QUICKSORT

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

Agenda

Quicksort[1]

Divides the input elements (array) according to their value

$$\underbrace{A[0]\ldots A[s-1]}_{\text{all are } \leq A[s]} \quad A[s] \quad \underbrace{A[s+1]\ldots A[n-1]}_{\text{all are } \geq A[s]}$$

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← Partition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

Quicksort:

- Divide: not immediate
- Conquer: immediate

Mergesort:

- Divide: immediate
- Conquer: not immediate

Centro de Informática UFPE

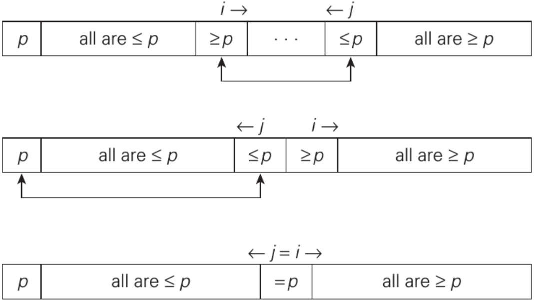
Universidade Federal de Pernambuco

[1] Source: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Partition strategy proposed by C.A.R. Hoare[3]

Idea: scan the array from both ends (i = left to right / j = right to left)

- If $A[i] < p$, increment i. If $A[j] > p$, decrement j.
- When both scans stop, three situations may arise[2].



[2] Considering the check $i \geq r$, we might have $i = j$, but $A[i] \neq p$ and $A[j] \neq p$.

[3] Source: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

`quicksort([5,2,1,7,0], 0,4)`

↓

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	7	0	-
	l				r	

$s = ?$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

quicksort(
[5,2,1,7,0],
0,4)

↓

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	7	0	-
	l				r	
	i					j

$p = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
// undo last swap when $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	7	0	-
	l				r	
				i	j	

$p = 5$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

quicksort([5,2,1,7,0], 0,4)

↓

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	0	7	-
	l				r	
				i	j	

$p = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

`quicksort([5,2,1,7,0], 0,4)`

↓

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	0	7	-
	l				r	
				j	i	

$p = 5$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
   0,4)
```

$\downarrow$

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	7	0	-
	l				r	
				j	i	

$p = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	5	2	1	0	7	-
	l				r	
				j	i	

$p = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

↓

$(\_, 0, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l				r	
				j	i	

$p = 5$

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

quicksort([5,2,1,7,0], 0,4)

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l				r	

$s = 3$

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**

  `// s = split position`

  $s \leftarrow HoarePartition(A, l, r)$;

  $Quicksort(A, l, s-1)$;

  $Quicksort(A, s+1, r)$;

`quicksort([5,2,1,7,0],0,4)`

$\downarrow$

$(\_, 0, 4)$

$\downarrow \qquad \searrow$

$(\_, 0, 2) \qquad (\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			

$s = ?$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

quicksort(
[5,2,1,7,0],
0,4)

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			
	i			j		

$p = 0$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

$\downarrow \qquad \searrow$

$(\_, 0, 2) \qquad (\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			
	j	i				

$p = 0$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

`quicksort([5,2,1,7,0], 0,4)`

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	2	0	1	5	7	-
	l		r			
	j	i				

$p = 0$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

quicksort([5,2,1,7,0],0,4)

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			
	j	i				

$p = 0$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			
	j	i				

$p = 0$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**
    // s = split position
    $s \leftarrow HoarePartition(A, l, r)$;
    $Quicksort(A, l, s-1)$;
    $Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

$(\_, 0, 2)$ $(\_, 4, 4)$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			

$s = 0$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

quicksort([5,2,1,7,0],0,4)

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

$\downarrow$ $\searrow$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l					

$r = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**

```
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

quicksort([5,2,1,7,0], 0,4)

$$\downarrow$$

$(\_, 0, 4)$

$\downarrow$ $(\_, 0, 2)$ $\searrow$ $(\_, 4, 4)$

$\downarrow$ $(\_, 0, -1)$ $\searrow$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
	l		r			

$s = 0$

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**
    `// s = split position`
    $s \leftarrow HoarePartition(A, l, r)$;
    $Quicksort(A, l, s-1)$;
    $Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

$\downarrow$ $\searrow$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
		l	r			

$s = ?$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

$p \leftarrow A[l]$;
$i \leftarrow l$;
$j \leftarrow r + 1$;
repeat
  repeat
    $i \leftarrow i + 1$;
  until $A[i] \geq p \vee i \geq r$;
  repeat
    $j \leftarrow j - 1$;
  until $A[j] \leq p$;
  swap $A[i]$ and $A[j]$;
until $i \geq j$;
`// undo last swap when` $i \geq j$
swap $A[i]$ and $A[j]$;
swap $A[l]$ and $A[j]$;
return j;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

$\downarrow$ $\searrow$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
		l	r			
		i		j		

$p = 2$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

`quicksort([5,2,1,7,0], 0,4)`

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

$\downarrow$ $\searrow$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
		l	r			
			i,j			

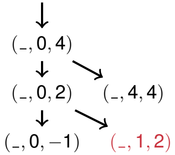
$p = 2$

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

`quicksort( [5,2,1,7,0], 0,4)`



index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
		l	r			
			i,j			

$p = 2$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

`quicksort([5,2,1,7,0], 0,4)`

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 4, 4)$

$\downarrow$ $\searrow$

$(\_, 0, -1)$ $(\_, 1, 2)$

index:	0	1	2	3	4	5
A:	0	2	1	5	7	-
		l	r			
			i,j			

$p = 2$

Centro de Informática UFPE

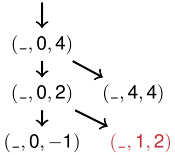
Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: HoarePartition(A[0..n-1],l,r)

```
p ← A[l];
i ← l;
j ← r + 1;
repeat
    repeat
        i ← i + 1;
    until A[i] ≥ p ∨ i ≥ r;
    repeat
        j ← j − 1;
    until A[j] ≤ p;
    swap A[i] and A[j];
until i ≥ j;
// undo last swap when i ≥ j
swap A[i] and A[j];
swap A[l] and A[j];
return j;
```

quicksort(
[5,2,1,7,0],
0,4)



index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
		l	r			
			i,j			

$p = 2$

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**

```
// s = split position
```

$s \leftarrow HoarePartition(A, l, r)$;

$Quicksort(A, l, s-1)$;

$Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$(\_, 0, 4)$

$(\_, 0, 2)$ $(\_, 4, 4)$

$(\_, 0, -1)$ $(\_, 1, 2)$

$(\_, 1, 1)$ $(\_, 3, 2)$

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
		l	r			

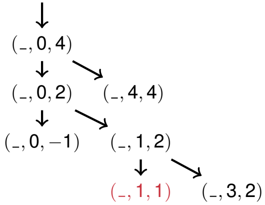
$s = 2$

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**
  `// s = split position`
  $s \leftarrow HoarePartition(A, l, r)$;
  $Quicksort(A, l, s-1)$;
  $Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```



index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
		l,r				

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**
    `// s = split position`
    $s \leftarrow HoarePartition(A, l, r)$;
    $Quicksort(A, l, s-1)$;
    $Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$(\_, 0, 4)$

$(\_, 0, 2)$ $(\_, 4, 4)$

$(\_, 0, -1)$ $(\_, 1, 2)$

$(\_, 1, 1)$ $(\_, 3, 2)$

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
		l	r			

$s = 2$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

if $l < r$ **then**
  // s = split position
  $s \leftarrow HoarePartition(A, l, r)$;
  $Quicksort(A, l, s-1)$;
  $Quicksort(A, s+1, r)$;

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$$\begin{array}{l} \downarrow \\ (\_, 0, 4) \\ \downarrow \quad \searrow \\ (\_, 0, 2) \quad (\_, 4, 4) \\ \downarrow \quad \searrow \\ (\_, 0, -1) \quad (\_, 1, 2) \\ \qquad\qquad \downarrow \quad \searrow \\ \qquad\qquad (\_, 1, 1) \quad (\_, 3, 2) \end{array}$$

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
			r	l		

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
	l				r	

$s = 3$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

```
quicksort(
    [5,2,1,7,0],
    0,4)
```

$(\_, 0, 4)$

$(\_, 0, 2)$ $(\_, 4, 4)$

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-
					l,r	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Partition strategy proposed by C.A.R. Hoare

Algorithm: Quicksort(A[0..n-1,l,r])

```
if l < r then
    // s = split position
    s ← HoarePartition(A, l, r);
    Quicksort(A, l, s − 1);
    Quicksort(A, s + 1, r);
```

`quicksort([5,2,1,7,0], 0,4)`

index:	0	1	2	3	4	5
A:	0	1	2	5	7	-

Quicksort: complexity (basic op. = comp. of keys)

Number of basic operations performed (after both scans):

- $n + 1$ if i crosses j
- n if i coincides with j

Best case: all splits happening in the middle

Assuming $n = 2^k$:

- $C_{best}(n) = 2C_{best}(n/2) + n$ for $n > 1$
- $C_{best}(1) = 0$ for $n \leq 1$

Considering the master theorem

- $C_{best}(n) = 2C_{best}(n/2) + n$ for $n > 1$, thus $a = b = 2$
- $f(n) = n \in \Theta(n) = \Theta(n^d)$, thus $d = 1$
- Since $a = b^d$ (i.e., $2 = 2^1$),
  $C_{best}(n) \in \Theta(n^d log\ n) = \Theta(n\ log\ n)$

Quicksort: complexity (basic op. = comp. of keys)

Worst case: one subarray is empty, the other one is just 1 less in size

Assuming $n = 2^k$:

- In the worst case, number of basic operations performed after both scans = $n + 1$ (j always crosses i)
- Size of the initial array ($A[0..n-1]$) = $(n-1) - 0 + 1 = n$
  - Number of basic operations for this array = $n + 1$
- Size of the last array ($A[n-2..n-1]$) = $(n-1) - (n-2) + 1 = 2$
  - Number of basic operations for this array = 3

Centro de Informática UFPE
Universidade Federal de Pernambuco

Quicksort: complexity (basic op. = comp. of keys)

$$
\begin{aligned}
C_{worst}(n) &= (n+1) + n + \cdots + 3 \\
&= ((n+1) + n + \cdots + 3 + 2 + 1) \\
&\quad -3 \\
&= \frac{(1+(n+1))((n+1)-1+1)}{2} - 3 \\
&= \frac{(n+2)(n+1)}{2} - 3 \\
&= \frac{n^2+n+2n+2-6}{2} \\
&= \frac{n^2+3n-4}{2} \in \Theta(n^2)
\end{aligned}
$$

Remember that:

- $S_n = \frac{(a_1+a_n)*n}{2}$ (arithm. prog.)

Quicksort: complexity (basic op. = comp. of keys)

$$
\begin{aligned}
C_{worst}(n) &= (n+1) + n + \cdots + 3 \\
&= ((n+1) + n + \cdots + 3 + 2 + 1) \\
&\quad -3 \\
&= \frac{(1+(n+1))((n+1)-1+1)}{2} - 3 \\
&= \frac{(n+2)(n+1)}{2} - 3 \\
&= \frac{n^2+n+2n+2-6}{2} \\
&= \frac{n^2+3n-4}{2} \in \Theta(n^2)
\end{aligned}
$$

Remember that:

- $S_n = \frac{(a_1+a_n)*n}{2}$ (arithm. prog.)

$C_{avg}(n) = 1.39n\ log\ n$

Pros and cons

- $\Uparrow$: typically, even better than mergesort
- $\Uparrow$: space efficiency
- $\Downarrow$: not stable

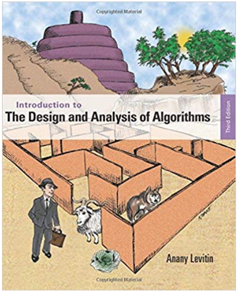
Complexity of sorting algorithms[4]

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

[4] Source: http://bigocheatsheet.com/

Agenda

Bibliography



Chapter 5 (pp. 176–181)
Anany Levitin.
Introduction to the Design and Analysis of Algorithms.
3rd edition. Pearson. 2011.

Chapter 7 (pp. 244–251)
Clifford Shaffer.
Data Structures and Algorithm Analysis.
Dover, 2013.

Centro de Informática UFPE
Universidade Federal de Pernambuco

QUICKSORT

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil