Exam: Lessons to be learnt

- $(\log n)^2 \neq \log (n^2)$
- If $A > C$ and $B > D$, then $A + B > C + D$
- To disprove $A + B > C + D$
  - it is not enough to show that $A <= C$ **or** $B <= D$
  - it is enough to show that $A <= C$ **and** $B <= D$
- $10/9 < 5/4$
  - because $10/9 = 40/36$, while $5/4 = 45/36$
- $\log_b a = 1$ if $a = b$
- $\log_b a < 1$ if $a < b$
- $\log_b a > 1$ if $a > b$
- Time complexity (#3) is written in terms of n and k because no relationship is known between n and k other than the simple $k <= n$.
- Time complexity (#4) is written in terms of only n because a relationship between n and k is provided.

Exam: Lessons to be learnt

- **Real numbers** are infinite precision numbers and in some cases cannot be written down in their entirety.
- **Theorem**: There are an uncountable number of real numbers between any two real numbers.
- In particular, real numbers cannot be sorted using Bucket sort or radix sort or counting sort even if they are within a range.
- Real numbers stored on a real computer are not really "real numbers" because they are finite precision numbers. We can only approximate real numbers using a computer. Integers can be stored precisely on a computer. The integer n can be stored using roughly $\log_2 n$ bits.

QuickSelect: a variant of QuickSort

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

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

if $(p = r)$

  then return $A[p]$

$q \leftarrow$ PARTITION(A, p, r)

$i \leftarrow q - p + 1$ $\triangleright$ Compute rank of pivot

if $(i = k)$

  then return $A[q]$

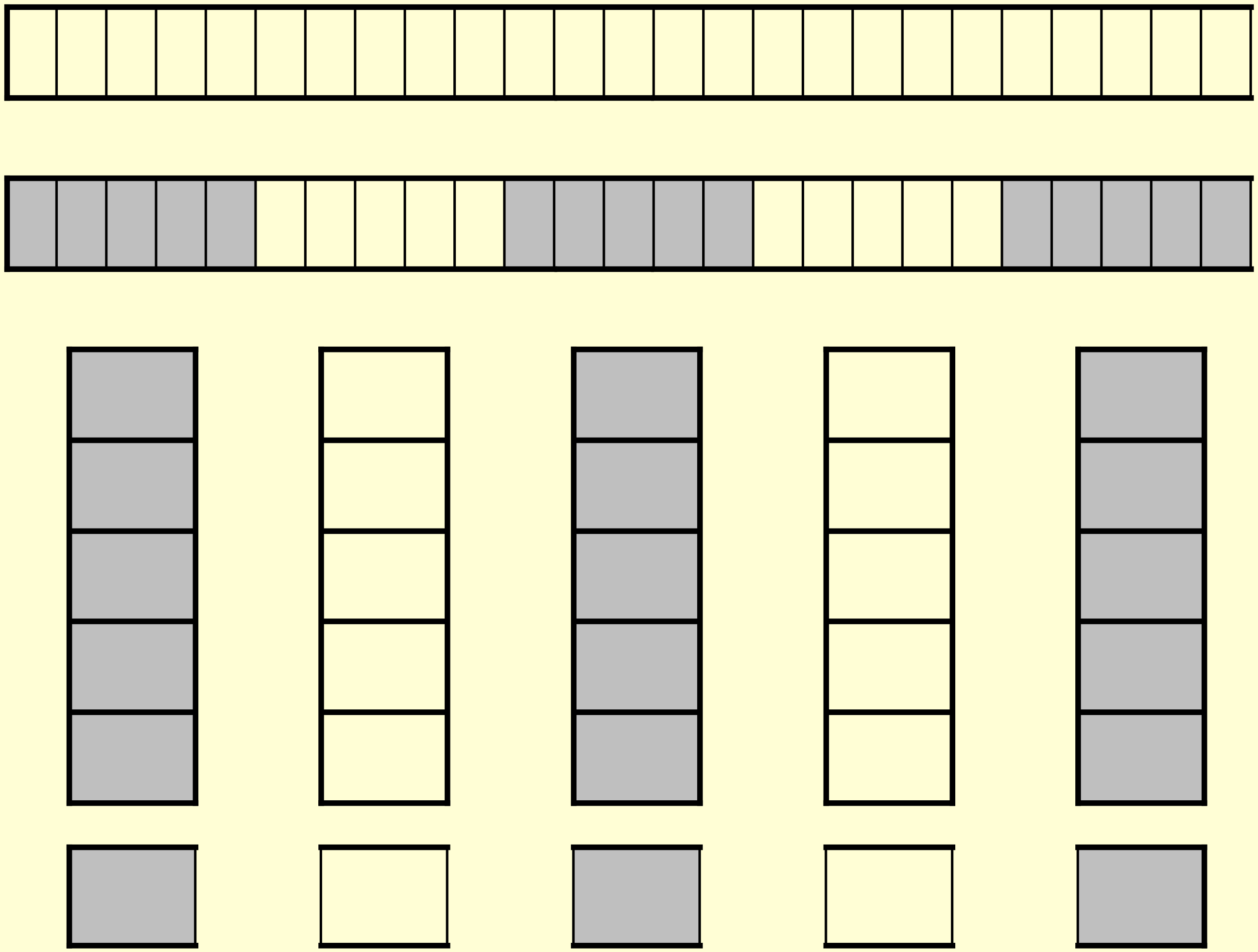
if $(i > k)$

  then return QUICKSELECT(A, k, p, q)

  else return QUICKSELECT$(A, k - i, q + 1, r)$

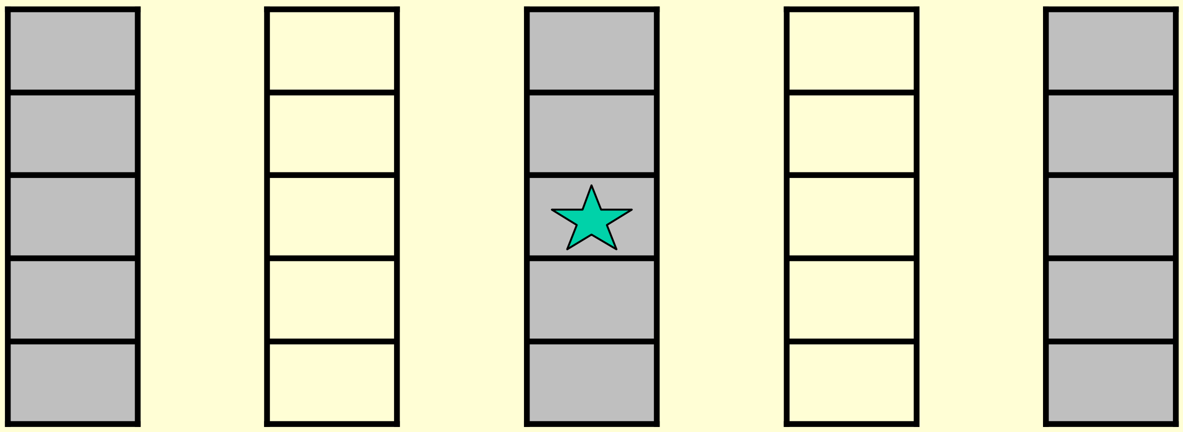
k-Selection & Median: Improved Algorithm

- 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

$\textsc{ImprovedSelect}(array\ A, int\ k, int\ p, int\ r)$

$\triangleright$ 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 $B[1..\lceil N/5 \rceil]$.

$Pivot \leftarrow \textsc{ImprovedSelect}(B, 1, \lceil N/5 \rceil, \lceil N/10 \rceil)$

$q \leftarrow \textsc{PivotPartition}(A, p, r, Pivot)$

$i \leftarrow q - p + 1$ $\triangleright$ Compute rank of pivot

if $(i = k)$

  then return $A[q]$

if $(i > k)$

  then return $\textsc{ImprovedSelect}(A, k, p, q - 1)$

  else return $\textsc{ImprovedSelect}(A, k - i, q + 1, r)$

PivotPartition

```
PIVOTPARTITION(array A, int p, int r, [item Pivot])
    ▷ Partition using provided Pivot
    i ← p − 1
    for j ← p to [r]
            do if (A[j] ≤ Pivot)
                    then i ← i + 1
                            exchange A[i] ↔ A[j]
    return i + 1
```

Analysis of ImprovedSelect

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

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

Why?

Our recurrence is given by:

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

Thus there exists a positive constant a such that

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

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

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

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**

BST: Search

TREESEARCH(*node* x, *key* k)

$\triangleright$ Search for key k in subtree rooted at node x

if ((x = NIL) or ($k = key[x]$))
  then return x
if ($k < key[x]$)
  then return TREESEARCH($left[x], k$)
  else return TREESEARCH($right[x], k$)

Time Complexity: O(h)
h = height of binary search tree

Not O(log n) — Why?

BST: Insert

Time Complexity: O(h)
h = height of binary search tree

```
TreeInsert(tree T, node z)
    ▷ Insert node z in tree T
    y ← NIL
    x ← root[T]
    while (x ≠ NIL)
          do y ← x
             if (key[z] < key[x])
                then x ← left[x]
                else x ← right[x]
    p[z] ← y
    if (y = NIL)
       then root[T] ← z
       else if (key[z] < key[y])
               then left[y] ← z
               else right[y] ← z
```

Search for x in T

Insert x as leaf in T

BST: Delete

Time Complexity: O(h)
h = height of binary search tree

```
TreeDelete(tree T, node z)
    ▷ Delete node z from tree T
if ((left[z] = NIL) or (right[z] = NIL))
    then y ← z
    else y ← Tree-Successor(z)
if (left[y] ≠ NIL)
    then x ← left[y]
    else x ← right[y]
if (x ≠ NIL)
    then p[x] ← p[y]
if (p[y] = NIL)
    then root[T] ← x
    else if (y = left[p[y]])
             then left[p[y]] ← x
             else right[p[y]] ← x
if (y ≠ z)
    then key[z] ← key[y]
         cop y's satellite data into z
return y
```

Set y as the node to be deleted. It has at most one child, and let that child be node x

If y has one child, then y is deleted and the parent pointer of x is fixed.

The child pointers of the parent of x is fixed.

The contents of node z are fixed.

Animations

- **BST:**

  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/BST-Example.html
- **Rotations:**

  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/index2.html
- **RB-Trees:**

  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/RedBlackTree-Example.html