Lecture 4

Divide and Conquer III: quicksort, quickselect, median, integer multiplication

CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Divide and Conquer (recap)

Steps of method:

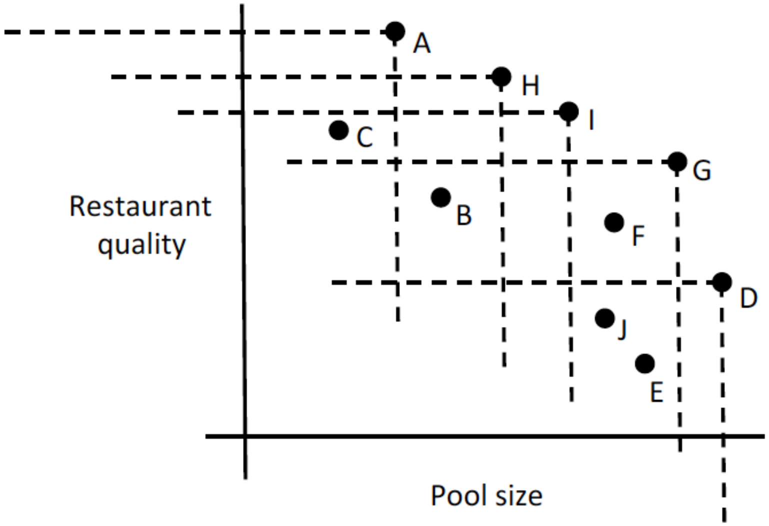
- **Divide** input into parts (smaller problems)
- **Conquer** (solve) each part recursively
- **Combine** results to obtain solution of original

$$T(n) = \text{divide time} + T(n_1) + T(n_2) + \ldots + T(n_k) + \text{combine time}$$

Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \dots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Example: x captures pool size and y restaurant quality. 10 hotels



Case study IV: Maxima Set

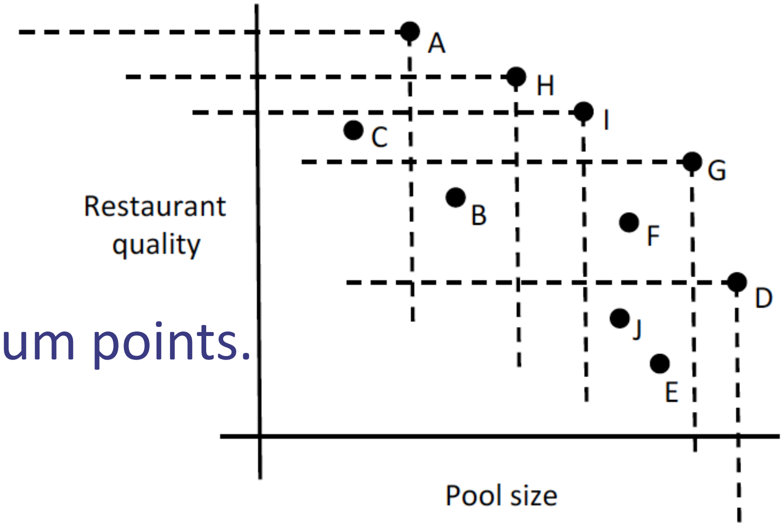
Problem: We are given n points $(x_1, y_1), \dots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Example: x captures pool size and y restaurant quality. 10 hotels

Explanation:

A, H, I, G, D are maximum points.

C, B, F, J, E are not.



Case study IV: Maxima Set

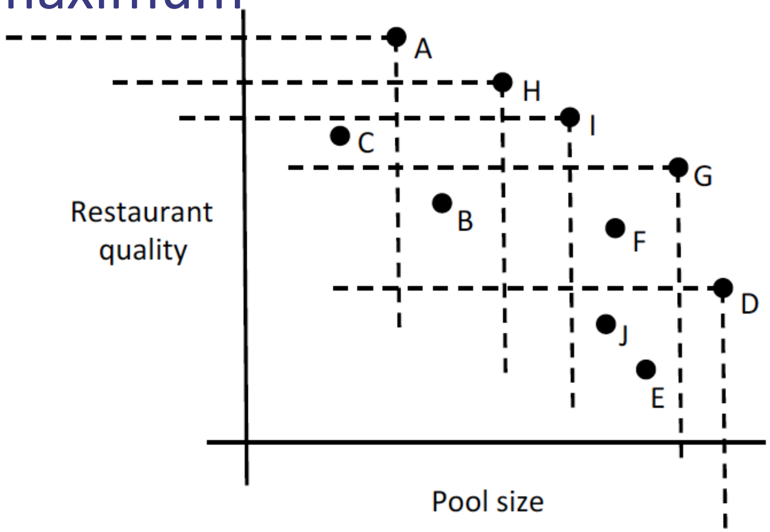
Problem: We are given n points $(x_1, y_1), \dots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Obvious approach:

For every point (x_i, y_i), check if it is maximum

To check if it is maximum, you check

the condition with all other points.



Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \ldots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Pseudocode:

$\text{counter} \leftarrow 0$
For $i = 1$ to n **do**
$\text{flag} \leftarrow 1$
    For $j = i + 1$ to n **do**
        If $(x_j > x_i$ and $y_j > y_i)$ **then** $\text{flag} \leftarrow 0$
$\text{counter} \leftarrow \text{counter} + \text{flag}$
return counter

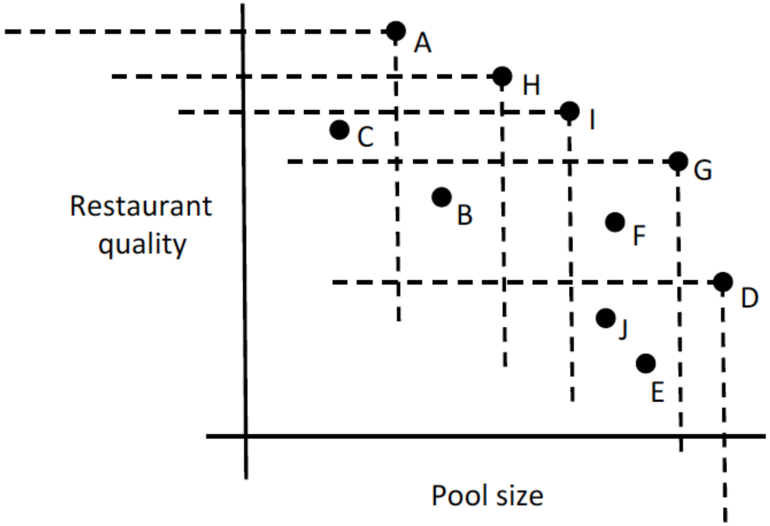
Running time $\Theta(n^2)$

Can we do better?

Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \ldots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Idea: Divide and conquer. Divide step and Combine step is challenging.



Case study IV: Maxima Set

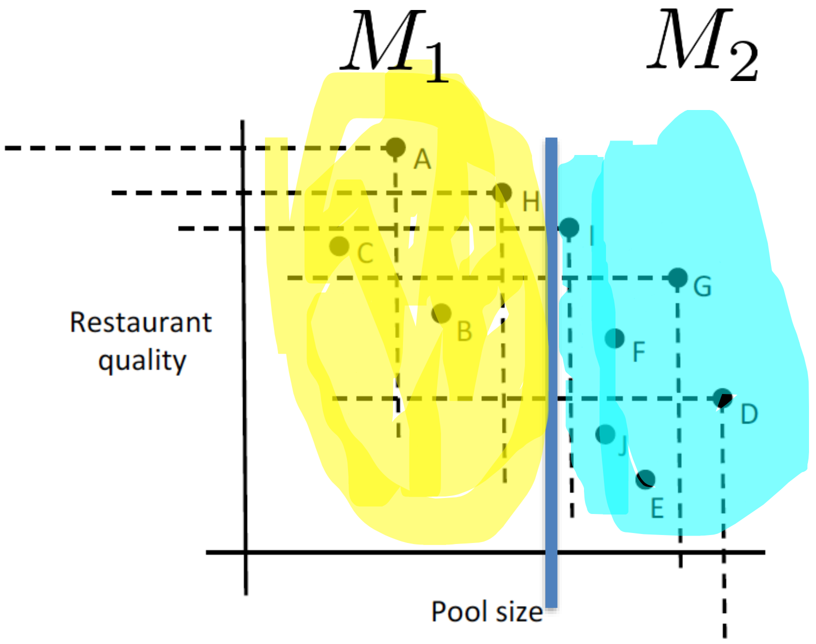
Divide step: It should split the points in two parts of equal size.

How?

Case study IV: Maxima Set

Divide step: It should split the points in two parts of equal size.

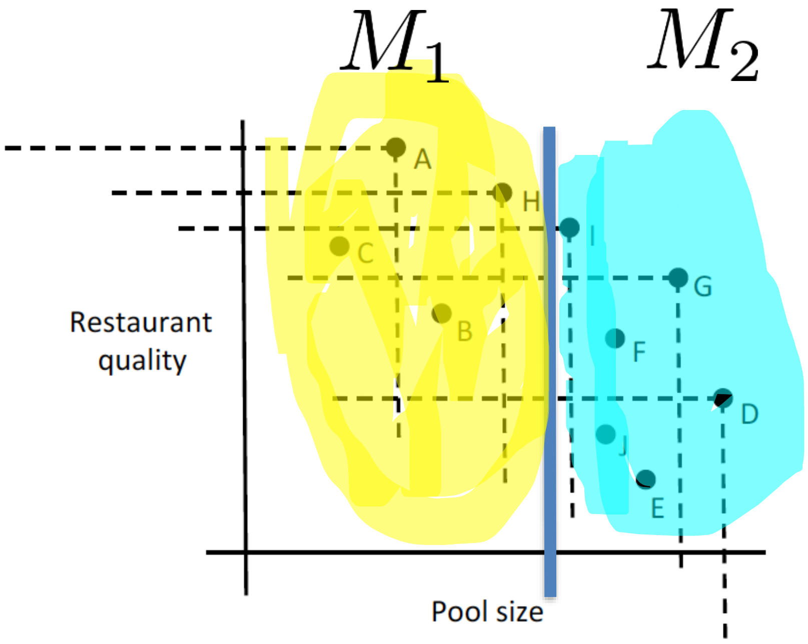
How? Choose the middle (median) point with respect to x coordinates.



Case study IV: Maxima Set

Divide step: It should split the points in two parts of equal size.

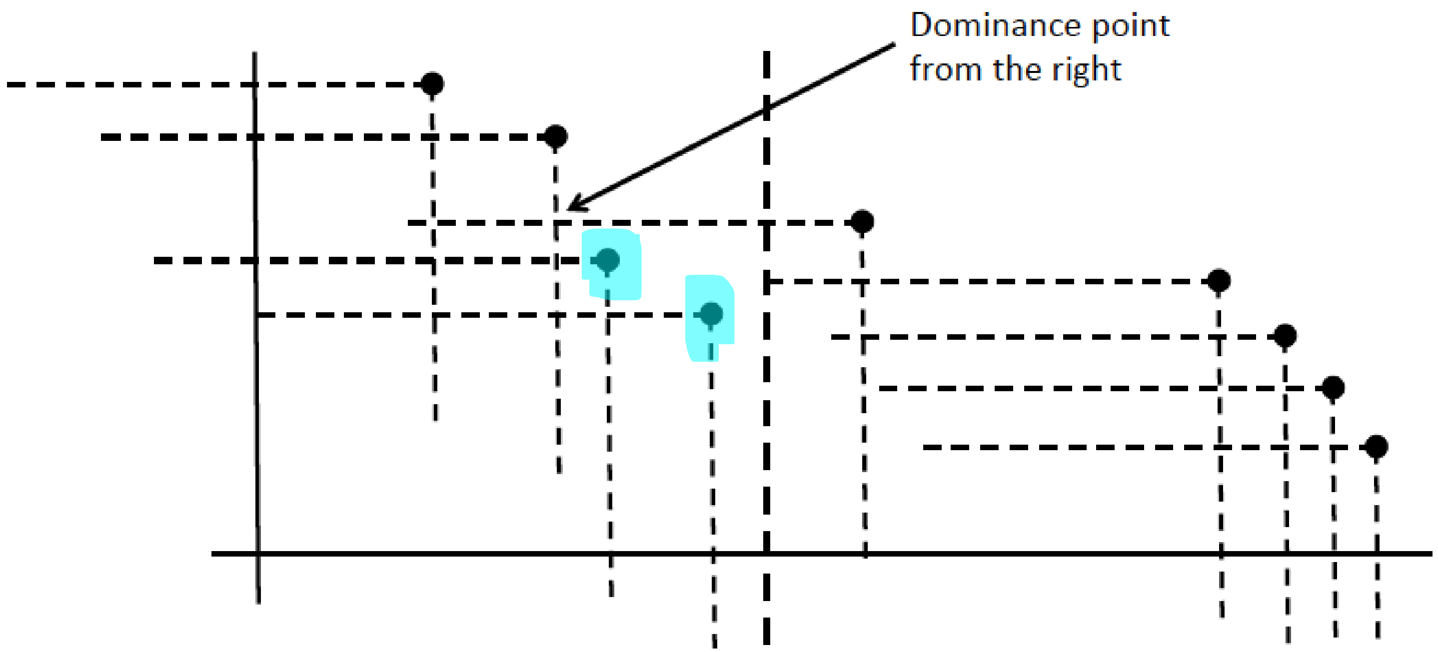
How? Choose the middle (median) point with respect to x coordinates.



Combine step: Return $M_1 \cup M_2$?

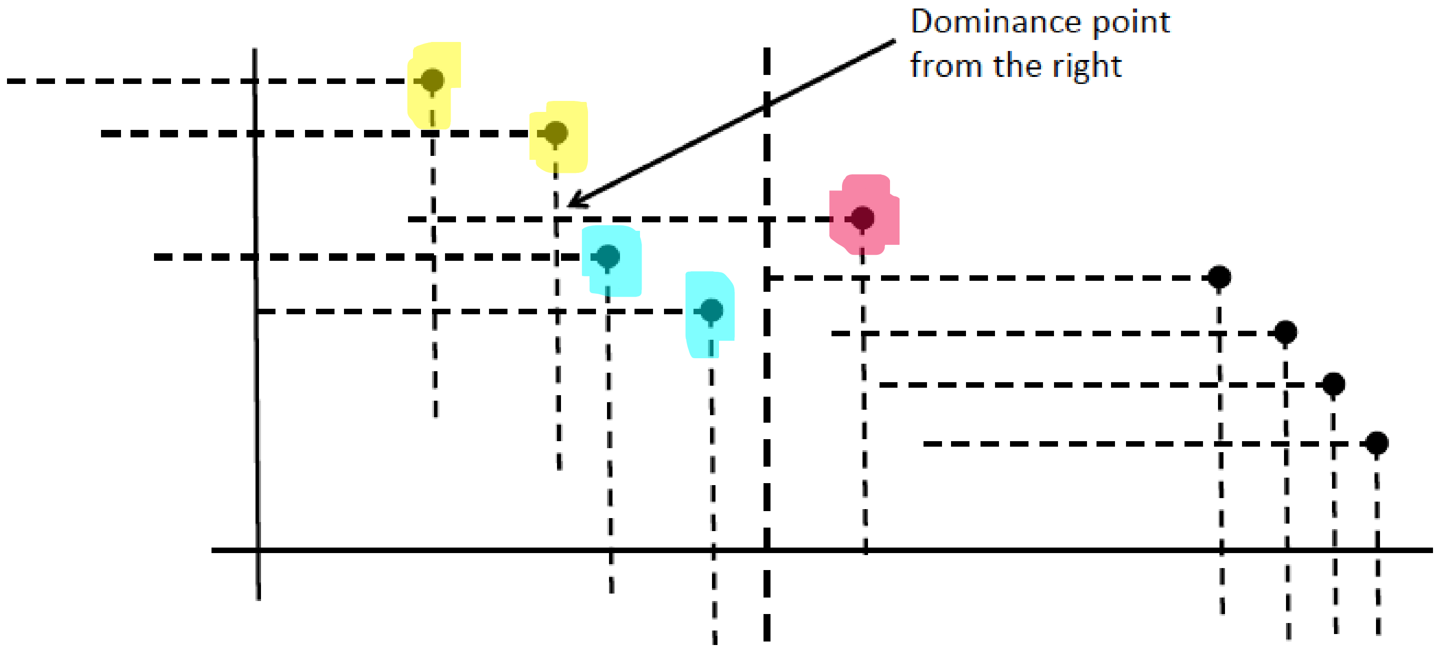
Case study IV: Maxima Set

Combine step: Return $M_1 \cup M_2$? Wrong: blue points below of M_1 are not part of the solution



Case study IV: Maxima Set

Combine step idea: M_2 points should part of the solution. From M_1, the points that are maximum should not be dominated by smallest with respect to x coordinates in M_2



Case study IV: Maxima Set

Pseudocode:

```
MaximaSet(S,n):
    if n = 1 then
        return S
    Let p be the median point in S, by x -coordinates
    Let L be the set of points less than p in S by x -coordinates
    Let G be the set of points greater than or equal to p in S by x -coordinates
    M1 ← MaximaSet(L)
    M2 ← MaximaSet(G)
    Let q be the smallest point in M2
    for each point, r, in M1 do by x -coordinates
        if x(r) ≤ x(q) and y(r) ≤ y(q) then
            Remove r from M1
    return M1 ∪ M2
```

Case study IV: Maxima Set

Pseudocode:

```
MaximaSet(S, n):
    if n = 1 then
        return S
    Let p be the median point in S, by x-coordinates
    Let L be the set of points less than p in S by x-coordinates
    Let G be the set of points greater than or equal to p in S by x-coordinates
    M_1 ← MaximaSet(L)
    M_2 ← MaximaSet(G)
    Let q be the smallest point in M_2
    for each point, r, in M_1 do by x-coordinates
        if x(r) ≤ x(q) and y(r) ≤ y(q) then
            Remove r from M_1
    return M_1 ∪ M_2
```

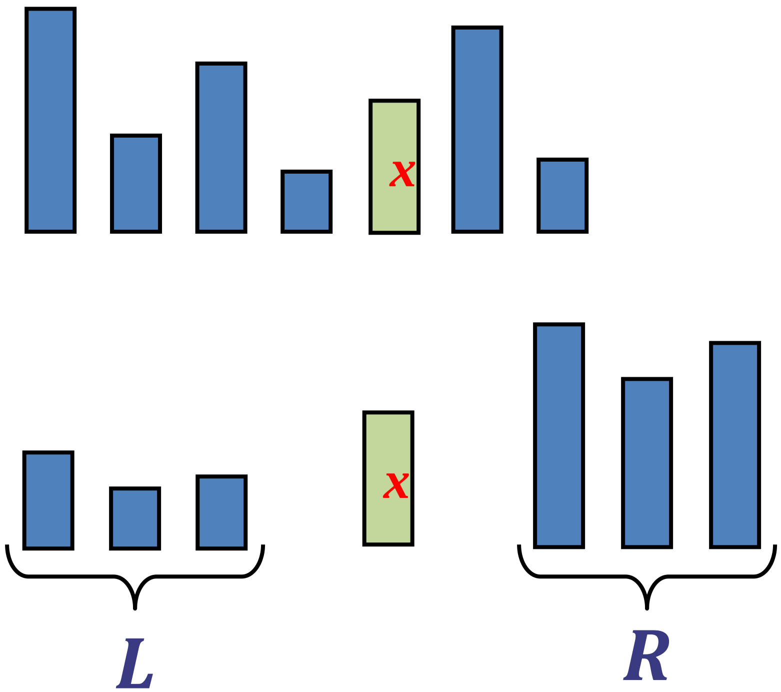
Running time??

$$\text{Running time is } T(n) = 2T(n/2) + T_{\text{media}}(n) + T_{\min}(n) + \Theta(n)$$
$$= 2T(n/2) + T_{\text{media}}(n) + \Theta(n)$$

Quicksort (recap)

Steps of Quicksort:

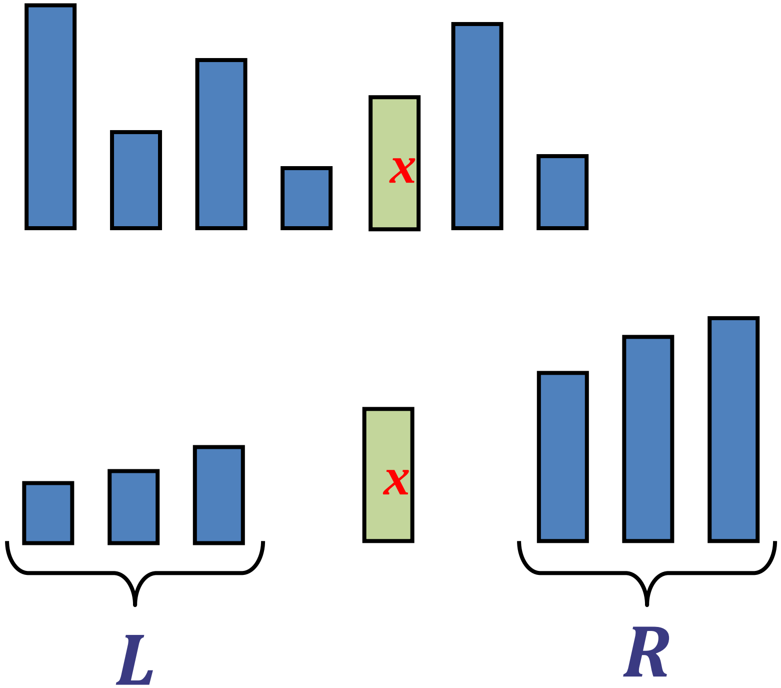
- **Divide:** pick an element x (called pivot) and partition into L, $\{x\}$ and R.



Quicksort (recap)

Steps of Quicksort:

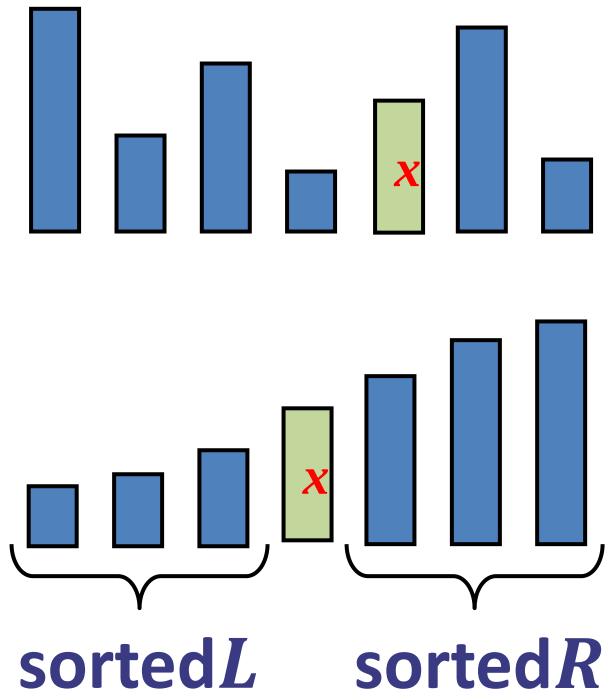
- **Divide:** pick an element x (called pivot) and partition into L, $\{x\}$ and R.
- **Conquer:** L and R are sorted recursively.



Quicksort (recap)

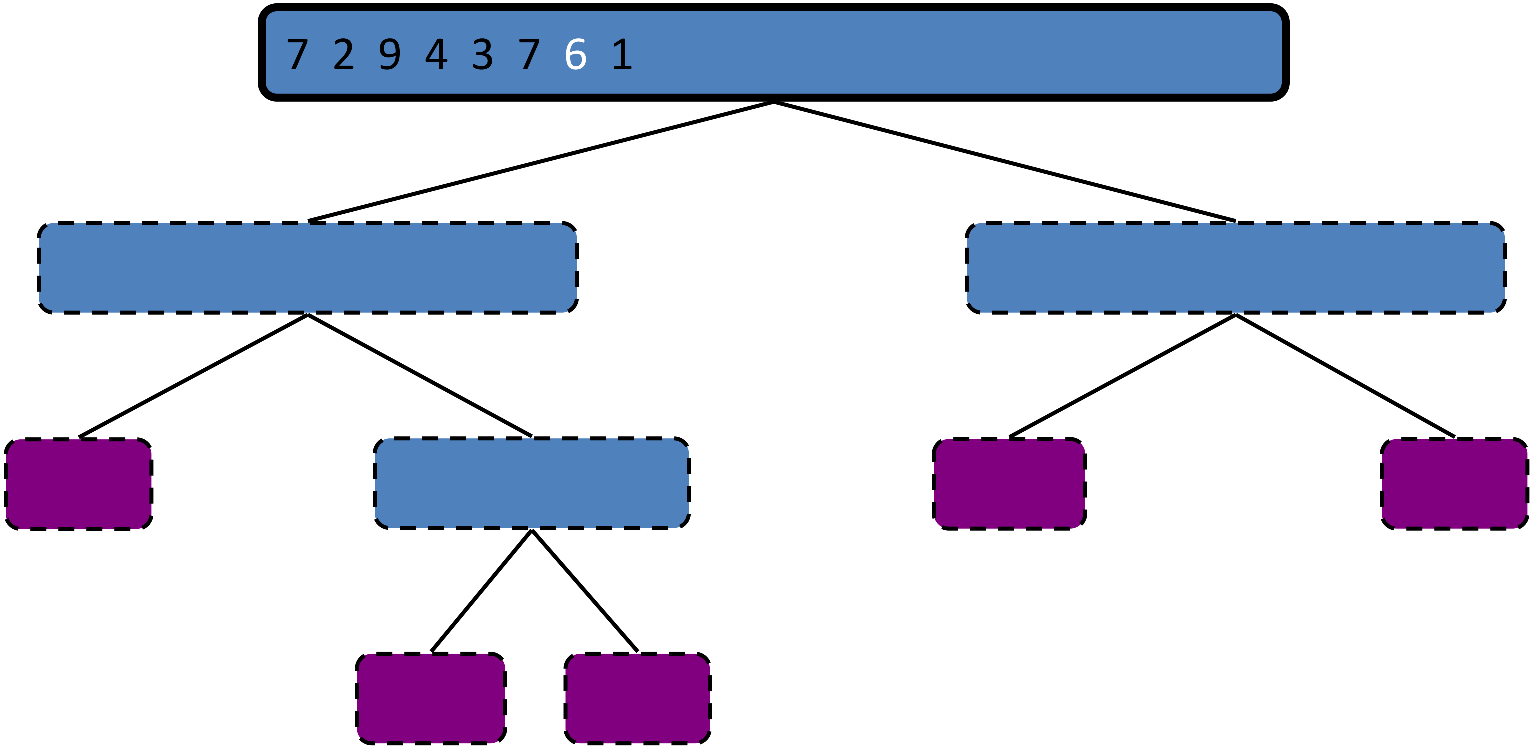
Steps of Quicksort:

- **Divide:** pick an element x (called pivot) and partition into L, $\{x\}$ and R.
- **Conquer:** L and R are sorted recursively.
- **Combine:** return **sortedL**, x, **sortedR**.



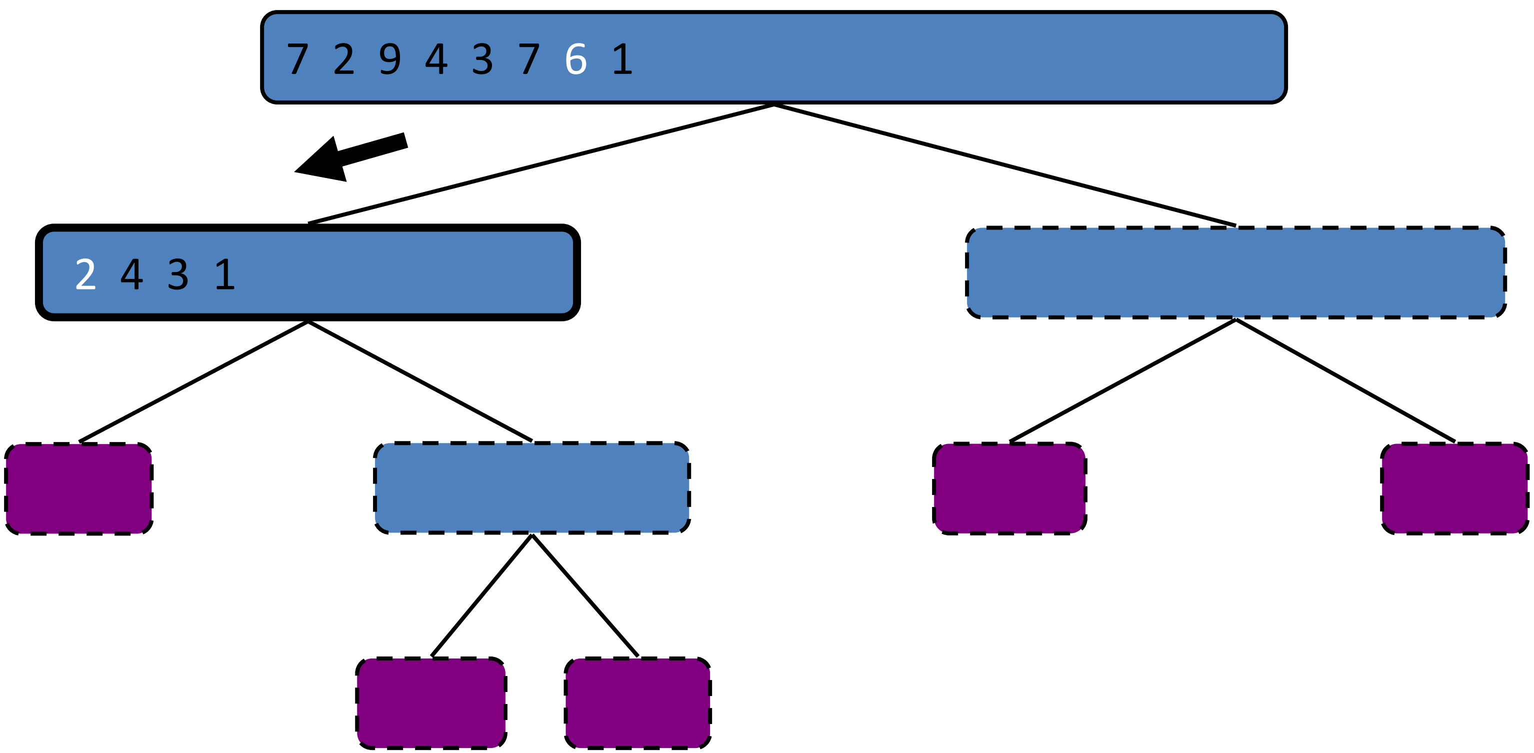
Quicksort (Example)

Pivot selection



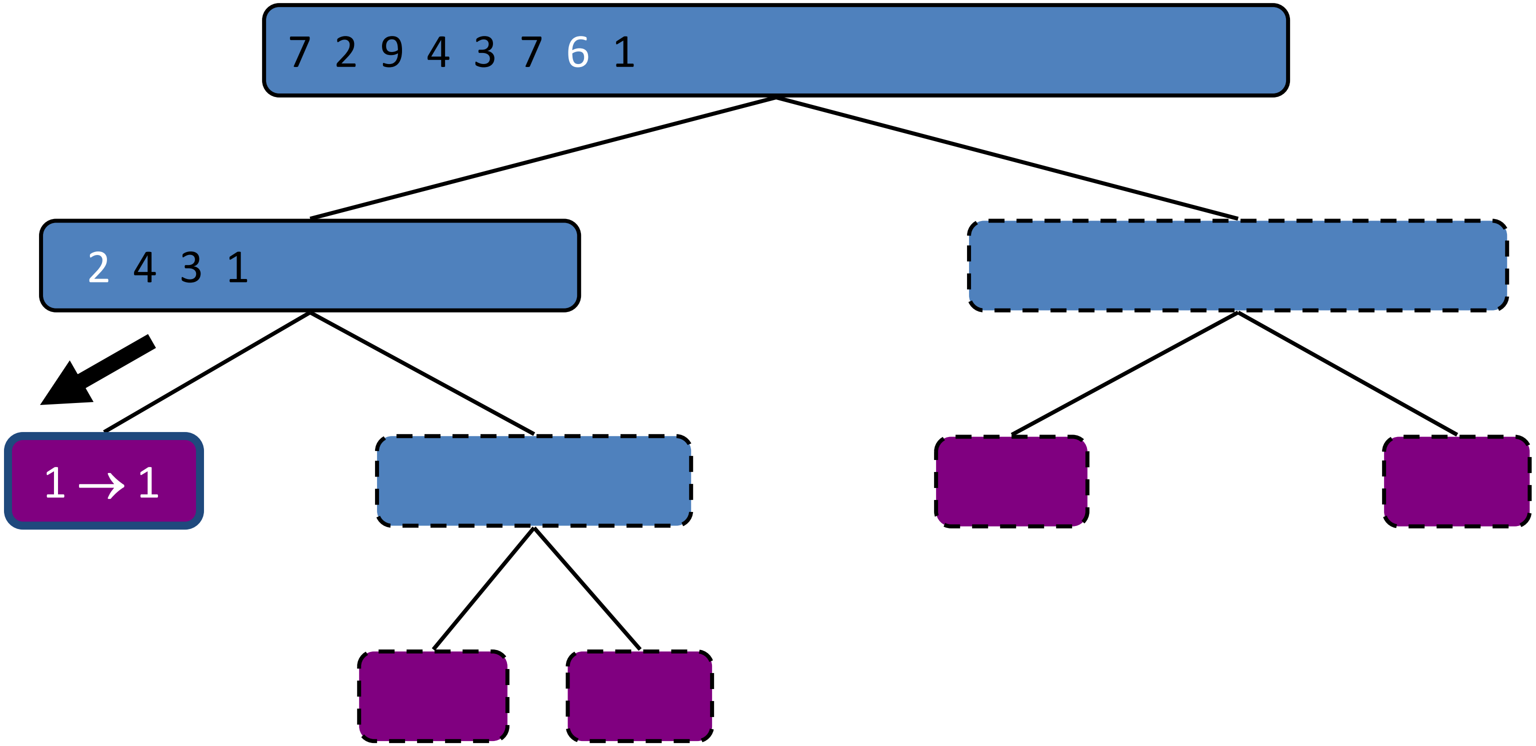
Quicksort (Example)

Partition, recursive call and pivot selection



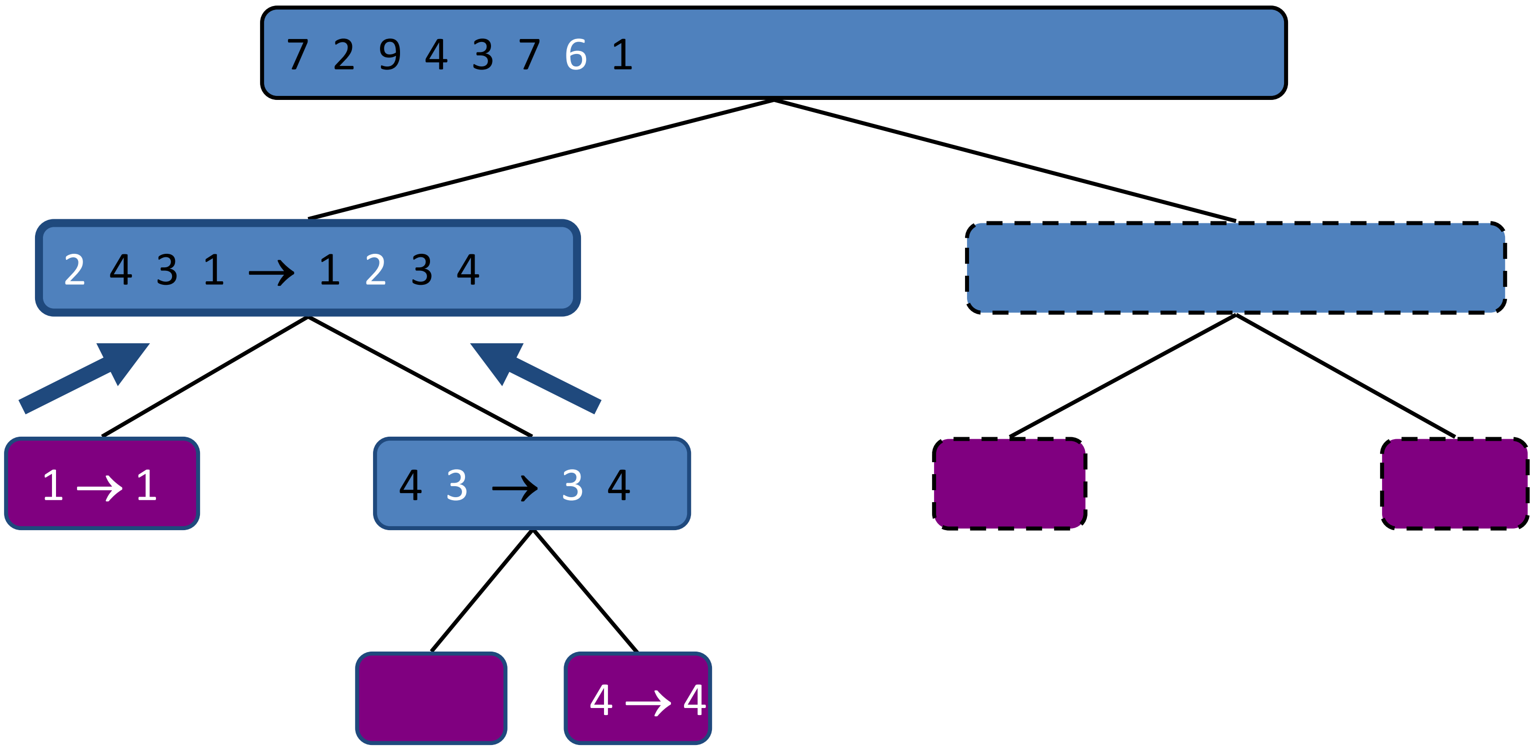
Quicksort (Example)

Partition, recursive call, base case



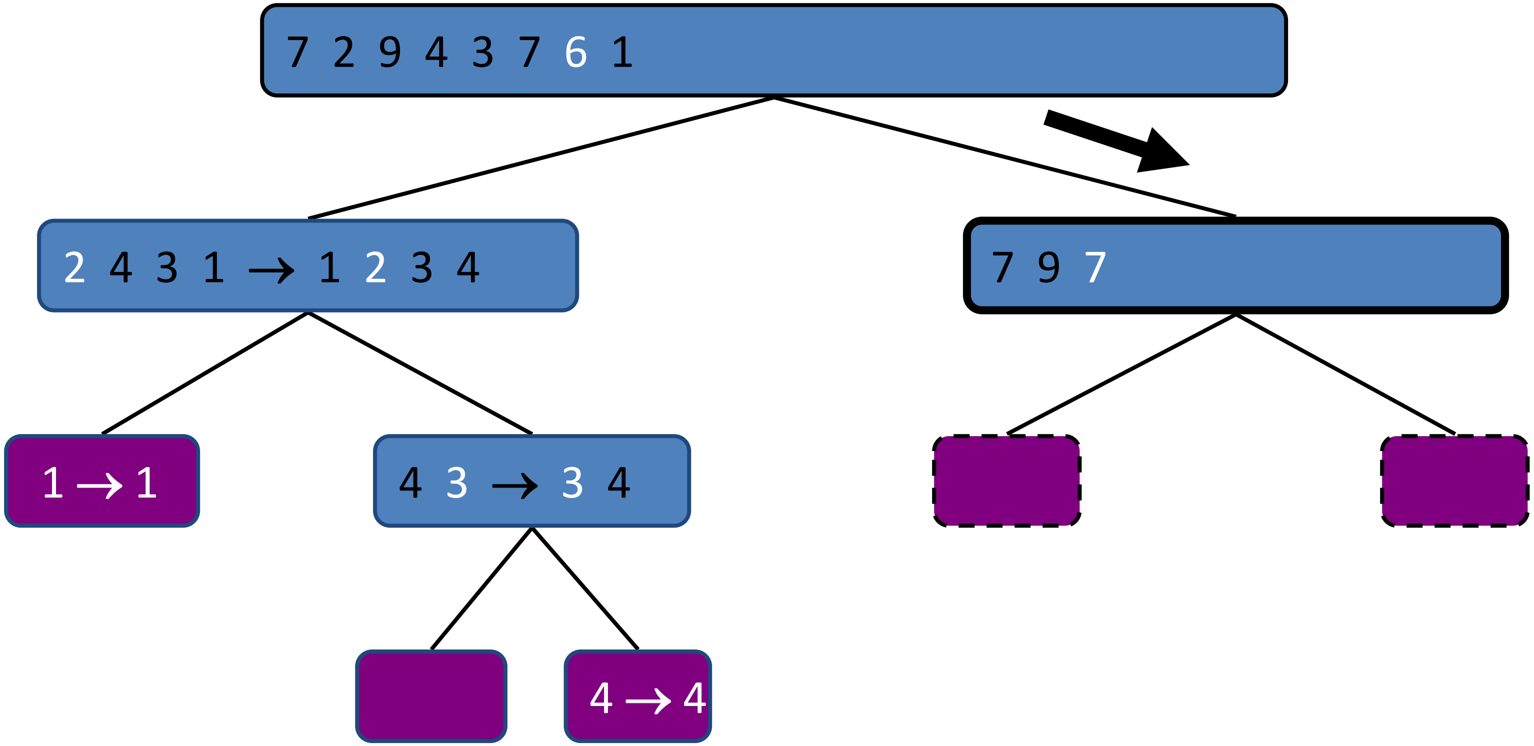
Quicksort (Example)

Recursive call, …, base case, join



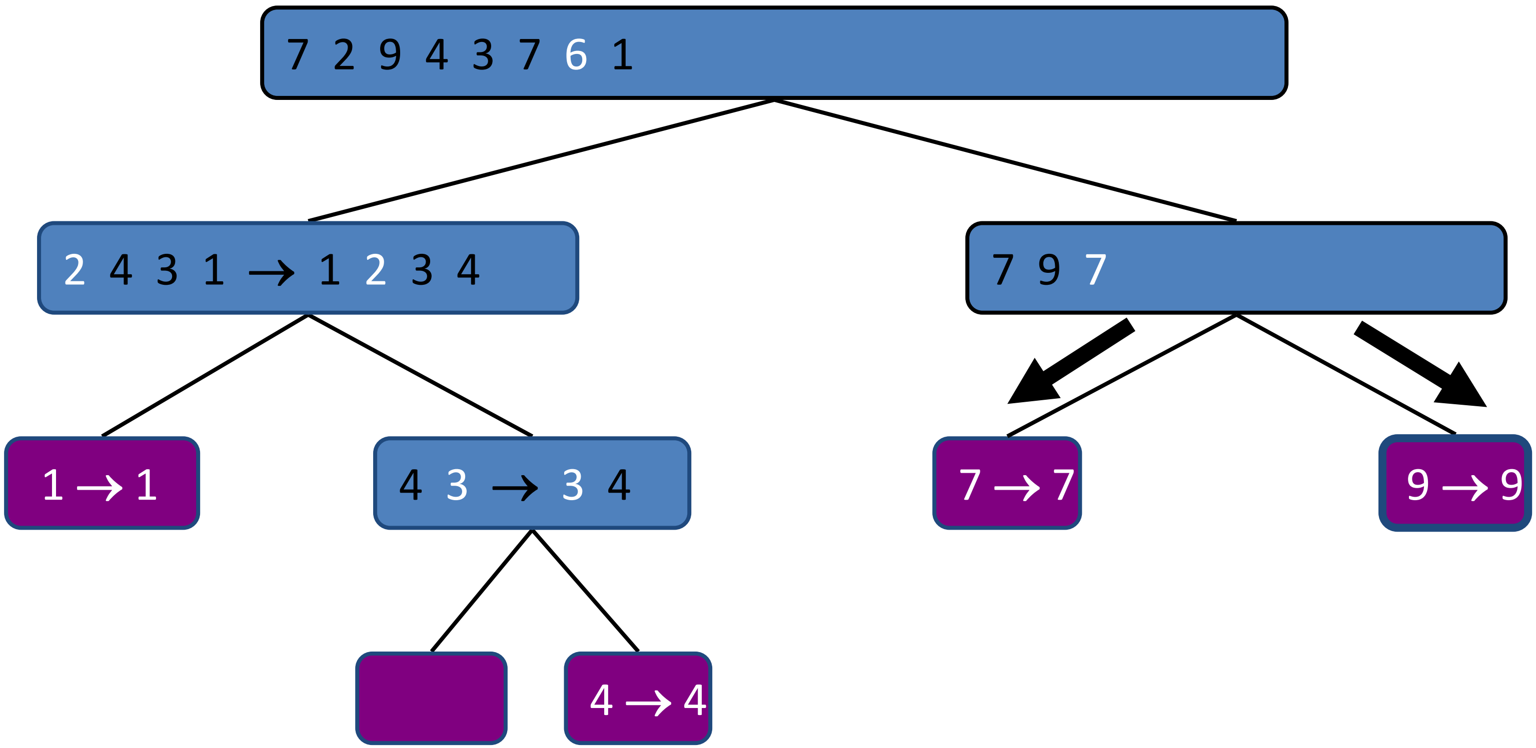
Quicksort (Example)

Recursive call, pivot selection



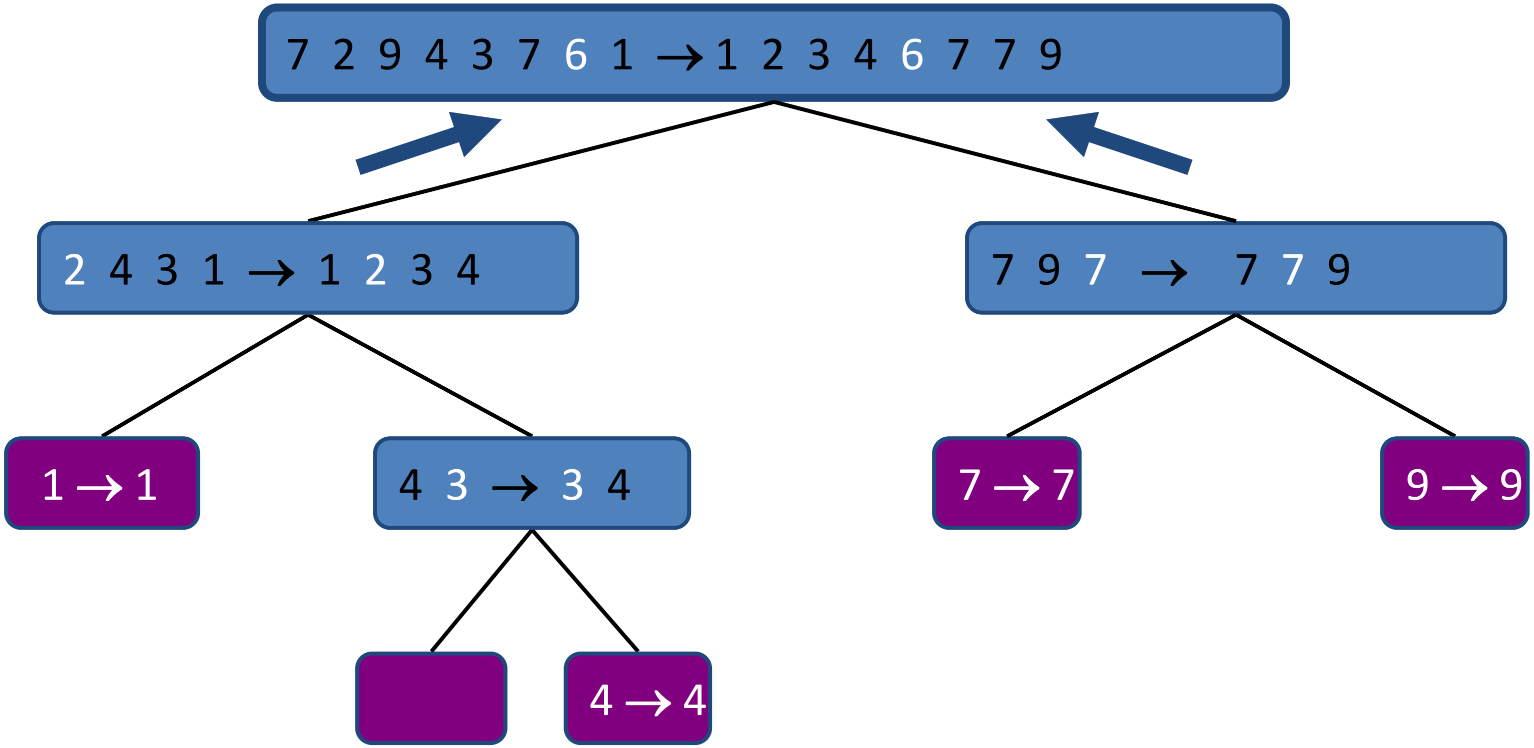
Quicksort (Example)

Partition, …, recursive call, base case



Quicksort (Example)

Join, join



Quicksort

Pseudocode:

$\text{Quicksort}(A[1:n])$ $\textbf{If } n == 0 \textbf{ then return } \text{null}$

$\quad \textbf{If } n == 1 \textbf{ then}$

$\quad\quad \textbf{return } A[1]$

$\quad \textbf{Choose pivot } x$

$\quad \text{splitindex} \leftarrow 1$

$\quad \textbf{For } i = 1 \text{ to } n \textbf{ do}$

$\quad\quad \textbf{If } A[i] < x \textbf{ then}$

$\quad\quad\quad \text{swap } A[i] \text{ with } A[\text{splitindex}]$

$\quad\quad\quad \text{splitindex} \leftarrow \text{splitindex} + 1$

$\quad \text{swap } x \text{ with } A[\text{splitindex}]$

$\quad L = \text{Quicksort } (A[1 : \text{splitindex} - 1])$

$\quad R = \text{Quicksort } (A[\text{splitindex} + 1 : n])$

$\quad \textbf{return } L[1 : \text{splitindex} - 1], x, R[\text{splitindex} + 1 : n]$

Quicksort

Pseudocode:

Quicksort($A[1:n]$) **If** $n == 0$ **then return** null

If $n == 1$ **then**

return $A[1]$

Choose pivot x

splitindex $\leftarrow 1$

For $i = 1$ to n **do**

If $A[i] < x$ **then**

swap $A[i]$ with $A[\text{splitindex}]$

splitindex $\leftarrow$ splitindex $+ 1$

swap x with $A[\text{splitindex}]$

$L = \text{Quicksort}\ (A[1 : \text{splitindex} - 1])$

$R = \text{Quicksort}\ (A[\text{splitindex} + 1 : n])$

return $L[1 : \text{splitindex} - 1], x, R[\text{splitindex} + 1 : n]$

Base case

Pivot

Partition

Recursion

Quicksort

Running time: Depends on the choice of the pivot.

Quicksort

Running time: Depends on the choice of the pivot.

Worst case: Pivot is the unique minimum or maximum element. Either ***L*** and ***R*** has size ***n*** - 1 and the other has size 0.

Example: 9 7 4 3 2 1

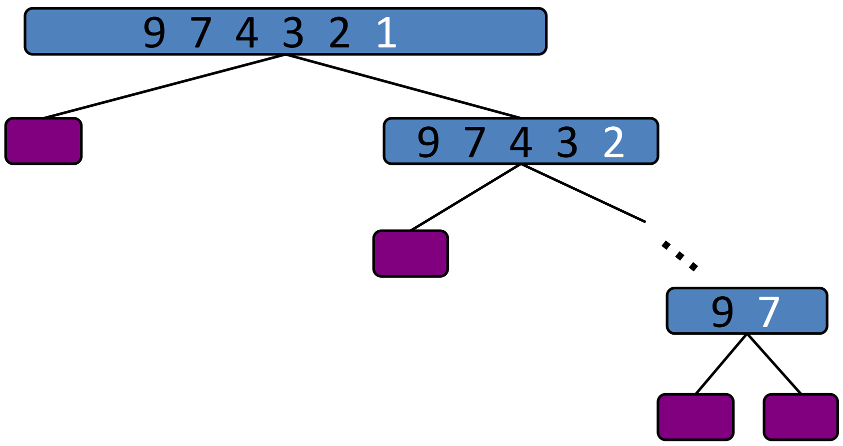
Choose pivots as follows: 1, then 2, then 3, then 4, then 7, then 9.

Quicksort

Running time: Depends on the choice of the pivot.

Worst case: Pivot is the unique minimum or maximum element. Either ***L*** and ***R*** has size ***n*** - 1 and the other has size 0.

Example: 9 7 4 3 2 1



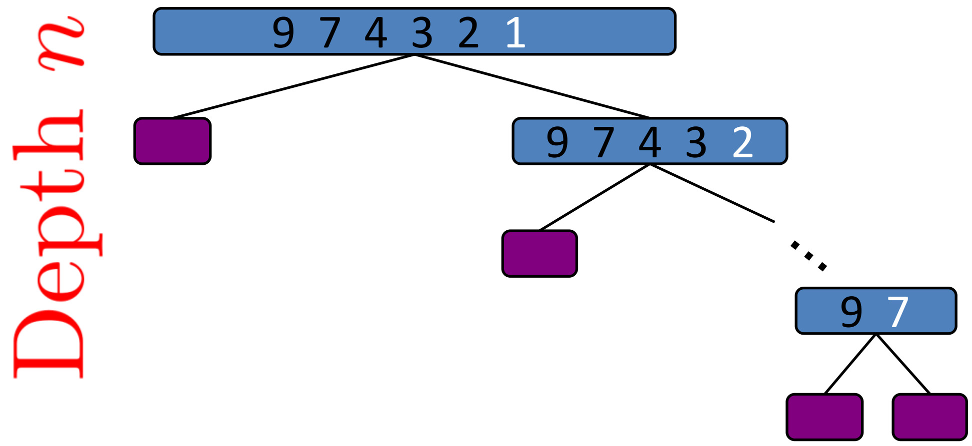
Quicksort

Running time: Depends on the choice of the pivot.

Worst case: Pivot is the unique minimum or maximum element. Either ***L*** and ***R*** has size ***n*** - 1 and the other has size 0.

Example: 9 7 4 3 2 1

Number of computations of order
$n + (n-1) + \cdots + 2 + 1 \in \Theta(n^2)$

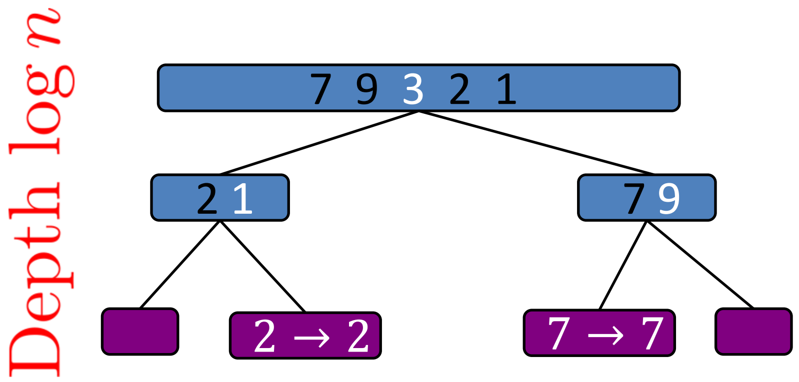


Quicksort

Running time: Depends on the choice of the pivot.

Average case: Random pivot gives expected time $\Theta(n \log n)$.

Idea: The pivot splits equally the array (the depth of the tree will be $\log n$)



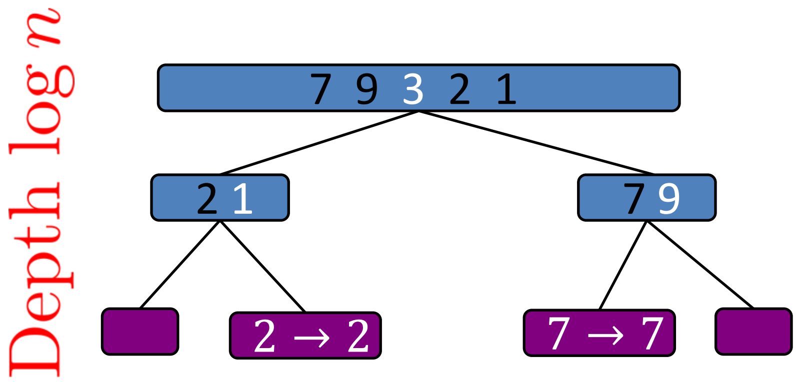
Quicksort

Running time: Depends on the choice of the pivot.

Average case: Random pivot gives expected time $\Theta(n \log n)$.

Idea: The pivot splits equally the array (the depth of the tree will be $\log n$)

Can we achieve
$\Theta(n\log n)$ in worst case?



Case study V: Median-selection

Idea: The pivot splits equally the array (the depth of the tree will be $\log n$). Choose median as pivot.

Quicksort Running time:

$$T(n) = 2T(n/2) + \Theta(n) + T_{median}(n)$$

If we can find median in $\Theta(n)$ then by Master thm: Quicksort in $\Theta(n \log n)$ time.

Case study V: Median-selection

Problem: Given an array A of n numbers, find the median in $\Theta(n)$ time.

Example 1: $A = [9,7,4,3,1,2]$. Answer: 3 or 4.
Example 2: $A = [9,7,17,3,10]$. Answer: 9.

Case study V: Median-selection

Problem: Given an array A of n numbers, find the median in $\Theta(n)$ time.

Example 1: $A = [9,7,4,3,1,2]$. Answer: 3 or 4.
Example 2: $A = [9,7,17,3,10]$. Answer: 9.

Idea: Unfortunately sorting and picking the middle position needs $\Theta(n \log n)$ time. Use divide and conquer.

Case study V: Median-selection

Problem: Given an array A of n numbers, find the median in $\Theta(n)$ time.

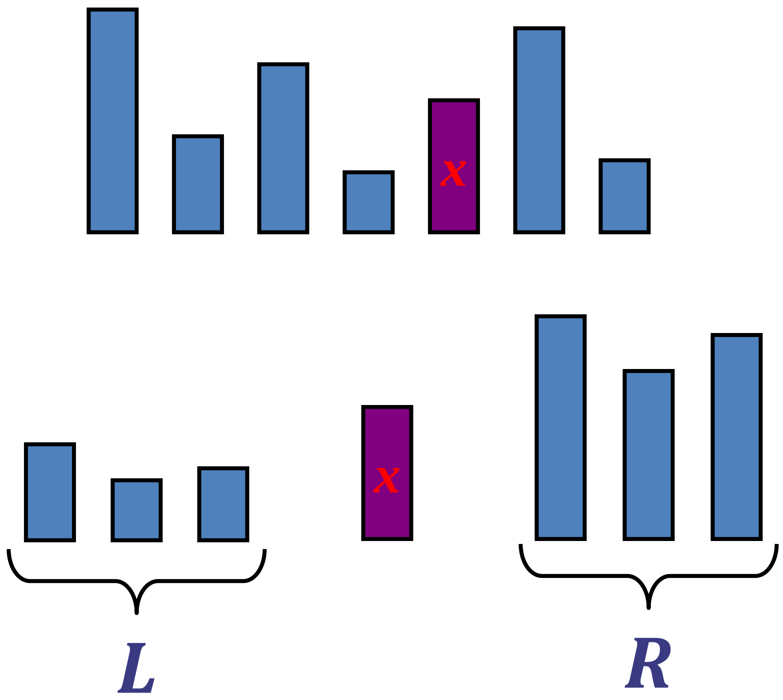
Idea: Use divide and conquer. Let’s try to solve the more general problem of selection.

Problem: Given an array A of n numbers and positive integer k, find the k-th smallest in $\Theta(n)$ time. Median when?

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

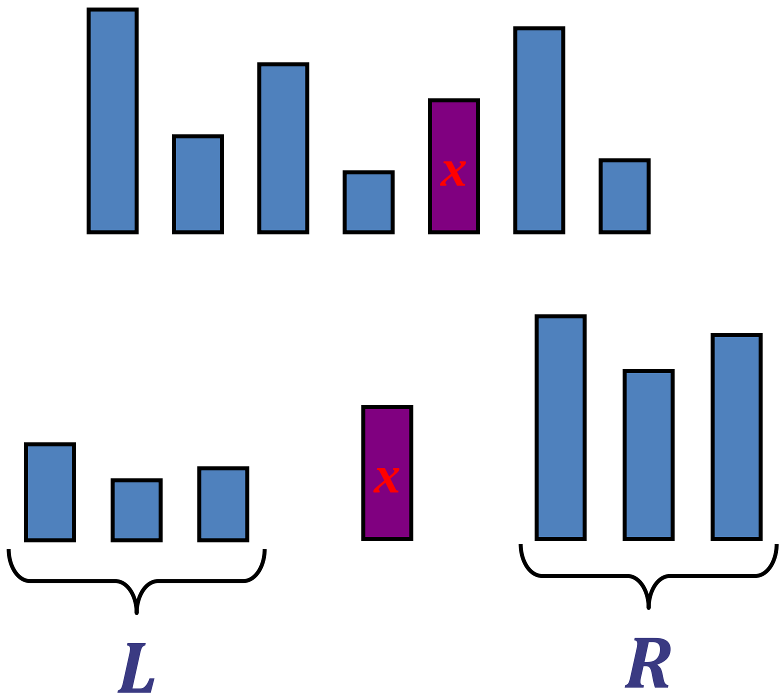
- **Divide:** pick an element $\boldsymbol{x}$ (called pivot) and partition into L, $\{x\}$ and R.



Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

- **Divide:** pick an element x (called pivot) and partition into L, $\{x\}$ and R.

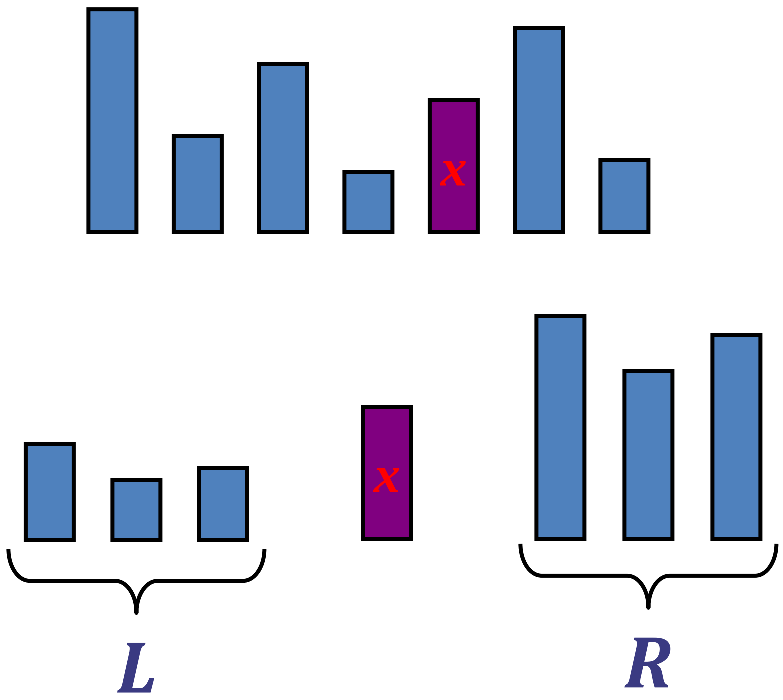


Where is the k-th element?

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

- **Divide:** pick an element $\boldsymbol{x}$ (called pivot) and partition into L, $\{x\}$ and R.



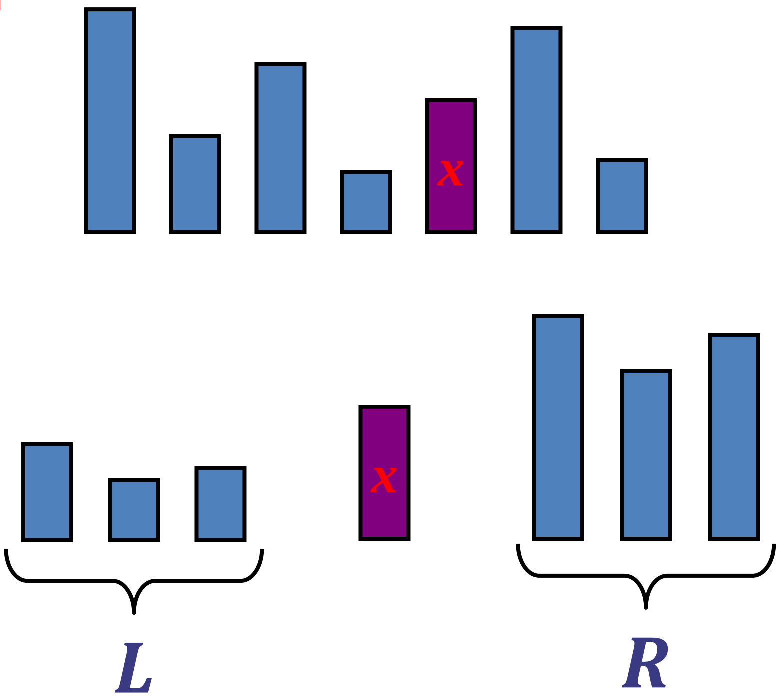
Where is the k-th element? Depends on the size of L!

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

- **Divide:** pick an element ***x*** (called pivot) and partition into L, $\{x\}$ and R.
- Conquer and Combine:

If $|L| \geq k$ then recursively find k-th in L.



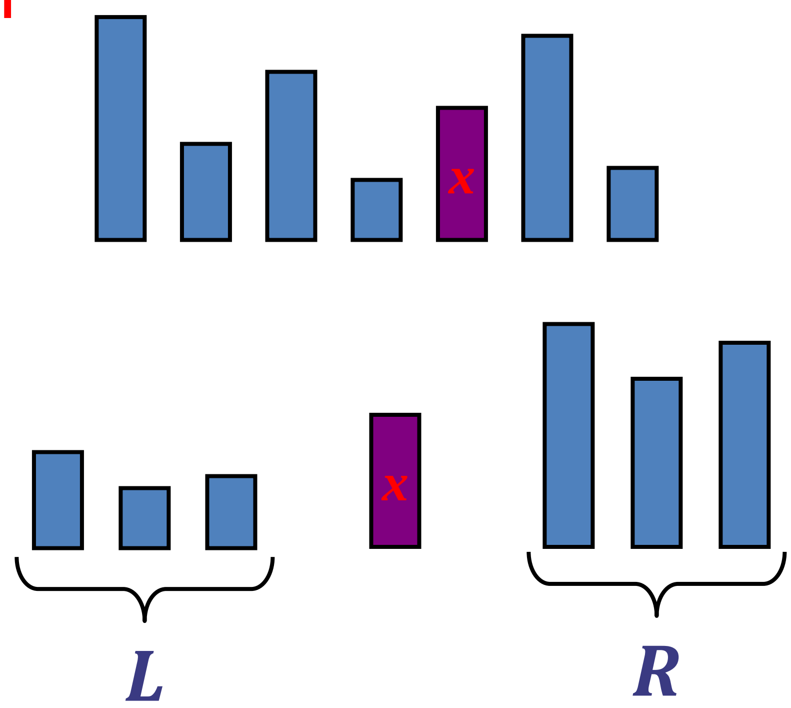
Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

- **Divide:** pick an element x (called pivot) and partition into L, $\{x\}$ and R.
- Conquer and Combine:

If $|L| \geq k$ then recursively find k-th in L.

If $|L| = k - 1$ then x is k-th element.



Case study V: Median-selection

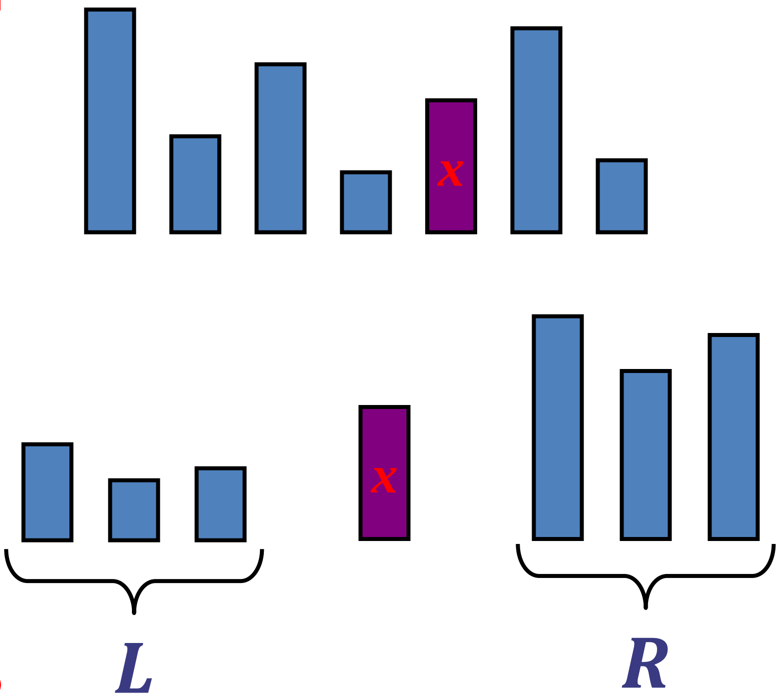
Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

- **Divide:** pick an element $\boldsymbol{x}$ (called pivot) and partition into L, $\{x\}$ and R.
- Conquer and Combine:

If $|L| \geq k$ then recursively find k-th in L.

If $|L| = k - 1$ then x is k-th element.

If $|L| < k - 1$ then recursively find $k - L - 1$-th element in R.



Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

Pseudocode:

Quickselect(A, k)

If $\text{len}(A) == 1$ **then**

return $A[1]$

Choose pivot x

$L =$ elements less than x

$R =$ elements greater than x

If $k <= |L|$ **then**

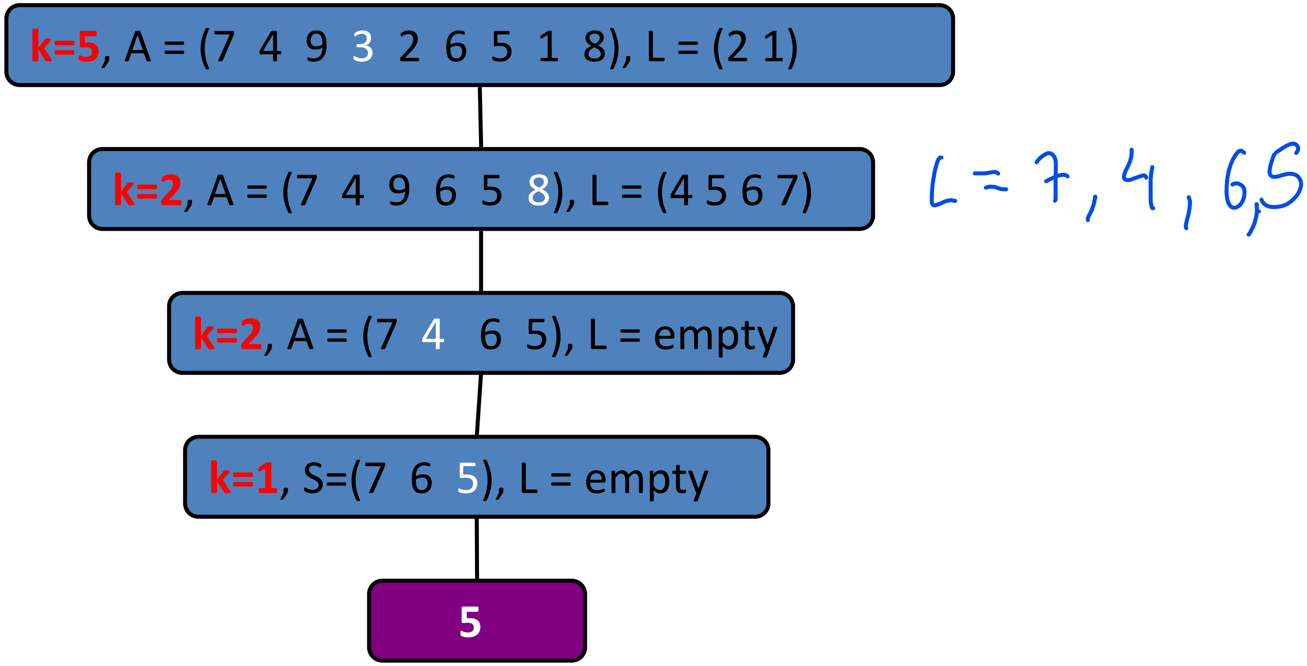
Quickselect(L, k)

else If $k == |L| + 1$ **then** **return** x

else Quickselect$(R, k - L - 1)$

Case study V: Median-selection

Example: Each node represents a recursive call of quick-select



Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

Pseudocode:

$\text{Quickselect}(A, k)$

If $\text{len}(A) == 1$ **then**
  return $A[1]$

Choose pivot x

$L =$ elements less than x

$R =$ elements greater than x

If $k <= |L|$ **then**
  $\text{Quickselect}(L, k)$

else If $k == |L| + 1$ **then** **return** x

else $\text{Quickselect}(R, k - L - 1)$

Running time?
Depends on the choice of pivot

Good pivots:
L and R have both at least $c \cdot n$ elements

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

Main idea: Recursively use quickselect algorithm itself to find a good pivot:

- Divide A into $n/5$ sets of 5 each
- Find a median in each 5-member set (constant time)
- Recursively find the median of the medians.

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

870	647	845	742	372
989	151	100	729	101
595	524	930	259	133
839	139	735	590	782
875	150	791	779	792

882	691	341	461	596
397	825	587	363	283
955	620	970	430	280
913	378	474	255	739

Median of 742, 596, 151, 397, 524, 620, 735, 474, 791 is 596 which is our **pivot.**

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

870	647	845	742	372	882	691	341	461	596
989	151	100	729	101	397	825	587	363	283
595	524	930	259	133	955	620	970	430	280
839	139	735	590	782	913	378	474	255	739
875	150	791	779	792					

100	283	255	133	341				
101	363	378	259	461				
151	397	474	524	596	620	735	742	791
				691	955	782	845	792
				882	970	839	870	875

Case study V: Median-selection

Problem: Given an array A of n numbers, find the k-th smallest in $\Theta(n)$ time.

100	283	255	133	341				
101	363	378	259	461				
151	397	474	524	596	620	735	742	791
				691	955	782	845	792
				882	970	839	870	875

Observation: L, R have size at least $3n/10$. So, to get the pivot we need time:

$T(n) = T(n/5) + T(7n/10) + \Theta(n)$. This yields $\Theta(n)$!

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101$, b=111. Answer: 100011.

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101$, b=111. Answer: 100011.

Standard Algorithm: $\Theta(n^2)$ time. Summing two n-bit numbers takes $\Theta(n)$ time.

Addition

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiplication

									1	1	0	1	0	1	0	1
								*	0	1	1	1	1	1	0	1
								1	1	0	1	0	1	0	1	0
							0	0	0	0	0	0	0	0	0	
						1	1	0	1	0	1	0	1	0		
					1	1	0	1	0	1	0	1	0			
				1	1	0	1	0	1	0	1	0				
			1	1	0	1	0	1	0	1	0					
		1	1	0	1	0	1	0	1	0						
	0	0	0	0	0	0	0	0	0							
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101$, b=111. Answer: 100011.

Standard Algorithm: $\Theta\left(n^2\right)$ time. Summing two n-bit numbers takes $\Theta(n)$ time.

Can we do better?

Addition

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiplication

									1	1	0	1	0	1	0	1
								*	0	1	1	1	1	1	0	1
								1	1	0	1	0	1	0	1	0
							0	0	0	0	0	0	0	0	0	
						1	1	0	1	0	1	0	1	0		
					1	1	0	1	0	1	0	1	0			
				1	1	0	1	0	1	0	1	0				
			1	1	0	1	0	1	0	1	0					
		1	1	0	1	0	1	0	1	0						
	0	0	0	0	0	0	0	0	0							
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L} \; \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L} \; \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L} \; \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L} \; \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Recursively

$$a \cdot b = a_R \cdot b_R + 2^{\frac{n}{2}} a_L \cdot b_R + a_R \cdot 2^{\frac{n}{2}} b_L + 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L$$

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L} \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L} \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Recursively

$$a \cdot b = a_R \cdot b_R + 2^{\frac{n}{2}} a_L \cdot b_R + a_R \cdot 2^{\frac{n}{2}} b_L + 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L$$

Running time: $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow \Theta(n^2)$ by Master thm

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L}\ \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L}\ \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

$$a \cdot b = 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L + a_R \cdot b_R +$$
$$2^{\frac{n}{2}}(\,(a_L - a_R) \cdot (b_R - b_L) + a_L \cdot b_L + a_R \cdot b_R)$$

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L}\ \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L}\ \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Recursively compute

1. $(a_L - a_R)(b_R - b_L)$
2. $a_L \cdot b_L$
3. $a_R \cdot b_R$

$$a \cdot b = 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L + a_R \cdot b_R +$$

$$2^{\frac{n}{2}}(\,(a_L - a_R) \cdot (b_R - b_L) + a_L \cdot b_L + a_R \cdot b_R)$$

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L} \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L} \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

Recursively compute

1. $(a_L - a_R)(b_R - b_L)$
2. $a_L \cdot b_L$
3. $a_R \cdot b_R$

$\Theta(n^{1.585})$

Running time: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow \Theta(n^{\log_2 3})$ by Master thm