SELECTION SORT | INSERTION SORT | MERGESORT

Gustavo Carvalho
(ghpc@cin.ufpe.br)

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

Agenda

1. Selection sort (brute force)
2. Insertion sort (decrease-and-conquer)
3. Mergesort (divide-and-conquer)
4. Bibliography

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	89	45	68	90	29	34	17
	i						
	min						

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
    $min \leftarrow i$;
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        if $A[j] < A[min]$ **then** $min \leftarrow j$;
    swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	89	45	68	90	29	34	17
	i						
							min

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	45	68	90	29	34	89
	i						
							min

Centro de Informática UFPE
Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	45	68	90	29	34	89
		i					
		min					

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	45	68	90	29	34	89
		i					
					min		

Centro de Informática UFPE
Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	68	90	45	34	89
		i					
					min		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	29	68	90	45	34	89
			i				
			min				

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n - 2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i + 1$ **to** $n - 1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	68	90	45	34	89
			i				
						min	

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n - 2$ **do**
    $min \leftarrow i$;
    for $j \leftarrow i + 1$ **to** $n - 1$ **do**
        if $A[j] < A[min]$ **then** $min \leftarrow j$;
    swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	90	45	68	89
			i				
						min	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n - 2$ **do**
    $min \leftarrow i$;
    for $j \leftarrow i + 1$ **to** $n - 1$ **do**
        if $A[j] < A[min]$ **then** $min \leftarrow j$;
    swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	90	45	68	89
				i			
				min			

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	90	45	68	89
				i			
					min		

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	29	34	45	90	68	89
				i			
					min		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n - 2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i + 1$ **to** $n - 1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	45	90	68	89
					i		
					min		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	29	34	45	90	68	89
					i		
						min	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n - 2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i + 1$ **to** $n - 1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	90	89
					i		
						min	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

```
for i ← 0 to n − 2 do
    min ← i;
    for j ← i + 1 to n − 1 do
        if A[j] < A[min] then min ← j;
    swap A[i] and A[min];
```

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	90	89
						i	
						min	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

1. **for** $i \leftarrow 0$ **to** $n-2$ **do**
2. $min \leftarrow i$;
3.   **for** $j \leftarrow i+1$ **to** $n-1$ **do**
4.   **if** $A[j] < A[min]$ **then** $min \leftarrow j$;
5.   swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	90	89
						i	
							min

Centro de Informática UFPE

Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	89	90
						i	
							min

Centro de Informática UFPE
Universidade Federal de Pernambuco

Selection sort

Idea: find the smallest element and exchange it with the first one

Algorithm: SelectionSort(A[0..n - 1])

for $i \leftarrow 0$ **to** $n-2$ **do**
  $min \leftarrow i$;
  for $j \leftarrow i+1$ **to** $n-1$ **do**
    if $A[j] < A[min]$ **then** $min \leftarrow j$;
  swap $A[i]$ and $A[min]$;

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	89	90
							i

Selection sort: complexity (basic op. = $A[j] < A[min]$)

$$
\begin{aligned}
C_{worst}(n) &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \\
&= \sum_{i=0}^{n-2} ((n-1) - (i+1) + 1) \\
&= \sum_{i=0}^{n-2} (n-1-i) \\
&= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i \\
&= (n-1) \sum_{i=0}^{n-2} 1 - \frac{(n-2)(n-1)}{2} \\
&= (n-1)^2 - \frac{(n-2)(n-1)}{2} \\
&= \frac{n^2-n}{2} \in \Theta(n^2)
\end{aligned}
$$

Remember that:

- $\sum_{i=l}^{u} c * a_i = c * \sum_{i=l}^{u} a_i$
- $\sum_{i=l}^{u} (a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i$
- $S_n = \frac{(a_1 + a_n) * n}{2}$ (arithm. prog.)

Regarding the #swaps: $S_{worst}(n) = n - 1 \in \Theta(n)$

Bubble sort: $C(n) \in \Theta(n^2)$, $S_{worst}(n) \in \Theta(n^2)$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Agenda

1 Selection sort (brute force)

2 Insertion sort (decrease-and-conquer)

3 Mergesort (divide-and-conquer)

4 Bibliography

Brute force *vs.* Decrease-and-conquer

Example: computing a^n

- Brute force ($\Theta(n)$): $\underbrace{a * \dots * a}_{n}$
- Decrease by a constant ($\Theta(n)$)

$$f(n) = \begin{cases} f(n-1) \cdot a & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

- Decrease by a constant factor ($\Theta(log\ n)$)

$$a^n = \begin{cases} (a^{n/2})^2 & \text{if } n \text{ is even and positive} \\ (a^{(n-1)/2})^2 \cdot a & \text{if } n \text{ is odd} \\ 1 & \text{if } n = 0 \end{cases}$$

Decrease-and-conquer

Design strategy: decrease-and-conquer

- Decrease by a constant
- Decrease by a constant factor
  - Other example: binary search
- Variable-size decrease
  - Example: $gcd(m, n) = gcd(n, m \bmod n)$

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	89	45	68	90	29	34	17
		i					
	j						

$v = 45$

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	89	89	68	90	29	34	17
		i					

$v = 45$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
    $v \leftarrow A[i]; j \leftarrow i-1;$
    while $j \geq 0 \wedge A[j] > v$ **do**
        $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
    $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	45	89	68	90	29	34	17
		i					

$v = 45$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we “insert” $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	45	89	68	90	29	34	17
			i				
		j					

$v = 68$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
  $v \leftarrow A[i]; j \leftarrow i-1;$
  while $j \geq 0 \land A[j] > v$ **do**
    $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
  $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	45	89	89	90	29	34	17
			i				
	j						

$v = 68$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	45	68	89	90	29	34	17
			i				
	j						

$v = 68$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
    $v \leftarrow A[i]; j \leftarrow i-1;$
    while $j \geq 0 \wedge A[j] > v$ **do**
        $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
    $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	45	68	89	90	29	34	17
				i			
			j				

$v = 90$

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	45	68	89	90	29	34	17
				i			
			j				

$v = 90$

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
  $v \leftarrow A[i]; j \leftarrow i-1;$
  while $j \geq 0 \wedge A[j] > v$ **do**
    $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
  $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	45	68	89	90	29	34	17
				i			
			j				

$v = 90$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we “insert” $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	45	68	89	90	29	34	17
					i		
				j			

$v = 29$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	45	45	68	89	90	34	17
					i		

$v = 29$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;      // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	29	45	68	89	90	34	17
					i		

$v = 29$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
  $v \leftarrow A[i]; j \leftarrow i-1;$
  while $j \geq 0 \wedge A[j] > v$ **do**
    $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
  $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	29	45	68	89	90	34	17
						i	
					j		

$v = 34$

Centro de Informática UFPE
Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	29	45	45	68	89	90	17
						i	
	j						

$v = 34$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	29	34	45	68	89	90	17
						i	
	j						

$v = 34$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	29	34	45	68	89	90	17
							i
						j	

$v = 17$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	29	29	34	45	68	89	90
							i

$v = 17$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \leq i \leq n$, to sort $A[0..n-i+1]$, we "insert" $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

for $i \leftarrow 1$ **to** $n-1$ **do**
  $v \leftarrow A[i]; j \leftarrow i-1;$
  while $j \geq 0 \wedge A[j] > v$ **do**
    $A[j+1] \leftarrow A[j]; j \leftarrow j-1;$ `// partial shift right`
  $A[j+1] \leftarrow v;$

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	89	90
							i

$v = 17$

$j = -1$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort

Idea: provided that $A[0..n-i]$ is sorted, where $2 \le i \le n$, to sort $A[0..n-i+1]$, we “insert” $A[n-i+1]$ appropriately (dec. by a constant)

Algorithm: InsertionSort(A[0..n - 1])

```
for i ← 1 to n − 1 do
    v ← A[i]; j ← i − 1;
    while j ≥ 0 ∧ A[j] > v do
        A[j + 1] ← A[j]; j ← j − 1;        // partial shift right
    A[j + 1] ← v;
```

index:	0	1	2	3	4	5	6
value:	17	29	34	45	68	89	90

$i = 7$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Insertion sort: complexity (basic op.: $A[j] > v$)

$$
\begin{aligned}
C_{worst}(n) &= \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} 1 \\
&= \sum_{i=1}^{n-1} ((i-1) - 0 + 1) \\
&= \sum_{i=1}^{n-1} i \\
&= \frac{(1+(n-1))((n-1)-1+1)}{2} \\
&= \frac{n^2-n}{2} \in \Theta(n^2)
\end{aligned}
$$

$$C_{best}(n) = \sum_{i=1}^{n-1} 1 = n - 1 - 1 + 1 = n - 1 \in \Theta(n)$$

$$C_{avg}(n) \approx \frac{n^2}{4} \in \Theta(n^2)$$

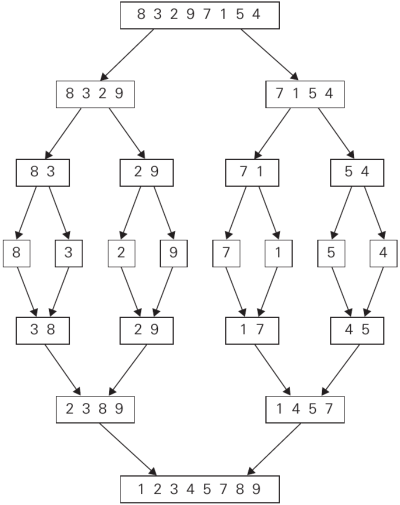
Agenda

1. Selection sort (brute force)
2. Insertion sort (decrease-and-conquer)
3. Mergesort (divide-and-conquer)
4. Bibliography

Divide-and-conquer

1. A problem is divided into several subproblems of the same type (ideally of about equal size)
2. The subproblems are solved
3. If necessary, the solutions to the subproblems are combined

Mergesort[1]



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

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$; $Mergesort(A, m + 1, r)$; $Merge(A, l, r)$;

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then** $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then** $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

1. **if** $l < r$ **then**
2. $m \leftarrow \lfloor (l + r)/2 \rfloor$;
3. $Mergesort(A, l, m)$;
4. $Mergesort(A, m + 1, r)$;
5. $Merge(A, l, r)$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

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

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

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

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

$$
\begin{array}{ll}
\downarrow & \\
(\_, 0, 4) & \\
\downarrow & \searrow \\
(\_, 0, 2) & (\_, 3, 4) \\
\downarrow & \searrow \\
(\_, 0, 1) & (\_, 2, 2) \\
\downarrow & \searrow \\
(\_, 0, 0) & (\_, 1, 1)
\end{array}
$$

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
- $m \leftarrow \lfloor (l + r)/2 \rfloor$;
- $Mergesort(A, l, m)$;
- $Mergesort(A, m + 1, r)$;
- $Merge(A, l, r)$;

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

$$
\begin{array}{ll}
\downarrow & \\
(\_, 0, 4) & \\
\downarrow & \searrow \\
(\_, 0, 2) & (\_, 3, 4) \\
\downarrow & \searrow \\
(\_, 0, 1) & (\_, 2, 2) \\
\downarrow & \searrow \\
(\_, 0, 0) & (\_, 1, 1)
\end{array}
$$

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

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

$\downarrow \quad \searrow$

$(\_, 0, 0) \quad (\_, 1, 1)$

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
- $m \leftarrow \lfloor (l + r)/2 \rfloor$;
- $Mergesort(A, l, m)$;
- $Mergesort(A, m + 1, r)$;
- $Merge(A, l, r)$;

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

$$
\begin{array}{ll}
\downarrow & \\
(\_, 0, 4) & \\
\downarrow & \searrow \\
(\_, 0, 2) & (\_, 3, 4) \\
\downarrow & \searrow \\
(\_, 0, 1) & (\_, 2, 2) \\
\downarrow & \searrow \\
(\_, 0, 0) & (\_, 1, 1)
\end{array}
$$

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

```
if l < r then
    m ← ⌊(l + r)/2⌋;
    Mergesort(A, l, m);
    Mergesort(A, m + 1, r);
    Merge(A, l, r);
```

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

↓ ↘

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

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

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

↓ ↘

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

index:	0	1	2	3	4
A:	5	2	1	7	0
	l	r			
temp:	5	2			
	i1	i2			

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$
$(\_,0,4)$
$\downarrow$ $\searrow$
$(\_,0,2)$ $(\_,3,4)$
$\downarrow$ $\searrow$
$(\_,0,1)$ $(\_,2,2)$

index:	0	1	2	3	4
A:	5	2	1	7	0
	l	r			
	c				
temp:	5	2			
	i1	i2			

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 3, 4)$

$\downarrow$ $\searrow$

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

index:	0	1	2	3	4
A:	2	2	1	7	0
	l	r			
	c				
temp:	5	2			
	i1		i2		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

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

index:	0	1	2	3	4
A:	2	2	1	7	0
	l	r			
		c			
temp:	5	2			
	i1		i2		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $(\_, 0, 2)$ $\searrow$ $(\_, 3, 4)$

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

index:	0	1	2	3	4
A:	2	5	1	7	0
	l	r			
		c			
temp:	5	2			
		i1	i2		

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 3, 4)$

$\downarrow$ $\searrow$

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

index:	0	1	2	3	4
A:	2	5	1	7	0
	l	r			
			c		
temp:	5	2			
		i1	i2		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
    $m \leftarrow \lfloor (l + r)/2 \rfloor$;
    $Mergesort(A, l, m)$;
    $Mergesort(A, m + 1, r)$;
    $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \qquad \searrow$

$(\_, 0, 2) \qquad (\_, 3, 4)$

$\downarrow \qquad \searrow$

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

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
- $m \leftarrow \lfloor (l + r)/2 \rfloor$;
- $Mergesort(A, l, m)$;
- $Mergesort(A, m + 1, r)$;
- $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

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

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 3, 4)$

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

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m+1$ **then** $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then** $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \qquad \searrow$

$(\_, 0, 2) \qquad (\_, 3, 4)$

index:	0	1	2	3	4
A:	2	5	1	7	0
	l		r		
temp:	2	5	1		
	i1		i2		

Mergesort

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

1. **for** $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
2. $m \leftarrow \lfloor (l+r)/2 \rfloor$;
3. $i1 \leftarrow l$; $i2 \leftarrow m+1$;
4. **for** $curr \leftarrow l$ **to** r **do**
5. **if** $i1 = m+1$ **then** $A[curr] \leftarrow temp[i2{+}{+}]$;
6. **else if** $i2 > r$ **then** $A[curr] \leftarrow temp[i1{+}{+}]$;
7. **else if** $temp[i1] \leq temp[i2]$ **then**
8. $A[curr] \leftarrow temp[i1{+}{+}]$;
9. **else** $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	2	5	1	7	0
	l		r		
	c				
temp:	2	5	1		
	i1		i2		

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow$ $\searrow$

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	5	1	7	0
	l		r		
	c				
temp:	2	5	1		
	i1			i2	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m+1$ **then**
        $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \qquad \searrow$

$(\_, 0, 2) \qquad (\_, 3, 4)$

index:	0	1	2	3	4
A:	1	5	1	7	0
	l		r		
		c			
temp:	2	5	1		
	i1			i2	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_,0,4)$

$\downarrow \quad \searrow$

$(\_,0,2) \quad (\_,3,4)$

index:	0	1	2	3	4
A:	1	2	1	7	0
	l		r		
		c			
temp:	2	5	1		
		i1		i2	

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

---

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m + 1$ **then**
      $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then**
      $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	1	7	0
	l		r		
			c		
temp:	2	5	1		
		i1		i2	

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then** $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then** $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	7	0
	l		r		
			c		
temp:	2	5	1		
			i1	i2	

Centro de Informática UFPE
UNIVERSIDADE FEDERAL DE PERNAMBUCO

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	7	0
	l		r		
				c	
temp:	2	5	1		
			i1	i2	

Mergesort

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

```
if l < r then
    m ← ⌊(l + r)/2⌋;
    Mergesort(A, l, m);
    Mergesort(A, m + 1, r);
    Merge(A, l, r);
```

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

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

Mergesort

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

if $l < r$ **then**

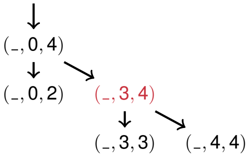
   $m \leftarrow \lfloor (l + r)/2 \rfloor$;

   $Mergesort(A, l, m)$;

   $Mergesort(A, m + 1, r)$;

   $Merge(A, l, r)$;

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



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

Mergesort

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

if $l < r$ **then**
- $m \leftarrow \lfloor (l + r)/2 \rfloor$;
- $Mergesort(A, l, m)$;
- $Mergesort(A, m + 1, r)$;
- $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

$(\_, 3, 3) \quad (\_, 4, 4)$

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

Centro de Informática UFPE

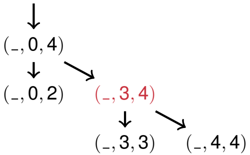
Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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



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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**

  $m \leftarrow \lfloor (l+r)/2 \rfloor$;

  $Mergesort(A, l, m)$;

  $Mergesort(A, m+1, r)$;

  $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

$\downarrow \quad \searrow$

$(\_, 3, 3) \quad (\_, 4, 4)$

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

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

$\downarrow$

$(\_, 0, 4)$

$\downarrow \quad \searrow$

$(\_, 0, 2) \quad (\_, 3, 4)$

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

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	7	0
				l	r
temp:				7	0
				i1	i2

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓
$(\_, 0, 4)$
↓ ↘
$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	7	0
				l	r
				c	
temp:				7	0
				i1	i2

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m+1$ **then**
        $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	0
				l	r
				c	
temp:				7	0
				i1	

$i2 = 5$

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	0
				l	r
					c
temp:				7	0
				i1	

$i2 = 5$

Centro de Informática UFPE
Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$
$(\_, 0, 4)$
$\downarrow \quad \searrow$
$(\_, 0, 2) \quad (\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	7
				l	r
					c
temp:				7	0
					i1

$i2 = 5$

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m+1$ **then**
        $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

↓ ↘

$(\_, 0, 2)$ $(\_, 3, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	7
				l	r
temp:				7	0
					i1

$c = 5$

$i2 = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
- $m \leftarrow \lfloor (l + r)/2 \rfloor$;
- $Mergesort(A, l, m)$;
- $Mergesort(A, m + 1, r)$;
- $Merge(A, l, r)$;

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

↓

$(\_, 0, 4)$

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	7
	l				r
temp:	1	2	5	0	7
	i1			i2	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
   $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	1	2	5	0	7
	l				r
	c				
temp:	1	2	5	0	7
	i1			i2	

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
   $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

$\downarrow$

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	2	5	0	7
	l				r
	c				
temp:	1	2	5	0	7
	i1				i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	2	5	0	7
	l				r
		c			
temp:	1	2	5	0	7
	i1				i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	5	0	7
	l				r
		c			
temp:	1	2	5	0	7
		i1			i2

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	5	0	7
	l				r
			c		
temp:	1	2	5	0	7
		i1			i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
    if $i1 = m + 1$ **then**
        $A[curr] \leftarrow temp[i2{+}{+}]$;
    else if $i2 > r$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else if $temp[i1] \leq temp[i2]$ **then**
        $A[curr] \leftarrow temp[i1{+}{+}]$;
    else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	0	7
	l				r
			c		
temp:	1	2	5	0	7
			i1		i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
   $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
   $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	0	7
	l				r
				c	
temp:	1	2	5	0	7
			i1		i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l+r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m+1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m+1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	5	7
	l				r
				c	
temp:	1	2	5	0	7
				i1	i2

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	5	7
	l				r
					c
temp:	1	2	5	0	7
				i1	i2

Mergesort

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

for $i \leftarrow l$ **to** r **do** $temp[i] \leftarrow A[i]$;
$m \leftarrow \lfloor (l + r)/2 \rfloor$;
$i1 \leftarrow l$; $i2 \leftarrow m + 1$;
for $curr \leftarrow l$ **to** r **do**
  if $i1 = m + 1$ **then**
    $A[curr] \leftarrow temp[i2{+}{+}]$;
  else if $i2 > r$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else if $temp[i1] \leq temp[i2]$ **then**
    $A[curr] \leftarrow temp[i1{+}{+}]$;
  else $A[curr] \leftarrow temp[i2{+}{+}]$;

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	5	7
	l				r
					c
temp:	1	2	5	0	7
				i1	

$i2 = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

```
for i ← l to r do temp[i] ← A[i];
m ← ⌊(l + r)/2⌋;
i1 ← l; i2 ← m + 1;
for curr ← l to r do
    if i1 = m + 1 then
        A[curr] ← temp[i2++];
    else if i2 > r then
        A[curr] ← temp[i1++];
    else if temp[i1] ≤ temp[i2] then
        A[curr] ← temp[i1++];
    else A[curr] ← temp[i2++];
```

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

↓

$(\_, 0, 4)$

index:	0	1	2	3	4
A:	0	1	2	5	7
	l				r
temp:	1	2	5	0	7
				i1	

$i2 = 5$

$c = 5$

Centro de Informática UFPE

Universidade Federal de Pernambuco

Mergesort

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

if $l < r$ **then**
  $m \leftarrow \lfloor (l + r)/2 \rfloor$;
  $Mergesort(A, l, m)$;
  $Mergesort(A, m + 1, r)$;
  $Merge(A, l, r)$;

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

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

Centro de Informática UFPE
Universidade Federal de Pernambuco

Divide-and-conquer

Complexity (in general terms):

- $C(n) = aC(n/b) + f(n)$, where $a \geq 1$, $b \geq 1$, and $a \leq b$
- n = size of the problem instance
- b = number of sub-instances
- a = number of sub-instances that need to be solved
- $f(n)$ = dividing and combining effort

Master theorem: if $f(n) \in \Theta(n^d)$ holds, where $d \geq 0$, then:

$$C(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

The same applies for O and Ω

Decrease by a constant factor: $a = 1$

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

Assuming $n = 2^k$

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

Worst case: when it is necessary to interleave both portions of A

Dividing and combining effort: $C_{merge}(n) = (n - 1)$

- Basic operation: number of key comparisons

Therefore

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

Centro de Informática UFPE

Universidade Federal de Pernambuco

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

Considering the master theorem

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

Pros and cons

- $\Uparrow$: stability
- $\Downarrow$: space efficiency

Complexity of sorting algorithms[2]

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)

Centro de Informática UFPE

Universidade Federal de Pernambuco

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

Agenda

1 Selection sort (brute force)

2 Insertion sort (decrease-and-conquer)

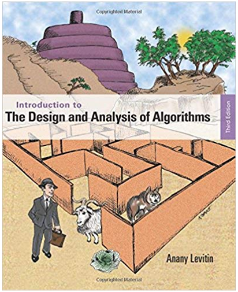
3 Mergesort (divide-and-conquer)

4 Bibliography

Centro de Informática UFPE

Universidade Federal de Pernambuco

Bibliography



Chapter 3 (pp. 98–102)
Chapter 4 (pp. 131–136)
Chapter 5 (pp. 169–175)
Anany Levitin.
Introduction to the Design and Analysis of Algorithms.
3rd edition. Pearson. 2011.

Chapter 7 (pp. 231–235,237–239, 241–244) Clifford Shaffer.
Data Structures and Algorithm Analysis.
Dover, 2013.

Centro de Informática
UFPE

UNIVERSIDADE FEDERAL DE PERNAMBUCO

SELECTION SORT | INSERTION SORT | MERGESORT

Gustavo Carvalho
(ghpc@cin.ufpe.br)

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