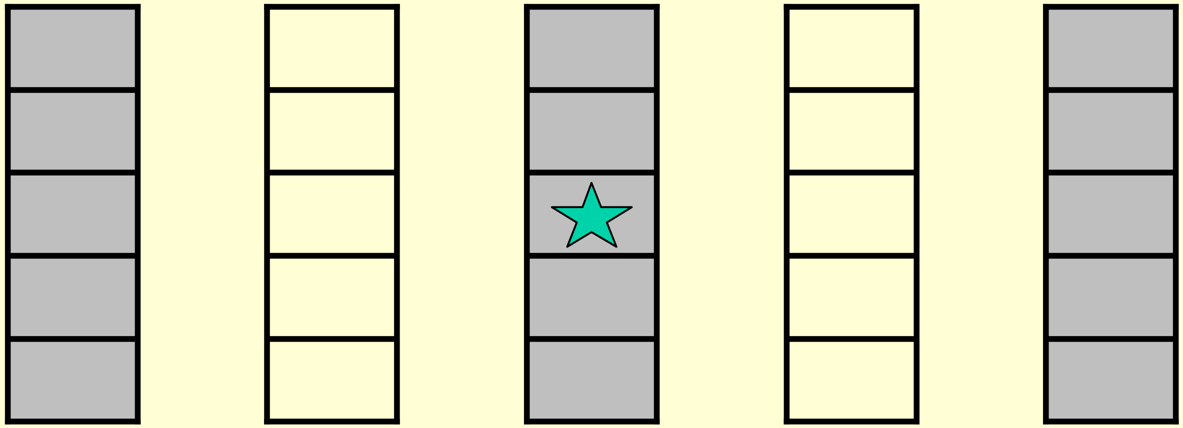
k-Selection & Median: Improved Algorithm

- Start with initial array



k-Selection & Median: Improved Algorithm(Cont'd)

- Use median of medians as pivot



- T(n) < O(n) + T(n/5) + T(3n/4)

ImprovedSelect

ImprovedSelect(*array* A, *int* k, *int* p, *int* r)

    $\triangleright$ Select k-th largest in subarray $A[p..r]$

if $(p = r)$
    then return $A[p]$
    else $N \leftarrow r - p + 1$
Partition $A[p..r]$ into subsets of 5 elements and
collect all medians of subsets in $B[1..\lceil N/5 \rceil]$.
$Pivot \leftarrow$ ImprovedSelect$(B, 1, \lceil N/5 \rceil, \lceil N/10 \rceil)$
$q \leftarrow$ PivotPartition$(A, p, r, Pivot)$
$i \leftarrow q - p + 1$ $\triangleright$ Compute rank of pivot
if $(i = k)$
    then return $A[q]$
if $(i > k)$
    then return ImprovedSelect$(A, k, p, q - 1)$
    else return ImprovedSelect$(A, k - i, q + 1, r)$

PivotPartition

PIVOTPARTITION(*array* A, *int* p, *int* r, *item Pivot*)

$\triangleright$ Partition using provided *Pivot*

$i \leftarrow p - 1$

for $j \leftarrow p$ **to** r

  do if $(A[j] \leq Pivot)$

    then $i \leftarrow i + 1$

      exchange $A[i] \leftrightarrow A[j]$

return $i + 1$

Analysis of ImprovedSelect

Number of elements greater than "median of medians" is at least

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

Why?

Our recurrence is given by:

$$T(n) = O(n) + T(\lceil n/5 \rceil) + T(3n/4)$$

Thus there exists a positive constant a such that

$$T(n) \leq an + T(\lceil n/5 \rceil) + T(3n/4)$$

Using the substitution method, let's guess that $T(n) = O(n)$, i.e., $T(n) \leq cn$. Then we need to show that

$$an + c\lceil n/5 \rceil + c(3n/4) \leq cn$$

What positive values of c and n_0 would enforce the above inequality? When $n > 70$, and choosing $c \geq 20a$ will satisfy above inequality.

Data Structure Evolution

- Standard operations on data structures
  - **Search**
  - **Insert**
  - **Delete**
- Linear Lists
  - Implementation: **Arrays** (**Unsorted and Sorted**)
- **Dynamic** Linear Lists
  - Implementation: **Linked Lists**
- **Dynamic** Trees
  - Implementation: **Binary Search Trees**