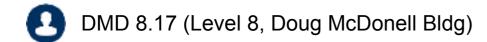


COMP90038 Algorithms and Complexity

Lecture 10: Detrosperande Conquer-by-a-Factor (with thanks to Harald Søndergaard)

Toby Murray







🦅 @tobycmurray

Decrease-and-Conquer



- Last lecture: to solve a problem of size n, try to express the solution in terms of a solution to the same problem of size n-1.
- A simple example was sorting: To sort an array of length n, just:
 - 1. sort the first n 1 items, then Add WeChat powcoder
 - 2. locate the cell A[j] that should hold the last item, right-shift all elements to its right, then place the last element in A[j].
- This led to an O(n²) algorithm called insertion sort. We can implement the idea either with recursion or iteration (we chose iteration).

Decrease-and-Conquer by-a-Factor



 We now look at better utilization of the approach, often leading to methods with logarithmic time behaviour or better!

Assignment Project Exam Help

- Decrease-by-a-constant-factor is exemplified by binary search.
 Add WeChat powcoder
- Decrease-by-a-variable-factor is exemplified by interpolation search.
- Let us look at these and other instances.

Binary Search



- This is a well-known approach for searching for an element k
 in a sorted array.
- Start by comparing against the array's middle element A[m]. If A[m] = k we are done.

Assignment Project Exam Help

- If A[m] > k, search the sub-array up to A[m 1] recursively.
- If A[m] < k, search the duberhay from Parm + 1] recursively.

$$= k?$$

$$\downarrow A[0] \cdots A[m-1] \qquad A[m] \qquad A[m+1] \cdots A[n-1]$$
search here if $A[m] > k$

Binary Search



 We have already seen a recursive formulation in Lecture 4. Here is an iterative one.

```
function BINSEARCH(A[\cdot], n, k)
    lo \leftarrow 0
    hi \leftarrow n-1 Assignment Project Exam Help
    while lo \leq hi do
                            https://powcoder.com
         m \leftarrow \lfloor (lo + hi)/2 \rfloor Add WeChat powcoder if A[m] = k then
             return m
         if A[m] > k then
             hi \leftarrow m-1
         else
             lo \leftarrow m+1
    return -1
```

Binary Search in Sorted Array function BINSEARCH($A[\cdot], n, k$)



$$hi \leftarrow n-1$$

while
$$lo \leq hi$$
 do

$$m \leftarrow \lfloor (lo + hi)/2 \rfloor$$

if A[m] = Astherment Project Exam Help

m: 3

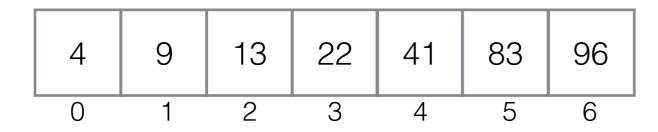
if
$$A[m] > k$$
 then
$$hi \leftarrow m - Add WeChat powcoder$$

else

$$lo \leftarrow m+1$$

return -1

A:



Binary Search in Sorted Array



function BINSEARCH($A[\cdot], n, k$)

$$lo \leftarrow 0$$
 $hi \leftarrow n-1$
while $lo \leq hi$ do
 $m \leftarrow \lfloor (lo + hi)/2 \rfloor$
if $A[m] = Astigement Project Exam Help$
return m
https://powcoder.com
if $A[m] > k$ then
 $hi \leftarrow m - Add$ WeChat powcoder
else
 $lo \leftarrow m+1$
return -1

k: 41

lo: 4

hi: 6

m: 3

A:



Binary Search in Sorted Array function BINSEARCH($A[\cdot], n, k$)



```
lo \leftarrow 0

hi \leftarrow n-1

while lo \le hi do lo: 4

m \leftarrow \lfloor (lo + hi)/2 \rfloor

if A[m] = Astlæment Project Exam Help

return m https://powcoder.com

if A[m] > k then

hi \leftarrow m - 1 Add WeChat powcoder
```

else

$$lo \leftarrow m+1$$

return -1

A:







```
lo \leftarrow 0
hi \leftarrow n-1
while lo \leq hi do
    m \leftarrow |(lo + hi)/2|
    if A[m] = Astherment Project Exam Help
         return m https://powcoder.com
    if A[m] > k then
hi \leftarrow m - Add \text{ WeChat powcoder}
    else
         lo \leftarrow m+1
```

k: 41

lo: 4

hi: 4

m: 5

return -1

A:



function BINSEARCH($A[\cdot], n, k$)





$lo \leftarrow 0$	lz. 1
$hi \leftarrow n-1$	k: 4
while $lo \leq hi$ do	lo: 4
$m \leftarrow \lfloor (lo + hi)/2 \rfloor$	
if $A[m] = Astherment Project Exam Help$	hi: 4
return m https://powcoder.com	m: 4
:C A[] > /. +la	

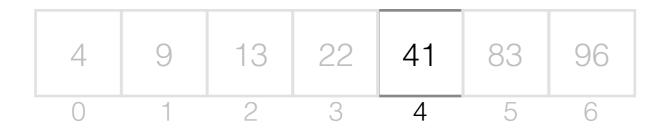
if A[m] > k then $hi \leftarrow m - Add WeChat powcoder$

 $lo \leftarrow m+1$

return -1

else

A:



Complexity of Binary Search MELBOURNE

- Worst-case input to binary sarch:
 - When k is not in the array
- In that case, its complexity is given by the following

recursive equations ignment Project Exam Help
$$C(n)$$
 if $n=1$ $C(n)$ if $n>1$ Add WeChat powcoder $C(n)$ if $n>1$

- A closed form is: $C(n) = \lfloor \log_2 n \rfloor + 1$
- In the worst case, searching for k in an array of size 1,000,000 requires 20 comparisons.
- The average-case time complexity is also Θ(log n)

Russian Peasant Multiplication



- A way of doing multiplication.
- For even *n*:

Add WeChat powcoder

• For odd *n*:

$$n \cdot m = \frac{n-1}{2} \cdot 2m + m$$

Thus, ~halve n repeatedly, until
 n = 1. Add up all odd values of m

	n	m	
	81	92	92
p	40	184	
	20	368	
	10	736	
	5	1472	1472
	2	2944	
	1	5888	5888
		=	7452

Finding the Median



 Given an array, an important problem is how to find the median, that is, an array value which is no larger than half the elements and no smaller than half.



• More generally, we would like to solve the problem of finding the kth smallest element. (e.g. when the problem of smallest element)



• If the array is sorted, the solution is straight-forward, so one approach is to start by sorting (as we'll soon see, this can be done in time O (n log n)).



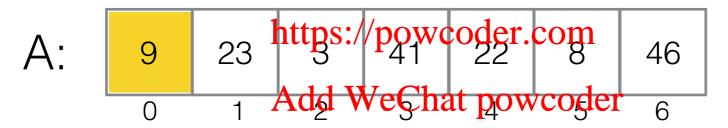
However, sorting the array seems like overkill.

A Detour via Partitioning

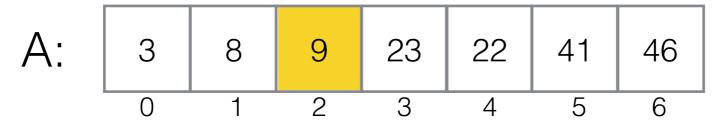


 Partitioning an array around some pivot element p means reorganizing the array so that all elements to the left of p are no greater than p, while those to the right are no smaller.

Assignment Project Exam Help



Partitioning around the pivot 9





$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo + Assistant Help | Assistant H$

lo	5		i	hi
р	< <i>p</i>	≥ <i>p</i>		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo$ | Assistant Help | 3 | 41 | 22 | 8 | 46

if $A[i] < p$ then poweder som | 2 | 3 | 4 | 5 | 6

 $s \leftarrow s + Ald$ | We Chat poweder | i

 $swap(A[s], A[i])$
 $swap(A[lo], A[s])$

return s

lo	5		i	hi
р	< <i>p</i>	≥ <i>p</i>		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo + Asign + Pi$ | Coject | Exam Help | 3 | 41 | 22 | 8 | 46 |
if $A[i] < p$ | the proposed error | 2 | 3 | 4 | 5 | 6 |
 $s \leftarrow s + Ald \text{ WeChat powcodes}$ | i
 $swap(A[s], A[i])$
 $swap(A[lo], A[s])$
return s

lo	5		i	hi
р	< <i>p</i>	≥ <i>p</i>		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo +A$ is confine Reject Exam Help | 23 | 41 | 22 | 8 | 46

if $A[i] < p$ then powcoder som | 2 | 3 | 4 | 5 | 6

 $s \leftarrow s + A$ id WeChat powcodes | i

 $swap(A[s], A[i])$
 $swap(A[lo], A[s])$

return s

lo	5		i	hi
р	< <i>p</i>	≥ <i>p</i>		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$
 $| lo$
 $| hi$
 $| for i \leftarrow lo + Asign for Project Exam Help | 23 | 41 | 22 | 8 | 46 |$
 $| if A[i]
 $| s \leftarrow s + Ald We Chat powcoder | i$
 $| swap(A[s], A[i])$
 $| swap(A[lo], A[s])$
 $| return s | swap(A[lo], A[s])$$

$$\begin{array}{|c|c|c|c|c|c|} \hline b & s & i & hi \\ \hline p &$$



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo + Assistant Placet Exam Help | 23 | 41 | 22 | 8 | 46$

if $A[i] < p$ then powcoder som | 2 | 3 | 4 | 5 | 6

 $s \leftarrow s + Ald We Chat powcodes$ | i

 $swap(A[s], A[i])$
 $swap(A[lo], A[s])$

return s

lo	5		i	hi
р	< <i>p</i>	$\geq p$		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$
 $s \leftarrow lo$

lo	5		i	hi
р	< p	$\geq p$		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$
 $s \leftarrow lo$

lo	5		i	hi
p	< p	$\geq p$		



$$p \leftarrow A[lo]$$
 $s \leftarrow lo$ | O | hi

for $i \leftarrow lo + Assistant Placet Exam Help | 8 | 41 | 22 | 23 | 46$

if $A[i] < p$ then powcoder som | 2 | 3 | 4 | 5 | 6

 $s \leftarrow s + Ald WeChat powcoder$ | S | i

 $swap(A[s], A[i])$
 $swap(A[lo], A[s])$

return s

lo	5		i	hi
р	< <i>p</i>	≥ <i>p</i>		

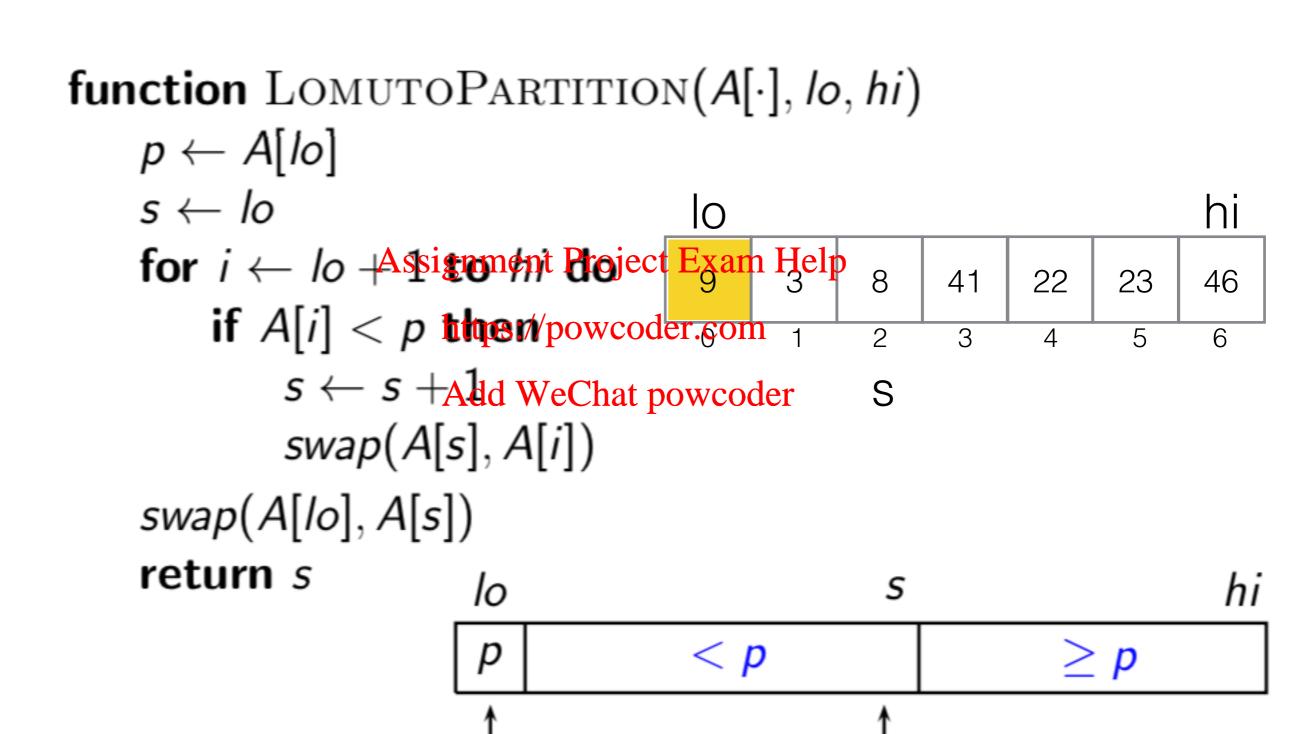


$$p \leftarrow A[lo]$$
 $s \leftarrow lo$
 $| o \rangle$
for $i \leftarrow lo + Asign + Point Project Exam Help | 8 | 41 | 22 | 23 | 46$
if $A[i] < p$ then powcoder som | 2 | 3 | 4 | 5 | 6
$$s \leftarrow s + A \text{ lid We Chat powcoder } S \qquad i$$

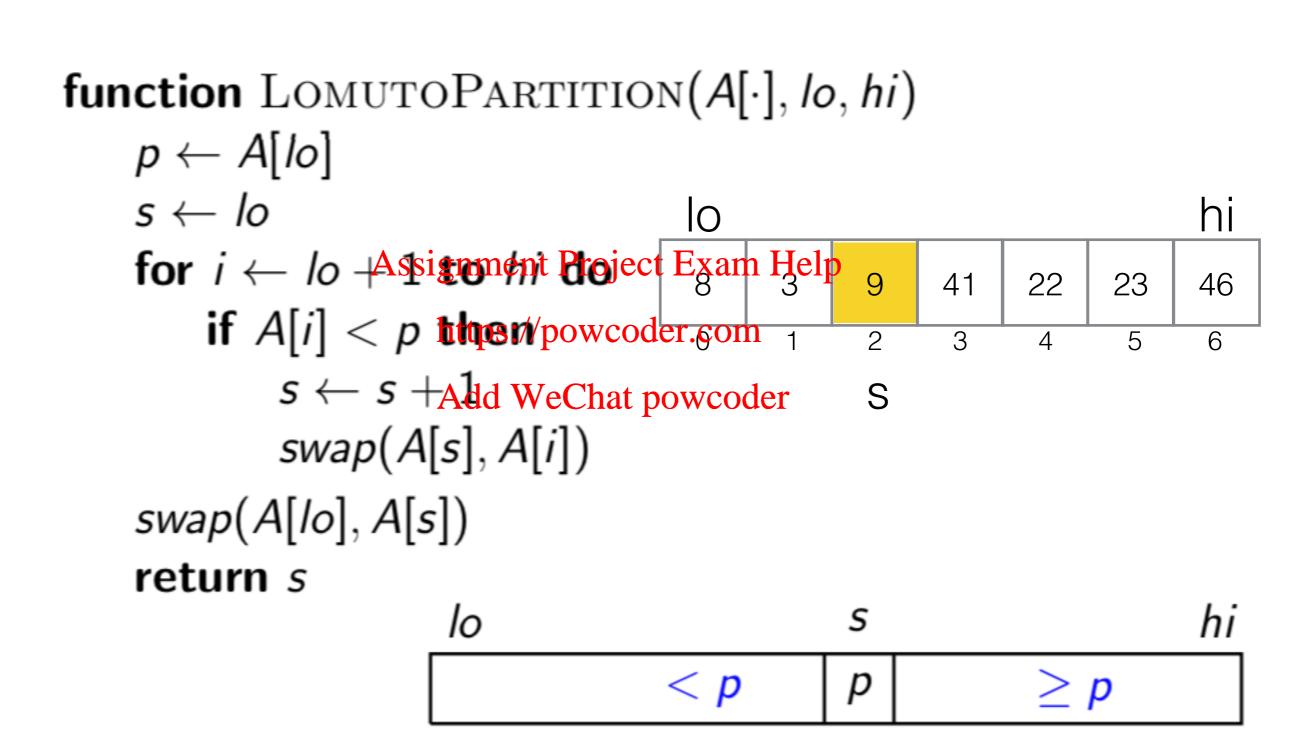
$$swap(A[s], A[i])$$
 $swap(A[lo], A[s])$
return s

lo	5		i	hi
р	< p	$\geq p$		









Finding the *k*th-smallest Element



```
function QuickSelect(A[\cdot], lo, hi, k)
   s \leftarrow \text{LomutoPartition}(A, lo, hi)
   if s - lo = k - 1 then
       return A[s]
Assignment Project Exam Help
   else
       if s - lo > k https://powcoder.com
           QUICKSELEd We hat powcoder k)
       else
           QuickSelect(A, s + 1, hi, (k - 1) - (s - lo))
                 10
                               41
                                    22
                                         23
                                              46
```



```
function QuickSelect(A[\cdot], lo, hi, k)

s \leftarrow \text{LomutoPartition}(A, lo, hi)

if s - lo = k - 1 then

return A[s]

else

Assignment Project Exam Help

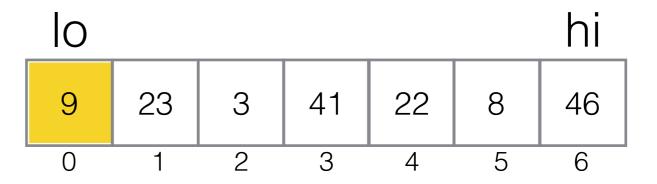
if s - lo > k - 1 then

QuickSelect(A, b, b, s = 1, k)

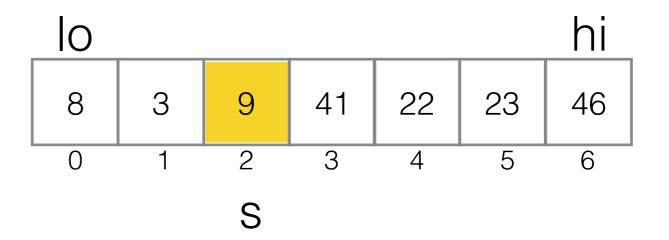
else

Add WeChat powcoder

QuickSelect(A, s + 1, hi, (k - 1) - (s - lo))
```









			lo			<u>hi</u>
8	3	9	41	22	23	46
0	1	2	3	4	5	6

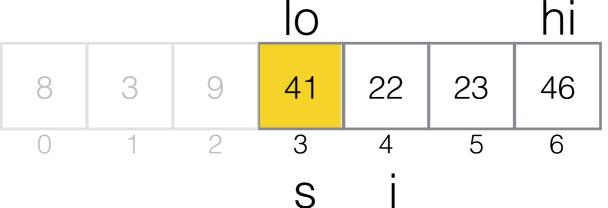


$$p \leftarrow A[lo]$$
 $s \leftarrow lo$

for $i \leftarrow lo + 1$ to hi do

if $A[i] < p$ simple Project Exam Help
 $s \leftarrow s + l$ ttps://powcoder.com
 $swap(A[s] dA[v] = code$ Chat powcoder

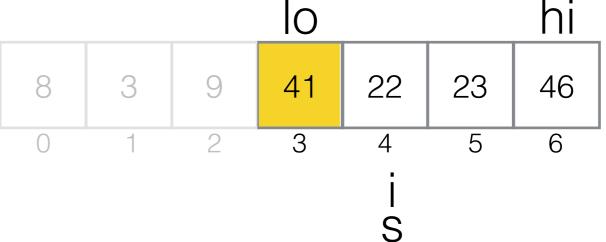
 $swap(A[lo], A[s])$
return s





$$p \leftarrow A[lo]$$

 $s \leftarrow lo$
for $i \leftarrow lo + 1$ to hi do
if $A[i] < p$ streent Project Exam Help
 $s \leftarrow s + 1$ ttps://powcoder.com
 $swap(A[s] A[v])$ Chat powcoder
 $swap(A[lo], A[s])$
return s

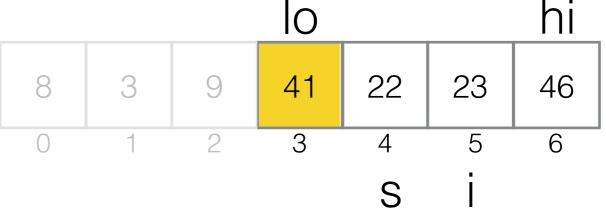




$$p \leftarrow A[lo]$$
 $s \leftarrow lo$

for $i \leftarrow lo + 1$ to hi do

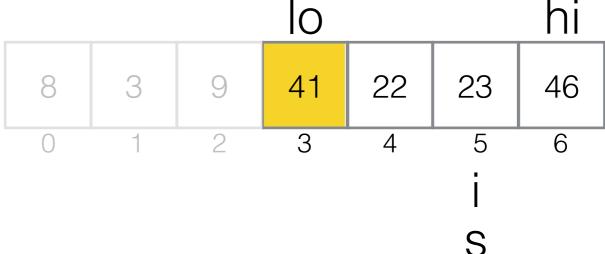
if $A[i] < \frac{1}{p}$ streent Project Exam Help
 $s \leftarrow s + \frac{1}{p}$ ttps://powcoder.com
 $swap(A[s] dA[v] = Chat powcoder$
 $swap(A[lo], A[s])$
return s





$$p \leftarrow A[lo]$$

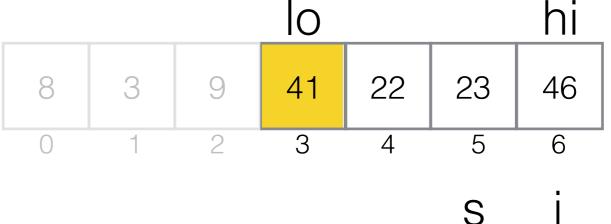
 $s \leftarrow lo$
for $i \leftarrow lo + 1$ to hi do
if $A[i] < p$ streen Project Exam Help
 $s \leftarrow s + \frac{1}{2}$ ttps://powcoder.com
 $swap(A[s] A[v] = Chat powcoder$
 $swap(A[lo], A[s])$
return s





$$p \leftarrow A[lo]$$

 $s \leftarrow lo$
for $i \leftarrow lo + 1$ to hi do
if $A[i] < p$ streen Project Exam Help
 $s \leftarrow s + l$ ttps://powcoder.com
 $swap(A[s] dA[v] e$ Chat powcoder
 $swap(A[lo], A[s])$
return s



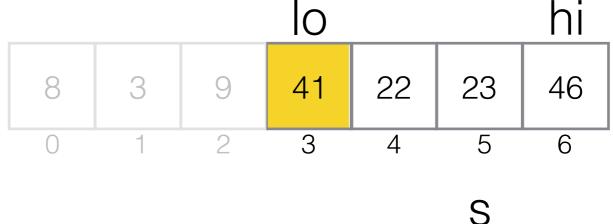


$$p \leftarrow A[lo]$$
 $s \leftarrow lo$

for $i \leftarrow lo + 1$ to hi do

if $A[i] < p$ simple Project Exam Help
 $s \leftarrow s + l$ ttps://powcoder.com
 $swap(A[s] dA[v] = code$ Chat powcoder

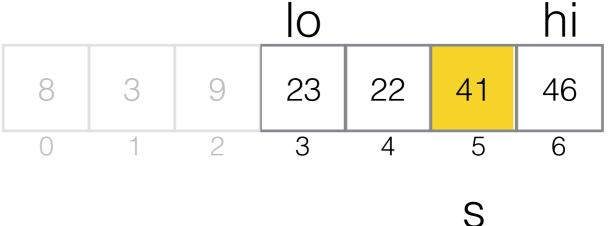
 $swap(A[lo], A[s])$
return s



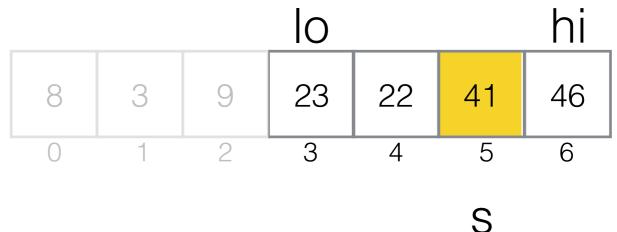


$$p \leftarrow A[lo]$$

 $s \leftarrow lo$
for $i \leftarrow lo + 1$ to hi do
if $A[i] < p$ streent Project Exam Help
 $s \leftarrow s + 1$ ttps://powcoder.com
 $swap(A[s] A[v])$ Chat powcoder
 $swap(A[lo], A[s])$
return s









```
function QuickSelect(A[\cdot], lo, hi, k)

s \leftarrow \text{LomutoPartition}(A, lo, hi)

if s - lo = k - 1 then

return A[s]

else

Assignment Project Exam Help

if s - lo > k - 1 then

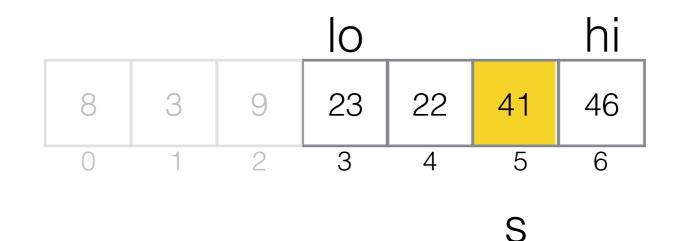
QuickSelect(A, B, S = 1, B)

else

Add WeChat powcoder

QuickSelect(A, S + 1, S
```

returns 41!



QuickSelect Complexity



• Worst case complexity for Quick Select is quadratic, https://powcoder.com

Add WeChat powcoder

Average-case complexity is linear.

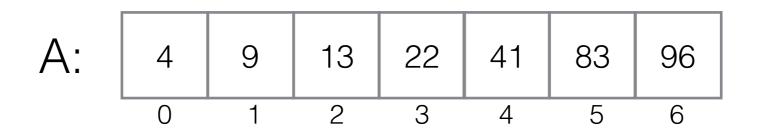
Interpolation Search



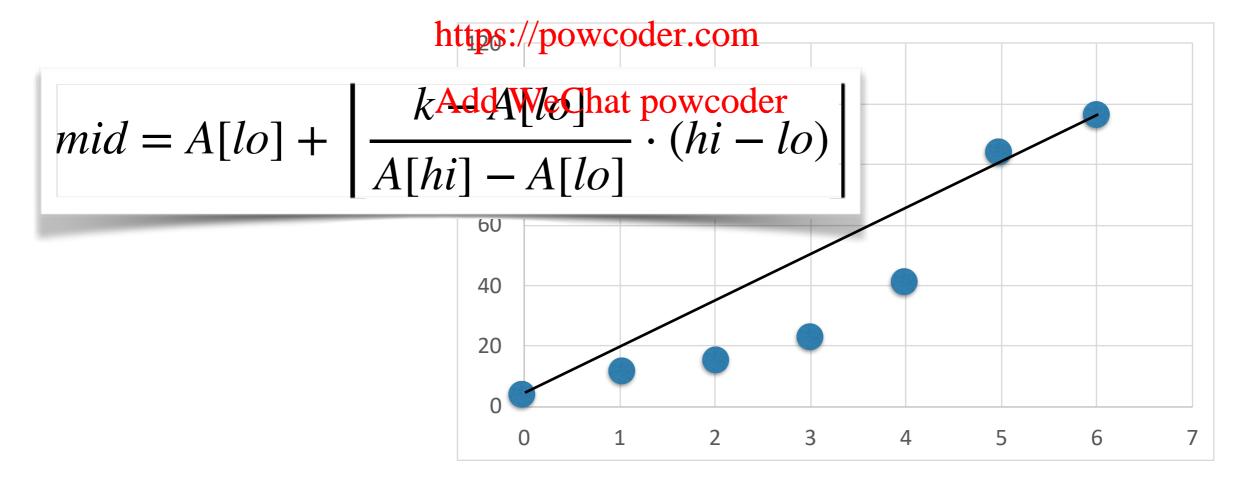
- If the elements of a sorted array are distributed reasonably evenly, we can do better than binary search!
- Think about how you search for an entry in the Assignment Project Exam Help telephone directory. If you look for 'Zobel', you make a rough estimate of white the the opposed the first probe—very close to the end of the directory wooder
- This is the idea in interpolation search.
- When searching for k in the array segment A[lo] to A[hi], take into account where k is, relative to A[lo] and A[hi].

Interpolation Search





Suppose is mare rejeant sain the for k = 83



Interpolation Search



 Instead of computing the mid-point m as in binary search:

$$m \leftarrow \lfloor (lo + hi)/2 \rfloor$$

we instead perform linear interpolation between the Assignment Project Exam Help points (lo,A[lo]) and (hi,A[hi]). That is, we use:

https://powcoder.com

$$m \leftarrow lo + \frac{\text{Add WkChatAplowcoder}}{A[hi] - A[lo]} (hi - lo)$$

- Interpolation search has average complexity O(log log n)
- It is the right choice for large arrays when elements are uniformly distributed

Next Week



Assignment Project Exam Help

- Learn to divide and conquer!
- Read Levitin Chapter 5, but skip 5.4.