Quicksort

COMS10018 - Algorithms

Dr Christian Konrad

Quicksort

Sorting Algorithms seen so far:

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes
Quicksort	$O(n^2)$	$O(n \log n)$	no	yes

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes
Quicksort	$O(n^2)$	$O(n \log n)$	no	yes

Quicksort

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes
Quicksort	$O(n^2)$	$O(n \log n)$	no	yes

Quicksort

- Very efficient in practice!

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes
Quicksort	$O(n^2)$	$O(n \log n)$	no	yes

Quicksort

- Very efficient in practice!
- *In place version of Mergesort:*

Sorting Algorithms seen so far:

	Worst case	Average case	stable?	in place?
Insertionsort	$O(n^2)$	$O(n^2)$	yes	yes
Mergesort	$O(n \log n)$	$O(n \log n)$	yes	no
Heapsort	$O(n \log n)$	$O(n \log n)$	no	yes
Quicksort	$O(n^2)$	$O(n \log n)$	no	yes

Quicksort

- Very efficient in practice!
- *In place version of Mergesort:*

$A[0, \lfloor \frac{n}{2} \rfloor] \leftarrow \textsc{MergeSort}(A[0, \lfloor \frac{n}{2} \rfloor])$
$A[\lfloor \frac{n}{2} \rfloor + 1, n-1] \leftarrow \textsc{MergeSort}(A[\lfloor \frac{n}{2} \rfloor, n-1])$
$A \leftarrow \textsc{Merge}(A)$
return A

recursive calls in Mergesort

Mergesort versus Quicksort

Mergesort versus Quicksort

- *Mergesort:* First solve subproblems recursively, then merge their solutions

Mergesort versus Quicksort

Mergesort versus Quicksort

- *Mergesort:* First solve subproblems recursively, then merge their solutions
- *Quicksort:* First partition problem into two subproblems in a clever way so that no extra work is needed when combining the solutions to the subproblems, then solve subproblems recursively

Quicksort

Divide and Conquer Algorithm:

Quicksort

Divide and Conquer Algorithm:

- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.

Divide and Conquer Algorithm:

- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.
- **Conquer:** Sort $A[0, p-1]$ and $A[p+1, n-1]$ recursively.

Quicksort

Divide and Conquer Algorithm:

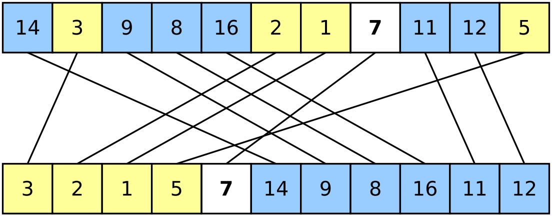
- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.
- **Conquer:** Sort $A[0, p-1]$ and $A[p+1, n-1]$ recursively.

14	3	9	8	16	2	1	**7**	11	12	5

				7						

Divide and Conquer Algorithm:

- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.
- **Conquer:** Sort $A[0, p-1]$ and $A[p+1, n-1]$ recursively.



Divide and Conquer Algorithm:

- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.
- **Conquer:** Sort $A[0, p-1]$ and $A[p+1, n-1]$ recursively.

14	3	9	8	16	2	1	**7**	11	12	5

1	2	3	5	**7**	8	9	11	12	14	16

Quicksort

Divide and Conquer Algorithm:

- **Divide:** Chose a good *pivot* $A[q]$. Rearrange A such that every element $\leq A[q]$ is left of $A[q]$ in the resulting ordering and every element $> A[q]$ is right of $A[q]$ in the resulting ordering. Let p be the new position of $A[q]$.
- **Conquer:** Sort $A[0, p-1]$ and $A[p+1, n-1]$ recursively.

14	3	9	8	16	2	1	**7**	11	12	5

1	2	3	5	**7**	8	9	11	12	14	16

- **Combine:** No work needed

Quicksort (2)

We need to address:

Quicksort (2)

We need to address:

1. We need to be able to rearrange the elements around the pivot in $O(n)$ time

Quicksort (2)

We need to address:

1. We need to be able to rearrange the elements around the pivot in $O(n)$ time
2. What is a good pivot? Ideally we would like to obtain subproblems of equal/similar sizes

The Partition Step

Partition Step:

The Partition Step

Partition Step:

- **Input:** Array A of length n

The Partition Step

Partition Step:

- **Input:** Array A of length n
- **Output:** Partitioning around pivot $A[n-1]$

Partition Step:

- **Input:** Array A of length n
- **Output:** Partitioning around pivot $A[n-1]$

```
Require: Array A of length n
  x ← A[n − 1]
  i ← −1
  for j ← 0 . . . n − 1 do
    if A[j] ≤ x then
      i ← i + 1
      exchange A[i] with A[j]
  return i
```

PARTITION

The Partition Step

Partition Step:

- **Input:** Array A of length n
- **Output:** Partitioning around pivot $A[n-1]$

Require: Array A of length n
   $x \leftarrow A[n-1]$
   $i \leftarrow -1$
   for $j \leftarrow 0 \dots n-1$ **do**
      if $A[j] \leq x$ **then**
         $i \leftarrow i+1$
         exchange $A[i]$ with $A[j]$
   return i

PARTITION

Pivot: Algorithm assumes pivot is $A[n-1]$ (if different pivot $A[q]$ is used: swap $A[q]$ with $A[n-1]$).

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

i	j										
	14	3	9	8	16	2	1	5	11	12	7

x: $\boxed{7}$

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

i		j									
	14	3	9	8	16	2	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

	i	j									
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Example

$$
\begin{array}{l}
x \leftarrow A[n-1] \\
i \leftarrow -1 \\
\textbf{for } j \leftarrow 0 \ldots n-1 \textbf{ do} \\
\quad \textbf{if } A[j] \leq x \textbf{ then} \\
\quad\quad i \leftarrow i+1 \\
\quad\quad \text{exchange } A[i] \text{ with } A[j]
\end{array}
$$

	i		j								
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

	i			j							
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

	i				j						
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

	i					j					
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

		i				j					
	3	2	9	8	16	14	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

		i					j				
	3	2	9	8	16	14	1	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

			i				j				
	3	2	1	8	16	14	9	5	11	12	7

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

			i					j			
	3	2	1	8	16	14	9	5	11	12	7

x: $\boxed{7}$

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

				i				j			
	3	2	1	5	16	14	9	8	11	12	7

x: 7

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

				i					j		
	3	2	1	5	16	14	9	8	11	12	7

x: $\boxed{7}$

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

				i						j	
	3	2	1	5	16	14	9	8	11	12	7

x: 7

Example

$$
\begin{array}{l}
x \leftarrow A[n-1] \\
i \leftarrow -1 \\
\textbf{for } j \leftarrow 0 \ldots n-1 \textbf{ do} \\
\quad \textbf{if } A[j] \leq x \textbf{ then} \\
\quad\quad i \leftarrow i+1 \\
\quad\quad \text{exchange } A[i] \text{ with } A[j]
\end{array}
$$

				i							j
	3	2	1	5	16	14	9	8	11	12	7

x: $\boxed{7}$

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

					i						j
	3	2	1	5	7	14	9	8	11	12	16

x: 7

Example

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

					i						j
	3	2	1	5	**7**	14	9	8	11	12	16

x: 7

Loop Invariant

Invariant:

Loop Invariant

Invariant: At the beginning of the for loop, the following holds:

1. Elements left of i (including i) are smaller or equal to x:

$$\text{For } 0 \leq k \leq i : A[k] \leq x$$

Loop Invariant

Invariant: At the beginning of the for loop, the following holds:

1. Elements left of i (including i) are smaller or equal to x:

$$\text{For } 0 \leq k \leq i : A[k] \leq x$$

2. Elements right of i (excluding i) and left of j (excluding j) are larger than x:

$$\text{For } i+1 \leq k \leq j-1 : A[k] > x$$

Proof of Loop Invariant

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$$
\begin{array}{l}
x \leftarrow A[n-1] \\
i \leftarrow -1 \\
\textbf{for } j \leftarrow 0 \ldots n-1 \textbf{ do} \\
\quad \textbf{if } A[j] \leq x \textbf{ then} \\
\quad\quad i \leftarrow i+1 \\
\quad\quad \text{exchange } A[i] \text{ with } A[j]
\end{array}
$$

Initialization: $i = -1, j = 0$

i	j												
	14	3	9	8	16	2	1	5	11	12	7	x:	7

Proof of Loop Invariant

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Initialization: $i = -1, j = 0$ ✓

i	j										
	14	3	9	8	16	2	1	5	11	12	7

x: 7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented

	i				j						
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented

	i				j						
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented $\checkmark$

	i					j					
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented ✓
2. $A[j] \leq x$: Then i is incremented, $A[i]$ and $A[j]$ are exchanged, and j is incremented

	i					j					
	3	14	9	8	16	2	1	5	11	12	7

x: 7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented ✓
2. $A[j] \leq x$: Then i is incremented, $A[i]$ and $A[j]$ are exchanged, and j is incremented

		i				j							
	3	14	9	8	16	2	1	5	11	12	7	x:	7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented ✓
2. $A[j] \leq x$: Then i is incremented, $A[i]$ and $A[j]$ are exchanged, and j is incremented

		i				j							
	3	2	9	8	16	14	1	5	11	12	7	x:	7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented ✓
2. $A[j] \leq x$: Then i is incremented, $A[i]$ and $A[j]$ are exchanged, and j is incremented

		i					j						
	3	2	9	8	16	14	1	5	11	12	7	x:	7

Proof of Loop Invariant (2)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \dots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Maintenance: Two cases:

1. $A[j] > x$: Then j is incremented $\checkmark$
2. $A[j] \leq x$: Then i is incremented, $A[i]$ and $A[j]$ are exchanged, and j is incremented $\checkmark$

		i					j						
	3	2	9	8	16	14	1	5	11	12	7	x:	7

Proof of Loop Invariant (3)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

```
x ← A[n − 1]
i ← −1
for j ← 0 . . . n − 1 do
    if A[j] ≤ x then
        i ← i + 1
        exchange A[i] with A[j]
```

Termination: (useful property showing that algo. is correct)

- $A[i]$ contains pivot element x that was located initially at position $n-1$
- All elements left of $A[i]$ are smaller equal to x
- All elements right of $A[i]$ are larger than x

Proof of Loop Invariant (3)

1. Left of i (including i): smaller equal to x
2. Right of i and left of j (excl. j): larger than x

$x \leftarrow A[n-1]$
$i \leftarrow -1$
for $j \leftarrow 0 \ldots n-1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i+1$
    exchange $A[i]$ with $A[j]$

Termination: (useful property showing that algo. is correct) ✓

- $A[i]$ contains pivot element x that was located initially at position $n-1$
- All elements left of $A[i]$ are smaller equal to x
- All elements right of $A[i]$ are larger than x

Require: array A of length n
  if $n \leq 1$ **then**
    return A
  else
    $i \leftarrow \text{Partition}(A)$
    $\textsc{Quicksort}(A[0, i-1])$
    $\textsc{Quicksort}(A[i+1, n-1])$

Algorithm $\textsc{Quicksort}$

Quicksort

Require: array A of length n
  if $n \leq 1$ **then**
    return A
  else
    $i \leftarrow$ Partition(A)
    $\textsc{Quicksort}(A[0, i-1])$
    $\textsc{Quicksort}(A[i+1, n-1])$

Algorithm $\textsc{Quicksort}$

What is the runtime of Quicksort?