Selection Problem: WHAT

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$,

find the $i$th smallest element in the list.

AKA: Find the element that is greater than exactly $i-1$ other elements.

What algorithm would you choose if $i=1$?

\[
\begin{align*}
\text{find min} & \quad \text{if } i = 1 \\
\text{find max} & \quad \text{if } i = n
\end{align*}
\]


Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$,

find the $i^{th}$ smallest element in the list.

*What algorithm would you choose in general?*
Selection Problem: HOW

Given list of distinct integers \( a_1, a_2, \ldots, a_n \) and integer \( i, 1 \leq i \leq n \), find the \( i \)th smallest element in the list.

What algorithm would you choose in general? Can sorting help?

Algorithm: first sort list and then step through to find \( i \)th smallest. What's its runtime?

A. \( \Theta(1) \)
B. \( \Theta(n) \)
C. \( \Theta(n \log n) \)
D. \( \Theta(n^2) \)
E. None of the above
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

*What algorithm would you choose in general? Different strategy …*

Pick random list element called “pivot.”

Partition list into those smaller than pivot, those bigger than pivot.

Using $i$ and size of partition sets, determine in which set to continue looking.
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot. Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$ $i = 3$
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$  $i = 3$  Random pivot: 17

$S = 3, 9, 2$
$B = 42, 19, 21$

$O(n)$ to make the list.
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot. Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$ $i = 3$ Random pivot: 17

Smaller than 17: 3, 8, 2
Bigger than 17: 42, 19, 21

17
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$  
    $i = 3$  
    Random pivot: 17

Smaller than 17: 3, 8, 2  
Bigger than 17: 42, 19, 21

Has 3 elements so third smallest must be in this set
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$  $i = 3$  Random pivot: 17
New list: $3, 8, 2$  $i = 3$
Selection Problem: HOW

Given list of distinct integers \( a_1, a_2, \ldots, a_n \) and integer \( i, 1 \leq i \leq n \), find the \( i^{th} \) smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using \( i \) and size of partition sets, determine in which set to continue looking.

ex. \( 17, 42, 3, 8, 19, 21, 2 \) \( i = 3 \) Random pivot: 17
New list: 3, 8, 2 \( i = 3 \) Random pivot: 8
Selection Problem: HOW

Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$  
New list: $3, 8, 2$  
Random pivot: $17$  
Smaller than $8$: $3, 2$  
Bigger than $8$: 

$i = 3$
Selection Problem: HOW

Given list of distinct integers \( a_1, a_2, \ldots, a_n \) and integer \( i \), \( 1 \leq i \leq n \), find the \( i^{\text{th}} \) smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using \( i \) and size of partition sets, determine in which set to continue looking.

ex. \[17, 42, 3, 8, 19, 21, 2\] \( i = 3 \) Random pivot: 17
New list: 3, 8, 2 \( i = 3 \) Random pivot: 3
Smaller than 8: 3, 2
Bigger than 8: 5, 2, 3

Has 2 elements so third smallest must be "next" element, i.e., 8
Given list of distinct integers $a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, find the $i^{th}$ smallest element in the list.

Pick random list element called “pivot.”
Partition list into those smaller than pivot, those bigger than pivot.
Using $i$ and size of partition sets, determine in which set to continue looking.

ex. $17, 42, 3, 8, 19, 21, 2$ $i = 3$ Random pivot: 17
New list: 3, 8, 2 $i = 3$ Random pivot: 8
Smaller than 8: 3, 2 Bigger than 8:

Return 8 compare to original list: 17, 42, 3, 8, 19, 21, 2
Selection Problem: HOW

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$,

Algorithm will incorporate both randomness and recursion!
Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, \[\text{RandSelect}(A, i)\]

1. If $n=1$ return $a_1$

What are we doing in this first line?

A. Establishing the base case of the recursion.
B. Establishing the induction step.
C. Randomly picking a pivot.
D. Randomly returning a list element.
E. None of the above.
Selection Problem: HOW

Given list of distinct integers \( A = a_1, a_2, \ldots, a_n \) and integer \( i, 1 \leq i \leq n \),

\[ \text{RandSelect}(A,i) \]

1. If \( n=1 \) return \( a_1 \)
2. Initialize lists \( S \) and \( B \).
3. Pick integer \( j \) uniformly at random from 1 to \( n \).
4. For each index \( k \) from 1 to \( n \) (except \( j \)):
   5. if \( a_k < a_j \), add \( a_k \) to the list \( S \).
   6. if \( a_k > a_j \), add \( a_k \) to the list \( B \).
Selection Problem: HOW

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i, 1 \leq i \leq n$, $\text{RandSelect}(A, i)$

1. If $n=1$ return $a_1$
2. Initialize lists $S$ and $B$.
3. Pick integer $j$ uniformly at random from 1 to $n$.
4. For each index $k$ from 1 to $n$ (except $j$):
   5. if $a_k < a_j$, add $a_k$ to the list $S$.
   6. if $a_k > a_j$, add $a_k$ to the list $B$.
7. Let $s$ be the size of $S$.
8. If $s = i-1$, return $a_j$. 


Selection Problem: HOW

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, $\text{RandSelect}(A, i)$

1. If $n=1$ return $a_1$
2. Initialize lists $S$ and $B$.
3. Pick integer $j$ uniformly at random from 1 to $n$.
4. For each index $k$ from 1 to $n$ (except $j$):
   5. if $a_k < a_j$, add $a_k$ to the list $S$.
   6. if $a_k > a_j$, add $a_k$ to the list $B$.
7. Let $s$ be the size of $S$.
8. If $s = i-1$, return $a_j$.
9. If $s > i$, return $\text{RandSelect}(S, i)$.
10. If $s < i$, return $\text{RandSelect}(B, \_???\_)$.

What's the right way to fill in this blank?
A. $i$
B. $s$
C. $i+s$
D. $i-(s+1)$
E. None of the above.
Selection Problem: WHEN

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i, 1 \leq i \leq n,$

$\text{RandSelect}(A,i)$

1. If $n=1$ return $a_1$
2. Initialize lists $S$ and $B.$
3. Pick integer $j$ uniformly at random from 1 to $n.$
4. For each index $k$ from 1 to $n$ (except $j$):
   5. if $a_k < a_j,$ add $a_k$ to the list $S.$
   6. if $a_k > a_j,$ add $a_k$ to the list $B.$
7. Let $s$ be the size of $S.$
8. If $s = i-1,$ return $a_j.$
9. If $s \geq i,$ return $\text{RandSelect}(S, i).$
10. If $s < i,$ return $\text{RandSelect}(B, i-(s+1)).$

What input gives the best-case performance of this algorithm?

A. When element we're looking for is the first in list.
B. When element we're looking for is $i^{\text{th}}$ in list.
C. When element we're looking for is in the middle of the list.
D. When element we're looking for is last in list.
E. None of the above.
Selection Problem: WHEN

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, $\text{RandSelect}(A, i)$

1. If $n=1$ return $a_1$
2. Initialize lists $S$ and $B$.
3. Pick integer $j$ uniformly at random from 1 to $n$.
4. For each index $k$ from 1 to $n$ (except $j$):
   5. if $a_k < a_j$, add $a_k$ to the list $S$.
   6. if $a_k > a_j$, add $a_k$ to the list $B$.
7. Let $s$ be the size of $S$.
8. If $s = i-1$, return $a_j$.
9. If $s \geq i$, return $\text{RandSelect}(S, i)$.
10. If $s < i$, return $\text{RandSelect}(B, i-(s+1))$.

Performance depends on more than the input!
Selection Problem: WHEN

Given list of distinct integers $A = a_1, a_2, \ldots, a_n$ and integer $i$, $1 \leq i \leq n$, 

$\text{RandSelect}(A,i)$

1. If $n=1$ return $a_1$
2. Initialize lists $S$ and $B$.
3. **Pick integer $j$ uniformly at random from 1 to $n$.**
4. For each index $k$ from 1 to $n$ (except $j$):
   5. if $a_k < a_j$, add $a_k$ to the list $S$.
   6. if $a_k > a_j$, add $a_k$ to the list $B$.
7. Let $s$ be the size of $S$.
8. If $s = i-1$, return $a_j$.
9. If $s \geq i$, return $\text{RandSelect}(S, i)$.
10. If $s < i$, return $\text{RandSelect}(B, i-(s+1))$.

Minimum time if we happen to pick pivot which is the $i^{th}$ smallest list element.

In this case, what’s the runtime?
A. $\Theta(1)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$
E. None of the above
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the worst case position of the pivot?
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the worst case position of the pivot? 1 or n

Then the set S has n-1 elements and the recursion is:

\[ T(n) = T(n - 1) + O(n) \]

\[ T(n) = O(n^2) \]
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the best case position of the pivot?

Actually pick the ith smallest.

What if you don’t pick ith smallest.
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the best case position of the pivot? around n/2

Then the sets S and B have around n/2 many elements each and the recursion looks like:

$$T(n) = T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = O(n^d) = O(n)$$
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the best case position of the pivot? around n/2

Then the sets S and B have around n/2 many elements each and the recursion looks like:

\[ T(n) = T \left( \frac{n}{2} \right) + O(n) \]
Selection Problem: WHEN

How can we give a time analysis for an algorithm that is allowed to pick and then use random numbers?

What is the expected runtime?

If we could figure out how often the pivot is chosen “close” to the median then we could have an idea of how fast the algorithm is expected to run.
Expected runtime

If you randomly select the $i$th element, then your list will be split into a list of length $i$ and a list of length $n-i$.

So when we recurse on the sublists, then the larger out of the two sublists has size $\max(i, n-i)$.
Clearly, the split with the smallest maximum size is when $i = n/2$

and worst case is $i = n$ or $i = 1$. 
What is the expected runtime?

Well what is our random variable?

For each input and sequence of random choices of pivots, The random variable is the runtime of that particular outcome.
So if we want to find the expected runtime, we must sum over all possibilities of choices.

Let $ET(n)$ be the expected runtime. Then

$$ET(n) = \frac{1}{n} \sum_{i=1}^{n} ET(\max(i, n-i)) + O(n)$$
What is the probability of choosing a value from 1 to $n$ in the interval $\left[\frac{n}{4}, \frac{3n}{4}\right]$ if all values are equally likely? \(\frac{1}{2}\)
If you did choose a value between $n/4$ and $3n/4$ then the sizes of the subproblems would both be $\leq \frac{3n}{4}$.

Otherwise, the subproblems would be $\leq n$.

So we can compute an upper bound on the expected runtime.

\[
ET(n) \leq P\left(i \in \left[\frac{n}{4}, \frac{3n}{4}\right]\right) \cdot ET\left(\frac{3n}{4}\right) + P\left(i \notin \left[\frac{n}{4}, \frac{3n}{4}\right]\right) \cdot ET(n) + O(n)
\]
Expected runtime

Plug into the master theorem with $a = 1, b = 4/3, d = 1$

$a < b^d$ so

$ET(n) \leq O(n)$
Selection Problem: WHEN

**Situation so far:**

Sort then search takes worst-case $\Theta(n \log n)$

Randomized selection takes worst-case expected time $\Theta(n)$
Selection Problem: WHEN

**Situation so far:**

Sort then search takes worst-case $\Theta(n \log n)$

Randomized selection takes worst-case expected time $\Theta(n)$

*How do we implement randomized algorithms? Are there deterministic algorithms that perform as well?*

For selection problem: Blum et al., yes!

In general: open 😊