# DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

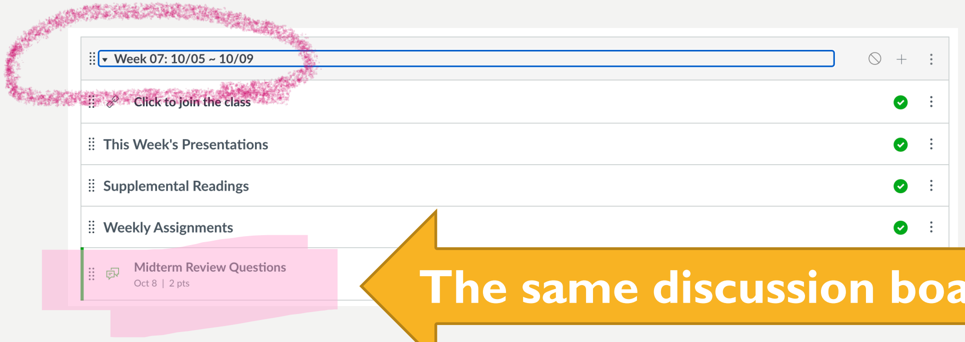
PRUNE AND SEARCH

# MIDTERM

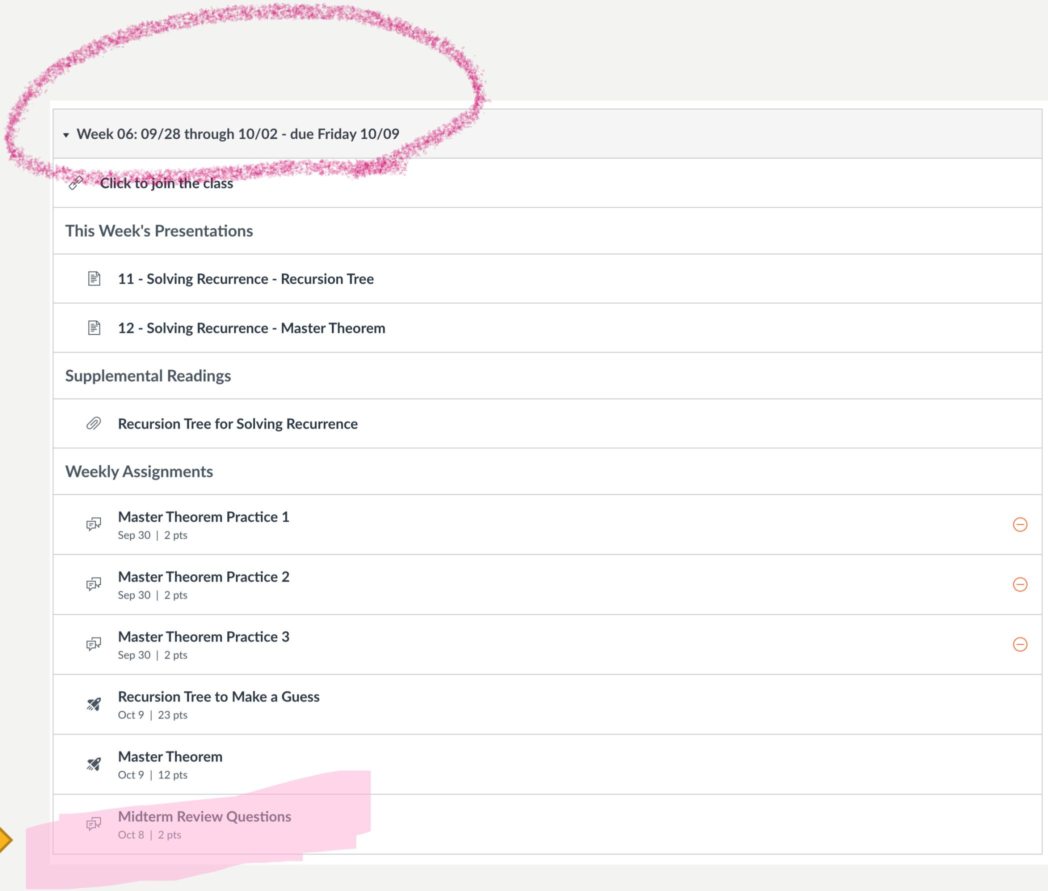
- Exam will be open on Monday Oct 12 and closed at 11:59pm Friday Oct 16 (8th week)
- Topics
  - – Analyze algorithm
    - Correctness (loop invariant)
    - Efficiency (cost-time columns)
  - – Asymptotic notations
  - – Divide and conquer
  - – Solving recurrence (substitution, recursion tree, master theorem)

# MIDTERM REVIEW

- A review of homework assignments will be arranged **next Friday Oct 9th.**
- I will be reviewing homework questions based on demands.
  - Please leave the question number in the discussion board




**The same discussion board!**



# AGENDA

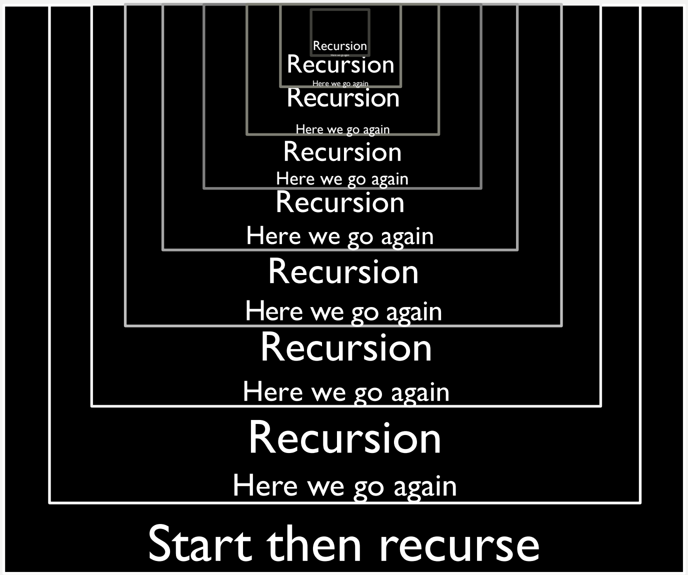
- **The topics moving forward will be assessed in the final exam.**
- Order statistic
- Prune and search steps
- Selection algorithm

# ORDER STATISTIC

- The ***i*th order statistic** of a set of $n$ elements is **the *i*th smallest element**.
- The **median** of $A[1..n]$.
  - The "halfway point" of the set.
    - Regardless of the parity of $n$.
  - Medians occur at $i = \left\lfloor \frac{n+1}{2} \right\rfloor$ and $i = \left\lceil \frac{n+1}{2} \right\rceil$.
    - Names: lower median and upper median, respectively.
    - By convention, we use the lower median.
  - We often use $i = \left\lfloor \frac{n+1}{2} \right\rfloor$ (the lower mid) as the ***halfway point*** (***midpoint***) of an array $A[1..n]$.

# PRUNE AND SEARCH

- Decrease and conquer
  - **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
  - **Prune** the subproblems and eliminate some instances based on certain criteria.
  - **Conquer** the original problem by solving remaining subproblem recursively.
    - As the problem gets smaller, a straight-forward method can be used.




- Example: Binary search, Selection

# SELECTION PROBLEM

- Problem
  - Input: An array $A[1..n]$ that contains $n$ distinct numbers, and an integer $i \in [1, n]$.
  - Output: An element $x \in A$ that $x$ is greater than exactly $i - 1$ other elements of $A$. $\Longleftrightarrow$ **$i$th statistic**
- Example
  - Input: Array $A = \{8, 25, 3, 37, 12, 16, 7, 22\}$, and $i = 4$.
  - Output: $x = \mathbf{12}$
- **Algorithm?**

# SELECTION PROBLEM
# BRAIN STORMING

- Problem
  - Input: An array $A[1..n]$ that contains $n$ distinct numbers, and an integer $i \in [1, n]$.
  - Output: An element $x \in A$ that $x$ is greater than exactly $i - 1$ other elements of $A$.

- How many ways to find the ***i*th statistic** of $A[1..n]$?
- What is their complexities?

# SELECTION PROBLEM
## SOLUTION 1

- Sort the numbers in array $A$ in increasing order Fastest $O(n \cdot \log n)$
- Return $A[i]$. $\Theta(1)$

# SELECTION PROBLEM
# SOLUTION 2

- Perform $i$ scans. Each scan finds the min of the array excluding the min found in previous scans.

| SELECT-BY-SCAN $(A, n, i)$ | | Cost | Time (Worst-case Scenario) |
|---|---|---|---|
| 1 | **for** $j = 1$ **to** $i$ | $\Theta(1)$ | $i+1$ |
| 2 | $k = j$ | $\Theta(1)$ | $i$ |
| 3 | $MIN = A[k]$ | $\Theta(1)$ | $i$ |
| 4 | **for** $k = j$ **to** $n$ | $\Theta(1)$ | $\sum_{j=1}^{i} t_j$ |
| 5 | **if** $A[k] < MIN$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 6 | $MIN = A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 7 | Swap $A[j]$ and $A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |

# SELECTION PROBLEM
## SOLUTION 2 (CONT'D)

- Perform $i$ scans. Each scan finds the min of the array excluding the min found in previous scans.
- Running time $T(n) = \Theta(i) + \Theta(1) + \Theta\left(\sum_{j=1}^{i} t_j\right) + \Theta\left(\sum_{j=1}^{i} (t_j - 1)\right)$, where $t_j$ denotes the # of exe of line 4 for a value of $j$.

| SELECT-BY-SCAN $(A, n, i)$ | | Cost | Time (Worst-case Scenario) |
|---|---|---|---|
| 1 | **for** $j = 1$ **to** $i$ | $\Theta(1)$ | $i + 1$ |
| 2 | $k = j$ | $\Theta(1)$ | $i$ |
| 3 | $MIN = A[k]$ | $\Theta(1)$ | $i$ |
| 4 | **for** $k = j$ **to** $n$ | $\Theta(1)$ | $\sum_{j=1}^{i} t_j$ |
| 5 | **if** $A[k] < MIN$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 6 | $MIN = A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 7 | Swap $A[j]$ and $A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |

# SELECTION PROBLEM
# SOLUTION 2 (CONT'D)

- Perform $i$ scans. Each scan finds the min of the array excluding the min found in previous scans.
- Obviously, for a value of $j$, the **for**-statement (line 4) will execute $n - j + 1 + 1 = \boldsymbol{n - j + 2}$ times.
- Running time
  $T(n) = \Theta(i) + \Theta(1) + \Theta\left(\sum_{j=1}^{i}(\boldsymbol{n - j + 2})\right) + \Theta\left(\sum_{j=1}^{i}(\boldsymbol{n - j + 1})\right)$

| SELECT-BY-SCAN $(A, n, i)$ | | Cost | Time (Worst-case Scenario) |
|---|---|---|---|
| 1 | **for** $j = 1$ **to** $i$ | $\Theta(1)$ | $i + 1$ |
| 2 | $k = j$ | $\Theta(1)$ | $i$ |
| 3 | $MIN = A[k]$ | $\Theta(1)$ | $i$ |
| 4 | **for** $k = j$ **to** $n$ | $\Theta(1)$ | $\sum_{j=1}^{i} t_j$ |
| 5 | **if** $A[k] < MIN$ | $\Theta(1)$ | $\sum_{j=1}^{i}(t_j - 1)$ |
| 6 | $MIN = A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i}(t_j - 1)$ |
| 7 | Swap $A[j]$ and $A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i}(t_j - 1)$ |

# SELECTION PROBLEM
## SOLUTION 2 (CONT'D)

- Perform $i$ scans. Each scan finds the min of the array excluding the min found in previous scans.
- Obviously, for a value of $j$, the **for**-statement (line 4) will execute $n - j + 1 + 1 = \boldsymbol{n - j + 2}$ times.
- Running time $T(n) = \Theta(i) + \Theta(1) +$

$$\Theta\left(i\boldsymbol{n} - \frac{i(1+i)}{2} + \boldsymbol{2}i\right) +$$

$$\Theta\left(i\boldsymbol{n} - \frac{i(1+i)}{2} + \boldsymbol{1}i\right) +$$

| SELECT-BY-SCAN $(A, n, i)$ | | Cost | Time (Worst-case Scenario) |
|---|---|---|---|
| 1 | **for** $j = 1$ **to** $i$ | $\Theta(1)$ | $i + 1$ |
| 2 | $k = j$ | $\Theta(1)$ | $i$ |
| 3 | $MIN = A[k]$ | $\Theta(1)$ | $i$ |
| 4 | **for** $k = j$ **to** $n$ | $\Theta(1)$ | $\sum_{j=1}^{i} t_j$ |
| 5 | **if** $A[k] < MIN$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 6 | $MIN = A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 7 | Swap $A[j]$ and $A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |

# SELECTION PROBLEM
## SOLUTION 2 (CONT'D)

- Perform $i$ scans. Each scan finds the min of the array excluding the min found in previous scans.
- Obviously, for a value of $j$, the **for**-statement (line 4) will execute $n - j + 1 + 1 = \boldsymbol{n - j + 2}$ times.
- Running time
  $T(n) = \Theta(i) + \Theta(1) +$

  $O(\boldsymbol{i^2}) +$

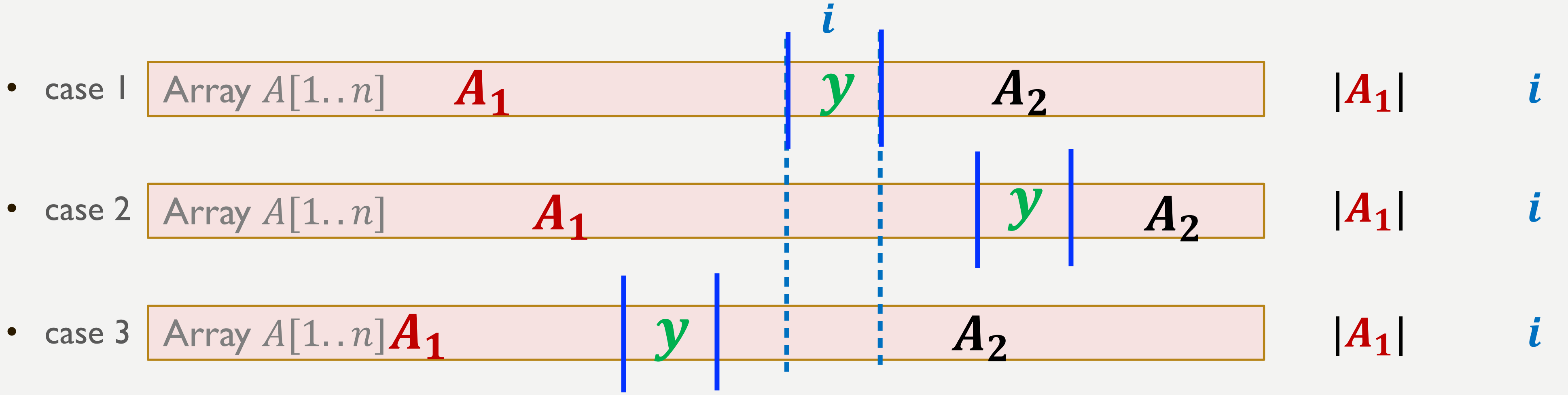
  $O(\boldsymbol{i^2}) +$

  $= O(\boldsymbol{i^2})$

| SELECT-BY-SCAN $(A, n, i)$ | | Cost | Time (Worst-case Scenario) |
|---|---|---|---|
| 1 | **for** $j = 1$ **to** $i$ | $\Theta(1)$ | $i + 1$ |
| 2 | $k = j$ | $\Theta(1)$ | $i$ |
| 3 | $MIN = A[k]$ | $\Theta(1)$ | $i$ |
| 4 | **for** $k = j$ **to** $n$ | $\Theta(1)$ | $\sum_{j=1}^{i} t_j$ |
| 5 | **if** $A[k] < MIN$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 6 | $MIN = A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |
| 7 | Swap $A[j]$ and $A[k]$ | $\Theta(1)$ | $\sum_{j=1}^{i} (t_j - 1)$ |

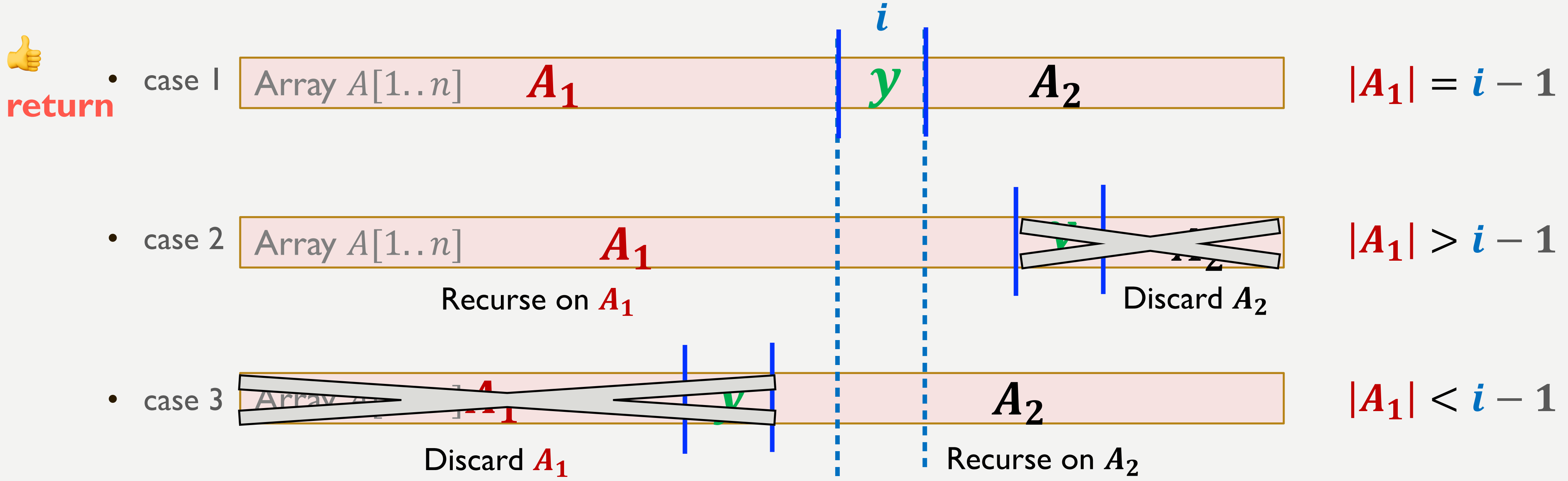
# SELECTION PROBLEM
## SOLUTION 3

- **Step 1**: Randomly pick a number $\boldsymbol{y}$ in $A$. ***Divide*** $A$ into two subarrays $A_1$ and $A_2$, such that
  - All the numbers in $\boldsymbol{A_1}$ are $< \boldsymbol{y}$; all the numbers in $\boldsymbol{A_2}$ are $> \boldsymbol{y}$
  - What the relationship between $|\boldsymbol{A_1}|$ and $\boldsymbol{i}$?



# SELECTION PROBLEM
## SOLUTION 3

- **Step 2**: **Prune** the subarrays based on the relation between $|A_1|$ and $i$.



# RANDOMIZED SELECT ALGORITHM
## SOLUTION 3

- Let the input be $(A, p, r, i)$
  - $A$ is the array
  - $p$ is the low index
  - $r$ is the high index
  - $i$ is the $i$th order statistic of $A$ that we wish to find

| | | RANDOMIZED-SELECT $(A, p, r, i)$ |
|---|---|---|
| Bottoms-out case | 1 | **if** $p == r$ |
| | 2 |     **return** $A[p]$ |
| Partition | 3 | $q =$ **RANDOMIZED-PARTITION** $(A, p, r)$ |
| | 4 | $k = q - p + 1$ |
| Prune (5, 7)-and-search (8,9) | 5 | **if** $i == k$ |
| | 6 |     **return** $A[q]$ |
| | 7 | **elseif** $i < k$ |
| | 8 |     **return** RANDOMIZED-SELECT$(A, p, q-1, i)$ |
| | 9 | **else return** RANDOMIZED-SELECT$(A, q+1, r, i-k)$ |

# RANDOMIZED SELECT ALGORITHM
## WORKING PROCESS CASE 1

- After RANDOMIZED-PARTITIONing the input $A[p..r]$, use $\boldsymbol{q}$ to denote the **pivot** index.
- Use the instance $A = \{8, 25, 3, 37, 12, 10, 35, 15, 28, 16, 7, \mathbf{22}\}$, $\boldsymbol{i} = 8$.
- **Compare** the length of $A[p..q] = k$ with $i$.
  - **Case 1**

| $A[p..\boldsymbol{q}-1]$ | | | | | | | $A[\boldsymbol{q}]$ | $A[\boldsymbol{q}+1..r]$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 3 | 12 | 10 | 15 | 16 | 7 | **22** | 25 | 37 | 35 | 28 |

  - $\boldsymbol{q} =$ ________________; the length of $A[p..\boldsymbol{q}] = k =$ ______________;
  - $i$ ________ $k$, the $i$th order statistic of $A$ is __________________.

# RANDOMIZED SELECT ALGORITHM
## WORKING PROCESS CASE 2

- After RANDOMIZED-PARTITIONing the input $A[p..r]$, use $\boldsymbol{q}$ to denote the **pivot** index.
- Use the instance $A = \{8, 25, 3, 37, 12, 10, 35, 15, 28,16, 7, \mathbf{22}\}, \boldsymbol{i} = 8$.
- **Compare** the length of $A[p..q] = k$ with $i$.
  - **Case 2**

| $A[p..\boldsymbol{q} - 1]$ | | | | | | | | | $A[\boldsymbol{q}]$ | $A[\boldsymbol{q} + 1..r]$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 25 | 3 | 12 | 10 | 15 | 16 | 7 | **22** | **28** | 37 | 35 |

  - $\boldsymbol{q} =$ ________________; the length of $A[p..\boldsymbol{q}] = k =$ ______________;
  - $i$ ________ $k$, the $i$th order statistic of $A$ lies in subarray __________________.

**Recurse** on $A[$ _____________ $]$. Find the _________ th order statistic of $A[$ _____________ $]$.

# RANDOMIZED SELECT ALGORITHM
## WORKING PROCESS CASE 3

- After RANDOMIZED-PARTITIONing the input $A[p..r]$, use $\boldsymbol{q}$ to denote the **pivot** index.
- Use the instance $A = \{8, 25, 3, 37, 12, 10, 35, 15, 28, 16, 7, \mathbf{22}\}$, $\boldsymbol{i} = 8$.
- **Compare** the length of $A[p..q] = k$ with $i$.
  - **Case 3**

| $A[p..\boldsymbol{q}-1]$ | | | | $A[\boldsymbol{q}]$ | $A[\boldsymbol{q}+1..r]$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 3 | 10 | 7 | 12 | 25 | 37 | 35 | 15 | 28 | 16 | 22 |
| | | | | | | | | 1 | | 2 | 3 |

  - $\boldsymbol{q} =$ ________________; the length of $A[p..\boldsymbol{q}] = k =$ ______________;
  - $i$ ________ $k$, the $i$th order statistic of $A$ lies in subarray _________________.

**Is 22 still the 8th order statistic of $A[q+1..r]$?** What is the next step and what is the new goal?

# RANDOMIZED SELECT ALGORITHM
# WORKING PROCESS CASE 3

- After RANDOMIZED-PARTITIONing the input $A[p..r]$, use $\boldsymbol{q}$ to denote the **pivot** index.
- Use the instance $A = \{8, 25, 3, 37, 12, 10, 35, 15, 28, 16, 7, \mathbf{22}\}$, $\boldsymbol{i} = 8$.
- **Compare** the length of $A[p..q] = k$ with $i$.
  - **Case 3**

| $A[p..\boldsymbol{q}-1]$ | | | | $A[\boldsymbol{q}]$ | $A[\boldsymbol{q}+1..r]$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 3 | 10 | 7 | 12 | 25 | 37 | 35 | 15 | 28 | 16 | 22 |

  - $\boldsymbol{q} =$ ________________; the length of $A[p..\boldsymbol{q}] = k =$ ______________;
  - $i$ ________ $k$, the $i$th order statistic of $A$ lies in subarray _________________.

**Recurse** on $A[$ _____________ $]$. Find the _________th order statistic of $A[$ _____________ $]$.

# RANDOMIZED SELECT ALGORITHM
## COST-TIME COLUMNS

- Complete the **cost-time** columns of the RANDOMIZED-SELECT $(A, p, r, i)$ on input $A[p..r]$.
  - Use function $\boldsymbol{f(r-p+1)}$ to denote the cost of RANDOMIZED-PARTITION $(A, p, r)$

| RANDOMIZED-SELECT $(A, p, r, i)$ | Cost | Time (Best) | Time (Worst) |
|---|---|---|---|
| **if** $p == r$ | | | |
|     **return** $A[p]$ | | | |
| $\boldsymbol{q}$ = **RANDOMIZED-PARTITION** $(\boldsymbol{A, p, r})$ | | | |
| $k = q - p + 1$ | | | |
| **if** $i == k$ | | | |
|     **return** $A[q]$ | | | |
| **elseif** $i < k$ | | | |
|     **return** RANDOMIZED-SELECT$(A, p, q-1, i)$ | | | |
| **else return** RANDOMIZED-SELECT$(A, q+1, r, i-k)$ | | | |

# RANDOMIZED SELECT ALGORITHM
## COST-TIME COLUMNS

- Complete the **cost-time** columns of the RANDOMIZED-SELECT $(A, p, r, i)$ on input $A[p..r]$.
  - Use function $\boldsymbol{f(r-p+1)}$ to denote the cost of RANDOMIZED-PARTITION $(A, p, r)$

| RANDOMIZED-SELECT $(A, p, r, i)$ | Cost | Time (Best) | Time (Norm.) |
|---|---|---|---|
| **if** $p == r$ | $\Theta(1)$ | base | base |
|     **return** $A[p]$ | $\Theta(1)$ | base | base |
| $\boldsymbol{q}$ = **RANDOMIZED-PARTITION** $\boldsymbol{(A, p, r)}$ | $\boldsymbol{f(r-p+1)}$ | 1 | 1 |
| $k\ = q - p + 1$ | $\Theta(1)$ | 1 | 1 |
| **if** $i == k$ | $\Theta(1)$ | 1 | 1 |
|     **return** $A[q]$ | $\Theta(1)$ | **1** | **0** |
| **elseif** $i < k$ | $\Theta(1)$ | **0** | **1** |
|     **return** RANDOMIZED-SELECT$(A, p, q-1, i)$ | $T(\boldsymbol{q} - p)$ | **0** | **0/1** |
| **else return** RANDOMIZED-SELECT$(A, q+1, r, i-k)$ | $T(r - \boldsymbol{q})$ | **0** | **1/0** |

# RANDOMIZED SELECT ALGORITHM
## RUNNING TIME FUNCTION

- **Best**-of-all scenario
  - $x =$ ______________.
- **Normal** scenario
  - $x$ lies in ___________.
- The running time of the normal scenario $T(r-p+1)$ $= \boldsymbol{\Theta(1)} +$ $\boldsymbol{f(r-p+1)} + \boldsymbol{max}\,\{T(\boldsymbol{q}-p), T(r-\boldsymbol{q})\}$

| RANDOMIZED-SELECT $(A, p, r, i)$ | | Cost | Time (Best) | Time (Norm.) |
|---|---|---|---|---|
| 1 | **if** $p == r$ | $\Theta(1)$ | base | base |
| 2 | **return** $A[p]$ | $\Theta(1)$ | base | base |
| 3 | $\boldsymbol{q}$ = **RANDOMIZED-PARTITION** $(\boldsymbol{A, p, r})$ | $\boldsymbol{f(r-p+1)}$ | 1 | 1 |
| 4 | $k\ = q - p + 1$ | $\Theta(1)$ | 1 | 1 |
| 5 | **if** $i == k$ | $\Theta(1)$ | 1 | 1 |
| 6 | **return** $A[q]$ | $\Theta(1)$ | **1** | **0** |
| 7 | **elseif** $i < k$ | $\Theta(1)$ | **0** | **1** |
| 8 | **return** RANDOMIZED-SELECT$(A, p, q-1, i)$ | $T(\boldsymbol{q}-p)$ | **0** | **0/1** |
| 9 | **else return** RANDOMIZED-SELECT$(A, q+1, r, i-k)$ | $T(r-\boldsymbol{q})$ | **0** | **1/0** |

# RANDOMIZED PARTITION ALGORITHM

- The algorithm uses a random number generator to generate a value $i \in [p, r]$.
- Then it moves $A[i]$ to **the tail of (sub)array $A[p..r]$**.
- Lastly, it applies normal partition on $A[p..r]$.

| RANDOMIZED-PARTITION $(A, p, r)$ | |
|---|---|
| 1 | $i =$ RANDOM $(p, r)$ |
| 2 | exchange $A[r]$ with $A[i]$ |
| 3 | **return** PARTITION $(A, p, r)$ |

# RANDOMIZED PARTITION ALGORITHM RUNNING TIME

- Complete the **cost** and **time** columns.
  - Use $\boldsymbol{f'(r-p+1)}$ to denote the cost of partitioning $A[p..r]$ around $A[r]$.
- The running time of the algorithm on (sub)array $A[p..r]$
  $\boldsymbol{f(r-p+1)}$
  $= \boldsymbol{\Theta(1)} + \boldsymbol{f'(r-p+1)}$

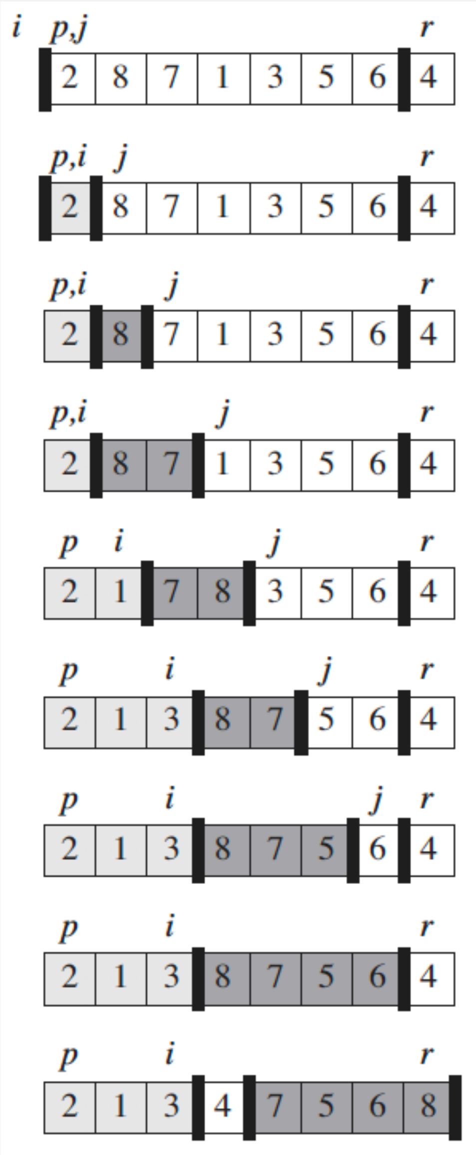
| | RANDOMIZED-**PARTITION** $(A, p, r)$ | Cost | Time (Best) | Time (Norm.) |
|---|---|---|---|---|
| 1 | $i =$ RANDOM $(p, r)$ | $\Theta(1)$ | 1 | 1 |
| 2 | exchange $A[r]$ with $A[i]$ | $\Theta(1)$ | 1 | 1 |
| 3 | **return PARTITION** $(A, p, r)$ | $\boldsymbol{f'(r-p+1)}$ | 1 | 1 |

# THE PARTITION PROCEDURE

- The **PARTITION** procedure uses **the tail of (sub)array $A[p..r]$** as the **pivot** to partition $A[p..r]$ in to three parts.
  - Subarray $A[p..i]$ that contains the elements $<$ pivot
  - $A[i+1]$ the pivot
  - Subarray $A[i+2..r]$ that contains the elements $>$ pivot.

**PARTITION** $(A, p, r)$

$x = A[r]$
$i = p - 1$
**for** $j = p$ **to** $r - 1$
    **if** $A[j] \leq x$
        $i = i + 1$
        exchange $A[i]$ with $A[j]$
exchange $A[i+1]$ with $A[r]$
**return** $i + 1$



# THE PARTITION PROCEDURE RUNNING TIME

- Best-case scenario
  - ______________________________.
- Worst-case scenario
  - ______________________________.
- The running time of **PARTITION**ing (sub)array $A[p..r]$ in the worst-case
  $\boldsymbol{f'(r-p+1)}$
  $= \boldsymbol{\Theta(1)} + \Theta(r-p) + \boldsymbol{\Theta(r-p-1)}$
  $= \boldsymbol{\Theta(1) + \Theta(r-p)}$

| **PARTITION** $(A, p, r)$ | Cost | Time (Best) | Time (Worst) |
|---|---|---|---|
| $x = A[r]$ | $\Theta(1)$ | 1 | 1 |
| $i = p - 1$ | $\Theta(1)$ | 1 | 1 |
| **for** $j = p$ **to** $r - 1$ | $\Theta(1)$ | $r - p$ | $r - p$ |
| **if** $A[j] \leq x$ | $\Theta(1)$ | $r - p - 1$ | $r - p - 1$ |
| $i = i + 1$ | $\Theta(1)$ | $\mathbf{0}$ | $\boldsymbol{r - p - 1}$ |
| exchange $A[i]$ with $A[j]$ | $\Theta(1)$ | $\mathbf{0}$ | $\boldsymbol{r - p - 1}$ |
| exchange $A[i + 1]$ with $A[r]$ | $\Theta(1)$ | 1 | 1 |
| **return** $i + 1$ | $\Theta(1)$ | 1 | 1 |

# RANDOMIZED PARTITION ALGORITHM RUNNING TIME (COMING BACK)

- The running time of the algorithm on (sub)array $A[p..r]$
  $f(r-p+1)$
  $= \Theta(1) + f'(r-p+1)$
  , where $f'(r-p+1)$ is the running time of the **PARTITION** procedure.

| RANDOMIZED-**PARTITION** $(A, p, r)$ | | Cost | Time (Best) | Time (Norm.) |
|---|---|---|---|---|
| 1 | $i =$ RANDOM $(p, r)$ | $\Theta(1)$ | 1 | 1 |
| 2 | exchange $A[r]$ with $A[i]$ | $\Theta(1)$ | 1 | 1 |
| 3 | **return PARTITION** $(A, p, r)$ | $f'(r-p+1)$ | 1 | 1 |

- $f'(r-p+1) = \Theta(1) + \Theta(r-p)$
- $f(r-p+1) = \Theta(1) + \Theta(1) + \Theta(r-p) = \Theta(1) + \Theta(r-p)$

# RANDOMIZED SELECT ALGORITHM
## RUNNING TIME FUNCTION (BACK)

- The running time of the normal scenario $T(r-p+1)$
  $= \mathbf{\Theta(1)} + \mathbf{\Theta(1)} + \boldsymbol{\Theta(r-p)} + \boldsymbol{max}\{T(\boldsymbol{q}-p), T(r-\boldsymbol{q})\}$
  $= \Theta(1) + \boldsymbol{\Theta(r-p)} + \boldsymbol{max}\{T(\boldsymbol{q}-p), T(r-\boldsymbol{q})\}$

| | RANDOMIZED-SELECT $(A, p, r, i)$ | Cost | Time (Best) | Time (Norm) |
|---|---|---|---|---|
| 1 | **if** $p == r$ | $\Theta(1)$ | base | base |
| 2 | **return** $A[p]$ | $\Theta(1)$ | base | base |
| 3 | $\boldsymbol{q}$ = **RANDOMIZED-PARTITION** $(\boldsymbol{A, p, r})$ | $\boldsymbol{f(r-p+1)}$ | 1 | 1 |
| 4 | $k = q - p + 1$ | $\Theta(1)$ | 1 | 1 |
| 5 | **if** $i == k$ | $\Theta(1)$ | 1 | 1 |
| 6 | **return** $A[q]$ | $\Theta(1)$ | **1** | **0** |
| 7 | **elseif** $i < k$ | $\Theta(1)$ | **0** | **1** |
| 8 | **return** RANDOMIZED-SELECT$(A, p, q-1, i)$ | $T(\boldsymbol{q}-p)$ | **0** | **0/1** |
| 9 | **else return** RANDOMIZED-SELECT$(A, q+1, r, i-k)$ | $T(r-\boldsymbol{q})$ | **0** | **1/0** |

# APPLY RANDOMIZED-SELECT ON $A[1..n]$

- Call RANDOMIZED-SELECT $(A, \_\_\_, \_\_\_, i)$
  - $n =$ ______________.
- The running time of the worst-case scenario

$T(r-p+1)$
$= \Theta(1) + \Theta(r-p) + \max\{T(q-p), T(r-q)\}$
$= \Theta(1) + \Theta(n-1) + \max\{T(q-1), T(n-q)\} = T(n)$

| RANDOMIZED-SELECT $(A,p,r,i)$ | | Cost | Time (Best) | Time (Norm.) |
|---|---|---|---|---|
| 1 | **if** $p == r$ | $\Theta(1)$ | base | base |
| 2 | **return** $A[p]$ | $\Theta(1)$ | base | base |
| 3 | $q =$ **RANDOMIZED-PARTITION** $(A,p,r)$ | $f(r-p+1)$ | 1 | 1 |
| 4 | $k = q-p+1$ | $\Theta(1)$ | 1 | 1 |
| 5 | **if** $i == k$ | $\Theta(1)$ | 1 | 1 |
| 6 | **return** $A[q]$ | $\Theta(1)$ | **1** | **0** |
| 7 | **elseif** $i < k$ | $\Theta(1)$ | **0** | **1** |
| 8 | **return** RANDOMIZED-SELECT$(A,p,q-1,i)$ | $T(q-p)$ | **0** | **0/1** |
| 9 | **else return** RANDOMIZED-SELECT$(A,q+1,r,i-k)$ | $T(r-q)$ | **0** | **1/0** |

# APPLY RANDOMIZED-SELECT ON $A[1..n]$ BEST CASE

- Simplify the running time function of the normal scenario

$$T(n) = \Theta(n) + max\{T(q-1), T(n-q)\}$$

  - The **best-case** scenario of the normal case
    - $q =$ ______________
    - yielding $T(n) =$ _____________________.
    - $T(n) = O($_____$)$.

# APPLY RANDOMIZED-SELECT ON $A[1..n]$ WORST CASE

- Simplify the running time function of the normal scenario

$$T(n) = \Theta(n) + max\,\{T(q-1), T(n-q)\}$$

  - The **worst**-of-all scenario
    - $q =$ ______________
    - yielding $T(n) =$ _____________________.
    - $T(n) = O($_____$)$.

# RANDOMIZED-SELECT TIME COMPLEXITY

- The **best-case** scenario of the normal case $T(n) = \underline{\quad \boldsymbol{\Theta(n) + T(n/2)} \quad} = O(\underline{\quad \boldsymbol{n} \quad})$.
- The **worst-case** scenario of the normal case $T(n) = \underline{\quad \boldsymbol{\Theta(n) + T(n-1)} \quad} = O(\underline{\quad \boldsymbol{n^2} \quad})$.
- The ***expected*** time complexity
  - We can find any order statistic in ***expected*** linear time, assuming that the elements are distinct.

$$\boldsymbol{E}[T(n)] = O(n)$$

    - See textbook page 217 – 219 for proof.

- **Improvement?**

# NEXT UP
## SELECT IN WORST-CASE LINEAR TIME

# REFERENCE