



THE UNIVERSITY OF
MELBOURNE

COMP90038

Algorithms and Complexity

Lecture 4: Analysis of Algorithms
(with thanks to Harald Søndergaard)

Toby Murray



toby.murray@unimelb.edu.au



DMD 8.17 (Level 8, Doug McDonnell Bldg)



<http://people.eng.unimelb.edu.au/tobym>



@tobycmurray

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n

- Measure **operation**

- **Time**
bas

- How

Problem	Size Measure	Basic Operation
Search in a list of n items	n	Key comparison
Multiply two matrices of floats	Matrix size (rows x columns)	Float multiplication
Compute a^n	$\log n$	Float multiplication
Graph problem	Number of nodes and edges	Visiting a node

of

- Asymptotic growth rate
- $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n

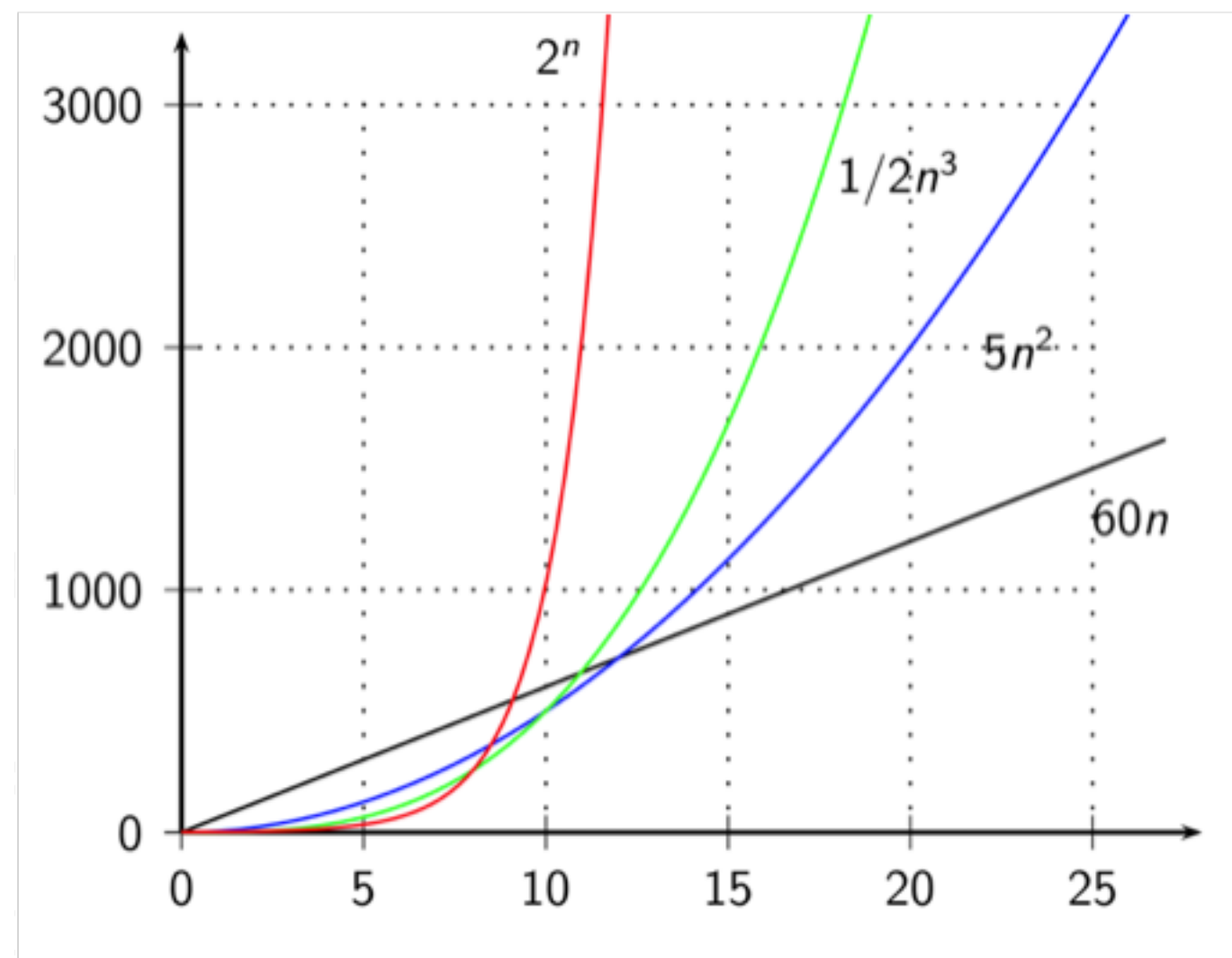
- Measure **operation**

- **Time complexity**

- How to compare

- Asymptotic

- $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$



basic

number of

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

$$f(n) < g(n) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n

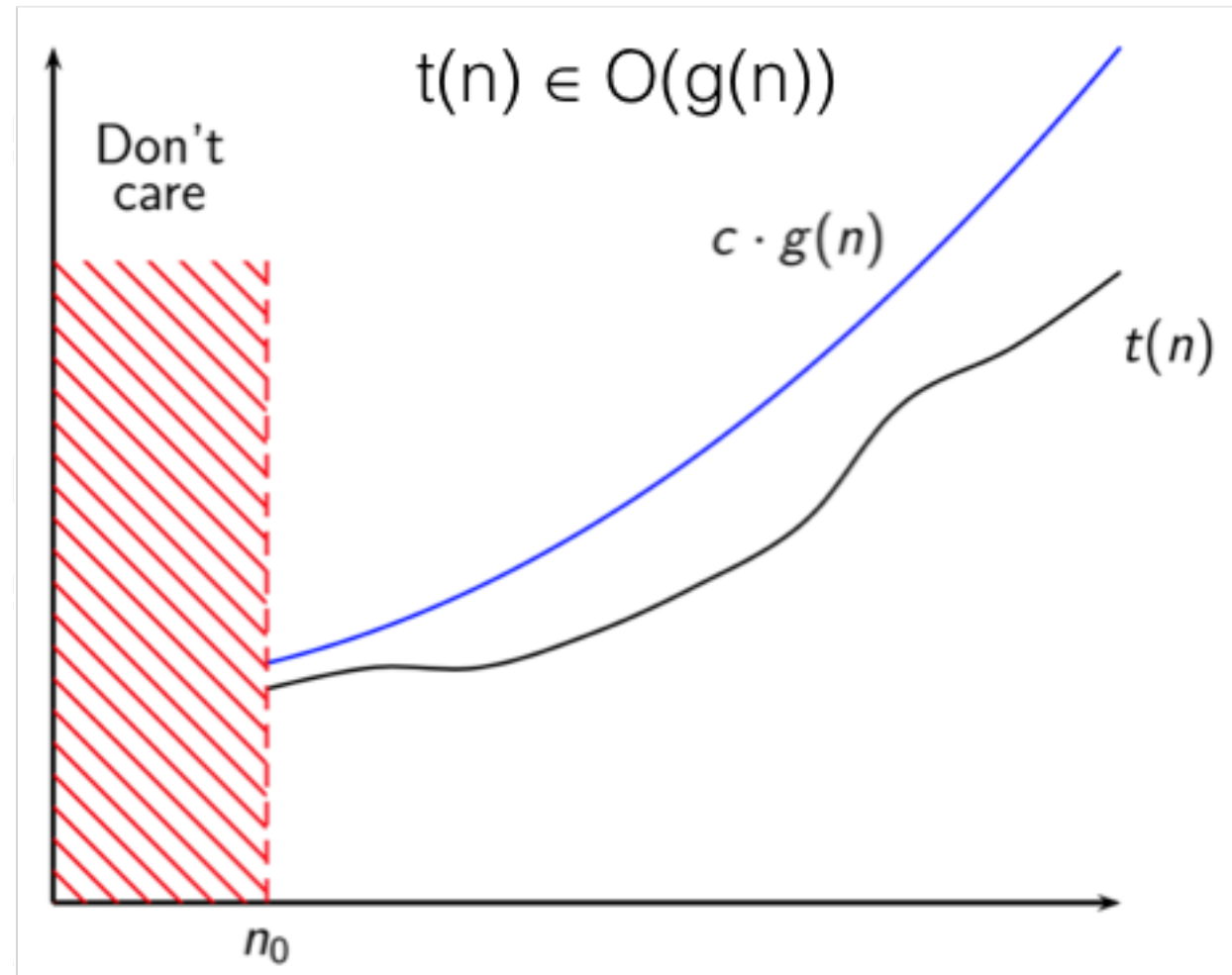
- Measure **operation**

- **Time complexity**

- How to **compare**

- Asymptotic

- $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$



basic

number of

$$f(n) \in o(g(n)) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

$$f(n) < g(n) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n

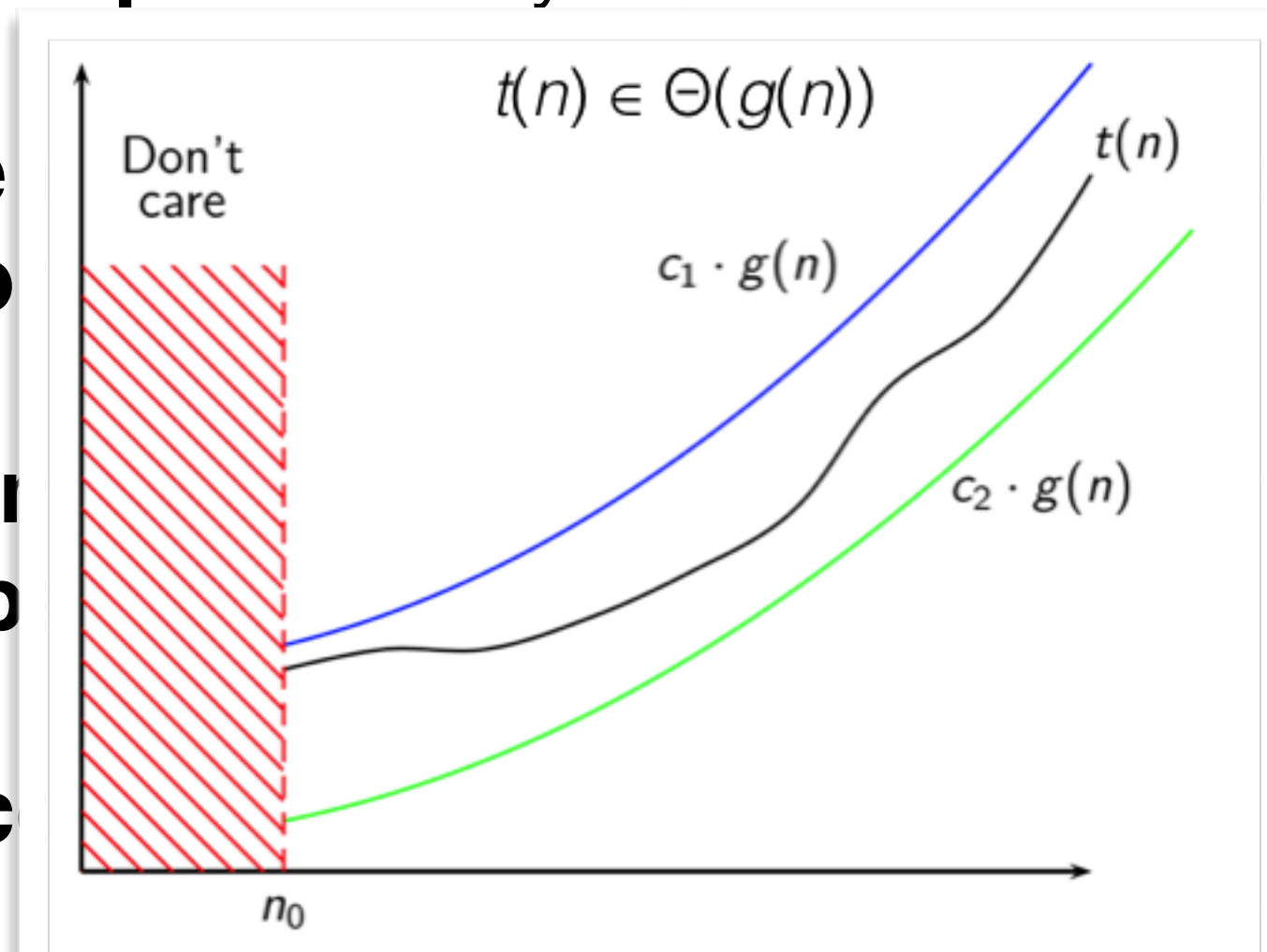
- Measure **operation**

- **Time complexity**

- How to compare

- Asymptotic growth rate

- $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$



basic

number of

$$f(n) = o(g(n)) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

$$f(n) < g(n) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n

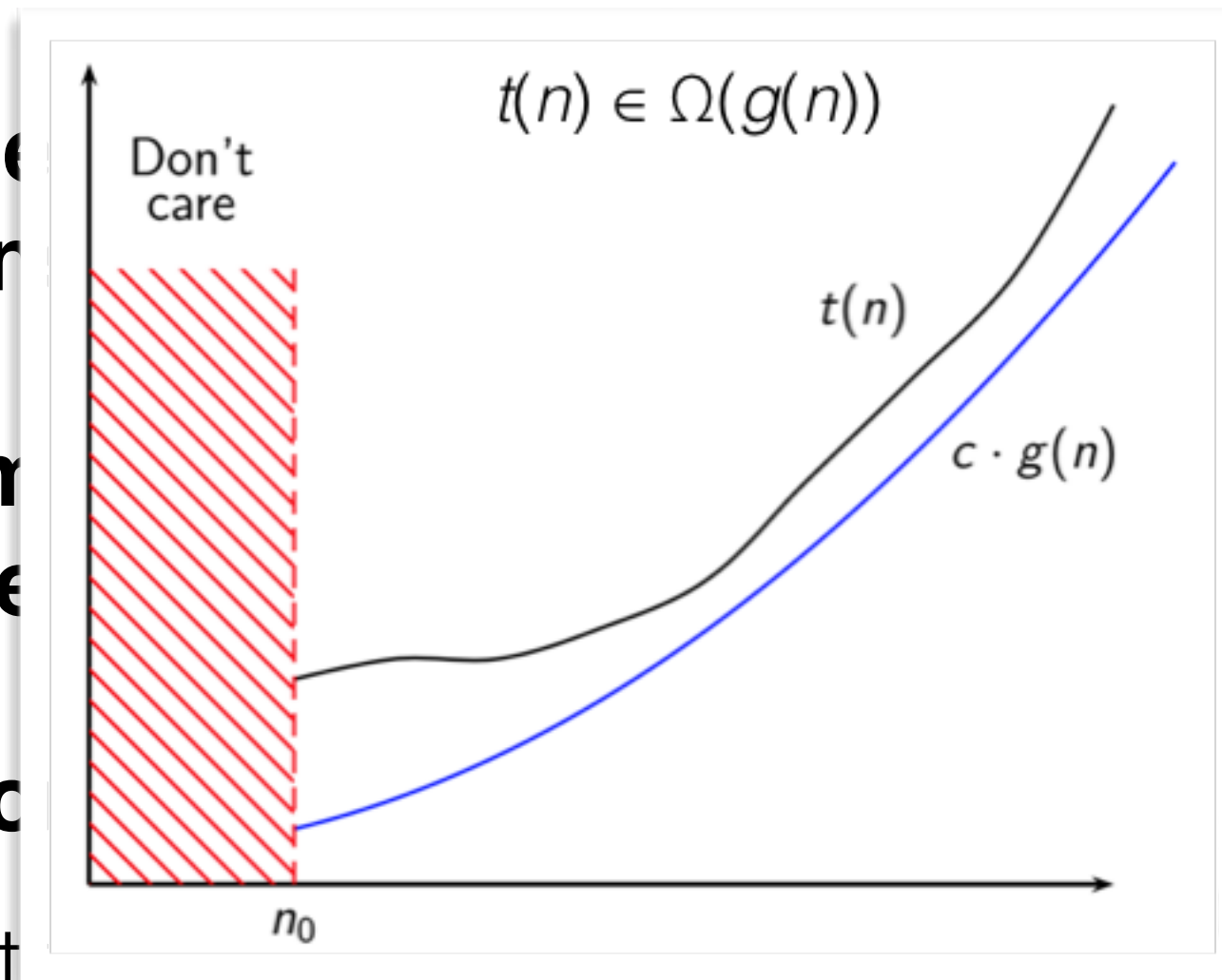
- Measure **operation**

- **Time complexity**

- How to **compare**

- Asymptotic growth rate

- $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$



basic

number of

$$f(n) \in o(g(n)) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Last Time: Time Complexity



THE UNIVERSITY OF
MELBOURNE

- Measure **input size** by natural number n
- Measure **execution time** as number of **basic operations** performed
- **Time complexity** $t(n)$ for an algorithm: number of **basic operations** as a function of n
- How to **compare** different $t(n)$?
 - Asymptotic growth rate
 - $O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

$$f(n) < g(n) \text{ iff } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Establishing Growth Rate

- In the last lecture we proved $t(n) \in O(g(n))$ for some cases of t and g , using the definition of O directly:

$$n > n_0 \Rightarrow t(n) < c \cdot g(n) \quad \text{for some } c \text{ and } n_0.$$

- A more common approach uses

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f \text{ grows asymptotically slower than } g \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f \text{ grows asymptotically faster than } g \end{cases}$$

- Use this to show that $1000n \in O(n^2)$

$$1000n \in O(n^2)$$



$$\lim_{n \rightarrow \infty} \frac{1000n}{n^2}$$

$$1000n \in O(n^2)$$



$$\lim_{n \rightarrow \infty} \frac{1000\cancel{n}}{n^2}$$

$$1000n \in O(n^2)$$



$$\lim_{n \rightarrow \infty} \frac{1000n}{n^2}$$

$$1000n \in O(n^2)$$



$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{1000n}{n^2} \\ &= \lim_{n \rightarrow \infty} \frac{1000}{n} \end{aligned}$$

$$1000n \in O(n^2)$$



$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{1000\cancel{n}}{\cancel{n}n^2} \\ &= \lim_{n \rightarrow \infty} \frac{1000}{n} \\ &= 0 \end{aligned}$$

$$1000n \in O(n^2)$$

$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{1000\cancel{n}}{\cancel{n}n^2} \\ &= \lim_{n \rightarrow \infty} \frac{1000}{n} \\ &= 0 \end{aligned}$$

So $1000n$ grows asymptotically slower than n^2

$$1000n \in O(n^2)$$

$$\begin{aligned} & \lim_{n \rightarrow \infty} \frac{1000\cancel{n}}{\cancel{n}n} \\ &= \lim_{n \rightarrow \infty} \frac{1000}{n} \\ &= 0 \end{aligned}$$

So $1000n$ grows asymptotically slower than n^2

Thus $1000n \in O(n^2)$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f \text{ grows asymptotically slower than } g \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f \text{ grows asymptotically faster than } g \end{cases}$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f \text{ grows asymptotically slower than } g \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f \text{ grows asymptotically faster than } g \end{cases}$$

$$f(n) \in O(g(n))$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f \text{ grows asymptotically faster than } g \end{cases}$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f \text{ grows asymptotically faster than } g \end{cases}$$

$$f(n) \in \Omega(g(n))$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f(n) \in \Omega(g(n)) \end{cases}$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c & f \text{ and } g \text{ have same order of growth} \\ \infty & f(n) \in \Omega(g(n)) \end{cases}$$

$$f(n) \in \Theta(g(n))$$

O, Ω, Θ

What this tells us about how $f(n)$ relates to
 $O(g(n))$, $\Omega(g(n))$ and $\Theta(g(n))$

$$\bullet \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c & f(n) \in \Theta(g(n)) \\ \infty & f(n) \in \Omega(g(n)) \end{cases}$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}}$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{(\log_e 2) \frac{1}{n}}{\frac{1}{2\sqrt{n}}}$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{(\log_e 2) \frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \left((\log_e 2) \frac{1}{n} \cdot 2\sqrt{n} \right)$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{(\log_e 2)^{\frac{1}{n}}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \left((\log_e 2)^{\frac{1}{n}} \cdot 2\sqrt{n} \right)$$

$$= 2 \log_e 2 \lim_{n \rightarrow \infty} \frac{\sqrt{n}}{n}$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{(\log_e 2)^{\frac{1}{n}}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \left((\log_e 2)^{\frac{1}{n}} \cdot 2\sqrt{n} \right)$$

$$= 2 \log_e 2 \lim_{n \rightarrow \infty} \frac{\sqrt{n}}{n} = 2 \log_e 2 \lim_{n \rightarrow \infty} \frac{1}{\sqrt{n}}$$

L'Hôpital's Rule

$$\lim_{h \rightarrow \infty} \frac{t(n)}{g(n)} = \lim_{h \rightarrow \infty} \frac{t'(n)}{g'(n)}$$

where t' and g' are the derivatives of t and g

- For example, show that $\log_2 n$ grows slower than \sqrt{n}

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{(\log_e 2)^{\frac{1}{n}}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \left((\log_e 2)^{\frac{1}{n}} \cdot 2\sqrt{n} \right)$$

$$= 2 \log_e 2 \lim_{n \rightarrow \infty} \frac{\sqrt{n}}{n} = 2 \log_e 2 \lim_{n \rightarrow \infty} \frac{1}{\sqrt{n}} = 0$$

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

max: 23

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 23

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 23

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 42

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 42

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 42

Example:

Finding Largest Element in an Array



function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 69

Example:

Finding Largest Element in an Array



function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 69

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

A:

23	12	42	6	69	18	3
0	1	2	3	4	5	6

↑
 $A[i]$

max: 69

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: comparison “ $A[i] > max$ ”

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: comparison “ $A[i] > max$ ”

Count the number of basic operations executed
for an array of size n :

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: comparison “ $A[i] > max$ ”

Count the number of basic operations executed
for an array of size n :

1

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: comparison “ $A[i] > max$ ”

Count the number of basic operations executed
for an array of size n :

$$C(n) = \sum_{i=1}^{n-1} 1$$

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: comparison “ $A[i] > max$ ”

Count the number of basic operations executed
for an array of size n :

$$C(n) = \sum_{i=1}^{n-1} 1 = ((n - 1) - 1 + 1)$$

Example:

Finding Largest Element in an Array



function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: c

Count the number of basic
for an array of size n :

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \dots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$C(n) = \sum_{i=1}^{n-1} 1 = ((n - 1) - 1 + 1)$$

Example:

Finding Largest Element in an Array



THE UNIVERSITY OF
MELBOURNE

function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: c

Count the number of basic
for an array of size n :

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \dots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$C(n) = \sum_{i=1}^{n-1} 1 = ((n - 1) - 1 + 1) = n - 1$$

Example:

Finding Largest Element in an Array



function MAXELEMENT($A[\cdot]$, n)

$max \leftarrow A[0]$

for $i \leftarrow 1$ to $n - 1$ **do**

if $A[i] > max$ **then**

$max \leftarrow A[i]$

return max

(where n is length of
the array)

Size of input, n :
length of the array

Basic operation: c

Count the number of basic
for an array of size n :

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \dots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

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

Example: Selection Sort



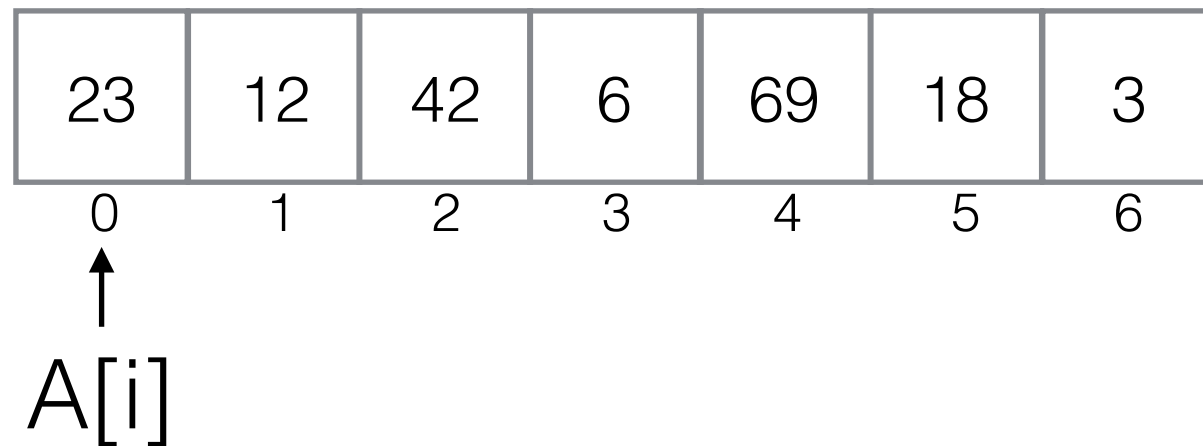
```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  for  $i \leftarrow 0$  to  $n - 2$  do  
     $min \leftarrow i$   
    for  $j \leftarrow i + 1$  to  $n - 1$  do  
      if  $A[j] < A[min]$  then  
         $min \leftarrow j$   
    swap  $A[i]$  and  $A[min]$ 
```

23	12	42	6	69	18	3
0	1	2	3	4	5	6

Example: Selection Sort



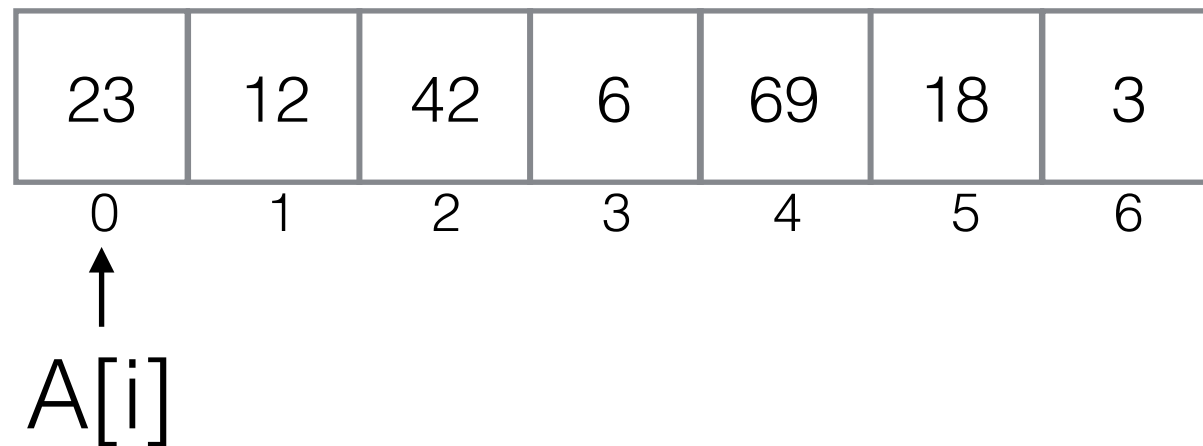
```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  for  $i \leftarrow 0$  to  $n - 2$  do  
     $min \leftarrow i$   
    for  $j \leftarrow i + 1$  to  $n - 1$  do  
      if  $A[j] < A[min]$  then  
         $min \leftarrow j$   
    swap  $A[i]$  and  $A[min]$ 
```



Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



min: 0

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

23	12	42	6	69	18	3
0	1	2	3	4	5	6
↑	↑					
$A[i]$	$A[j]$					

min: 0

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

23	12	42	6	69	18	3
0	1	2	3	4	5	6
↑	↑					
$A[i]$	$A[j]$					

min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

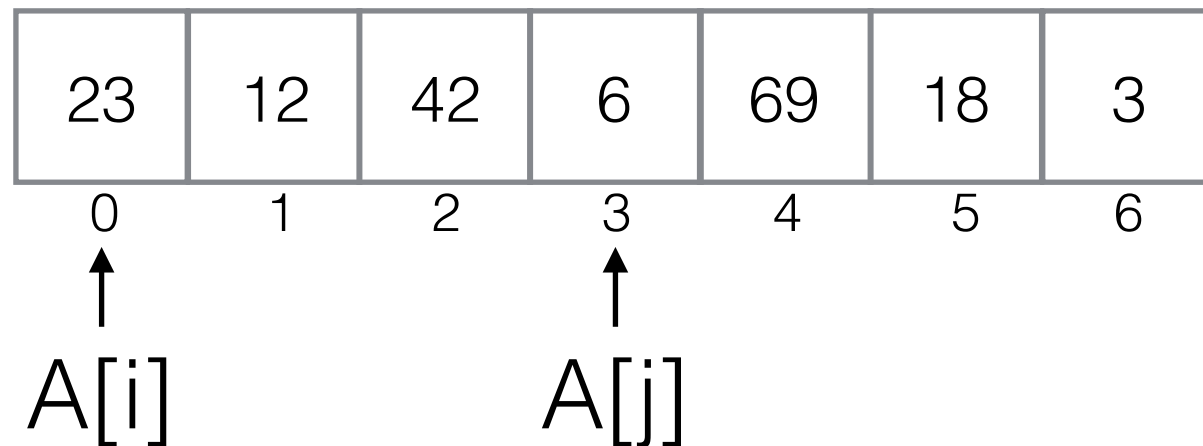
23	12	42	6	69	18	3
0	1	2	3	4	5	6
↑		↑				
$A[i]$		$A[j]$				

min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

23	12	42	6	69	18	3
0	1	2	3	4	5	6
\uparrow			\uparrow			
$A[i]$			$A[j]$			

min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

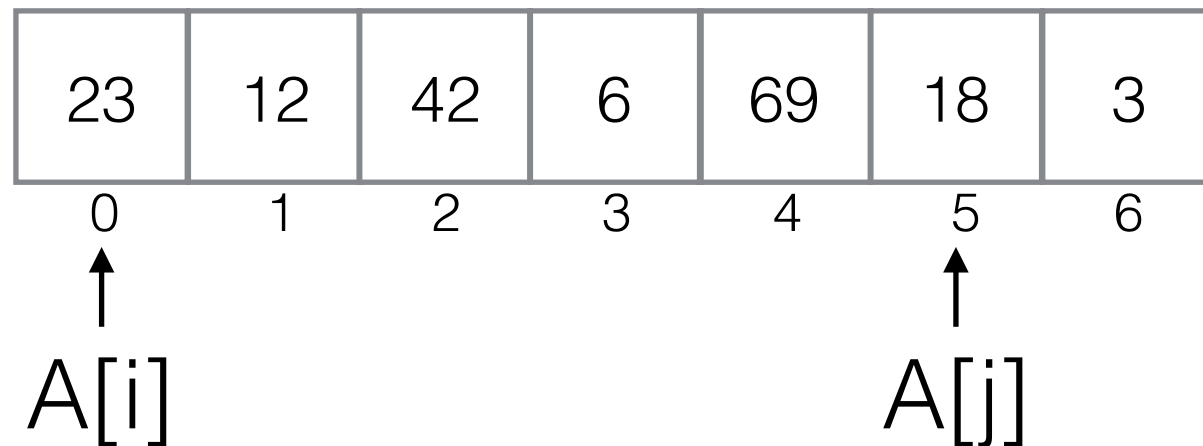
23	12	42	6	69	18	3
0	1	2	3	4	5	6
↑				↑		
$A[i]$				$A[j]$		

min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

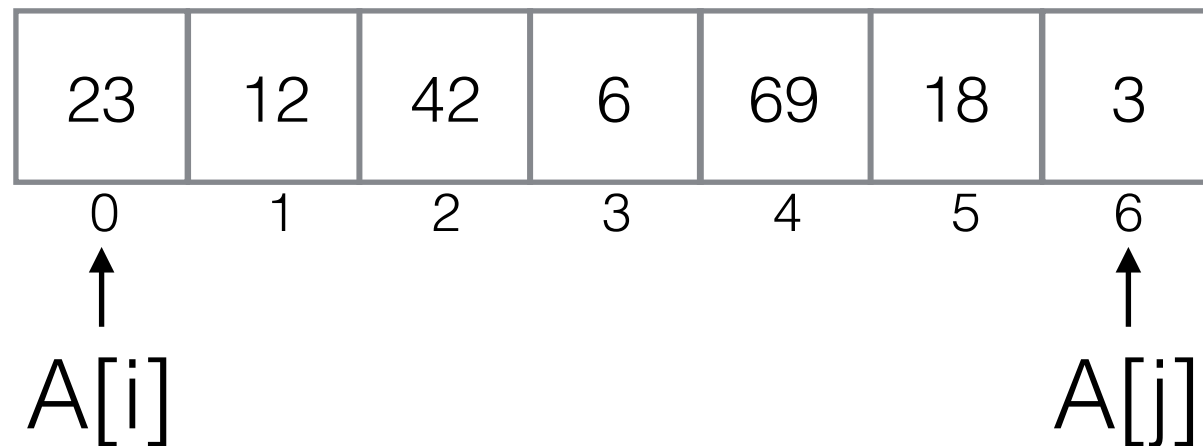


min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

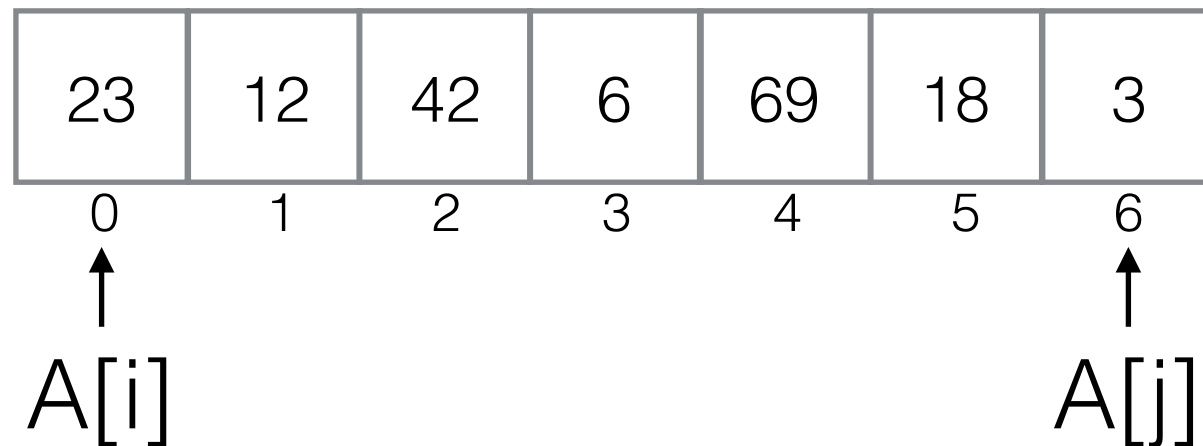


min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



min: 6

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

3	12	42	6	69	18	23
0	1	2	3	4	5	6

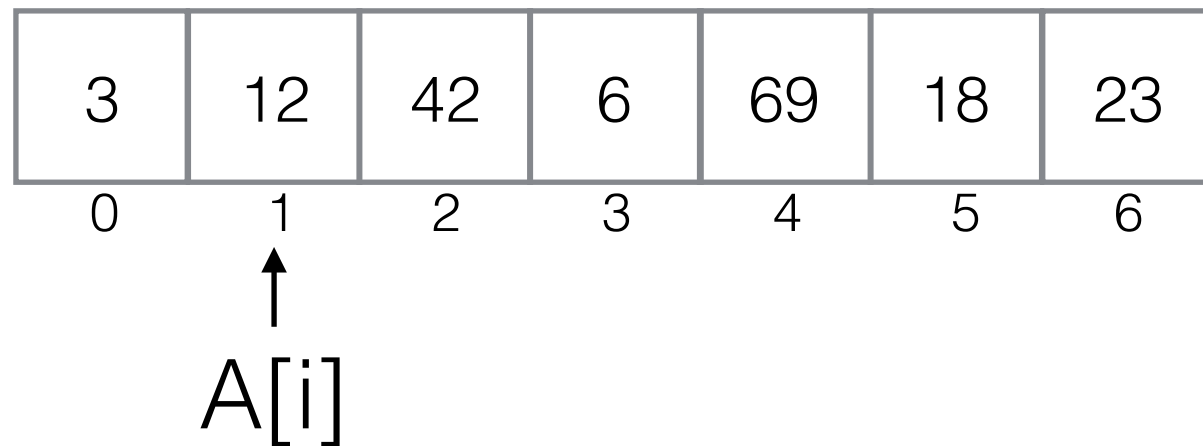
↑
 $A[i]$

min: 6

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

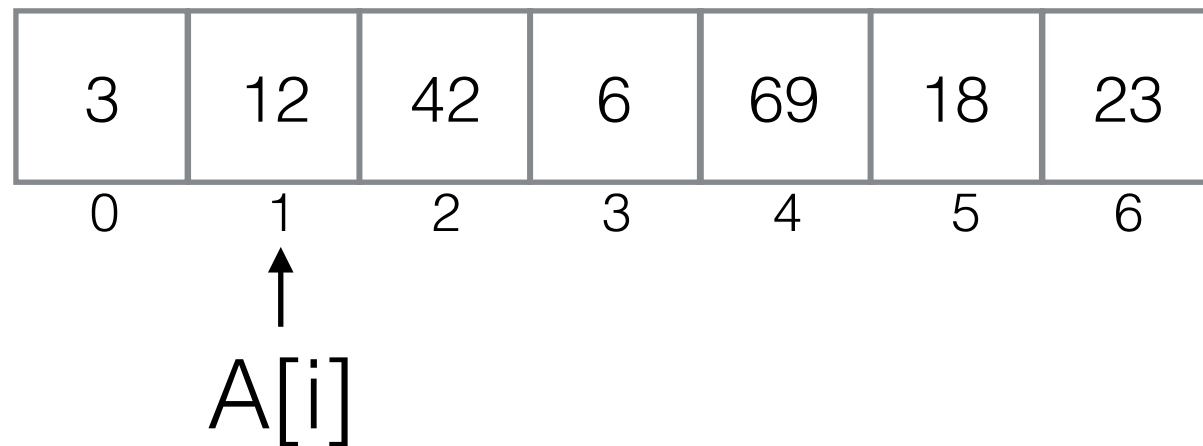


min: 6

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

3	12	42	6	69	18	23
0	1	2	3	4	5	6

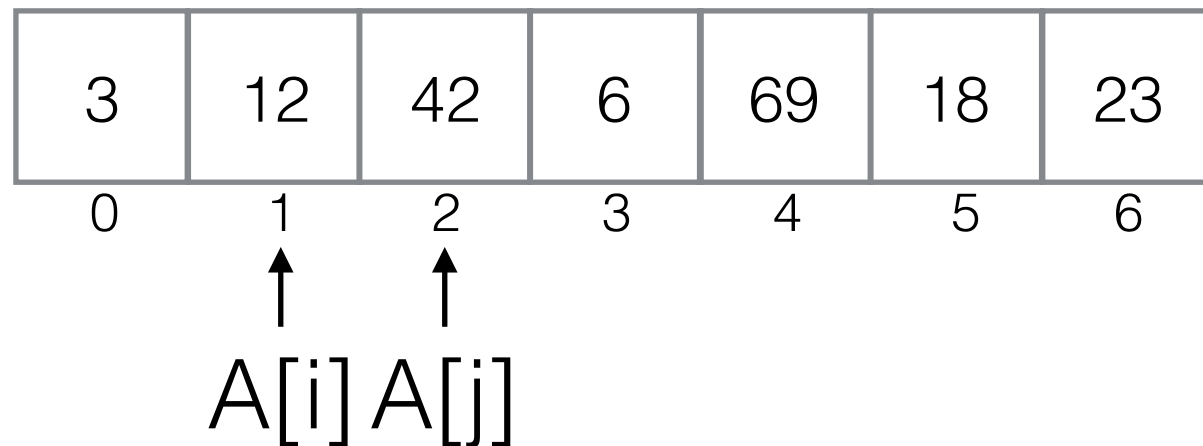
\uparrow
 $A[i]$

min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

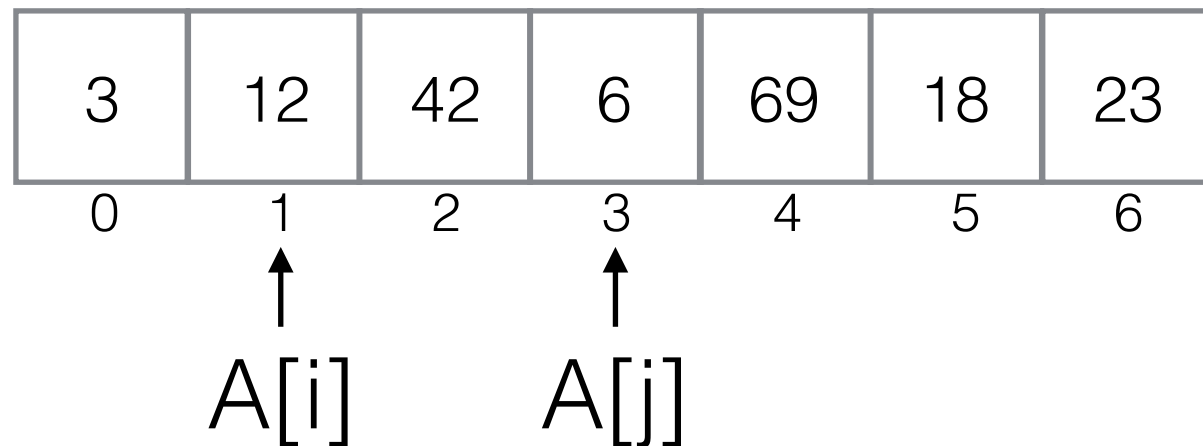


min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



min: 1

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

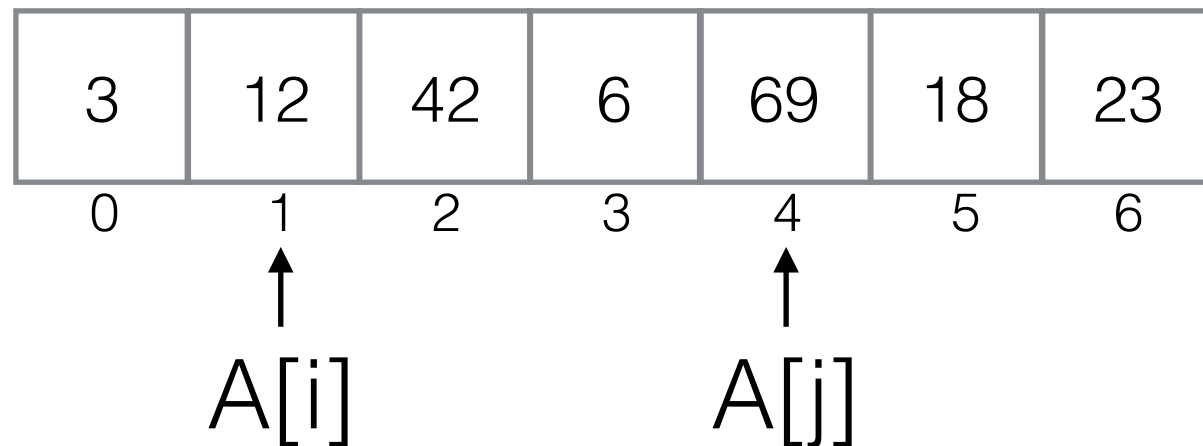
3	12	42	6	69	18	23
0	1	2	3	4	5	6
	↑		↑			
	$A[i]$		$A[j]$			

min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

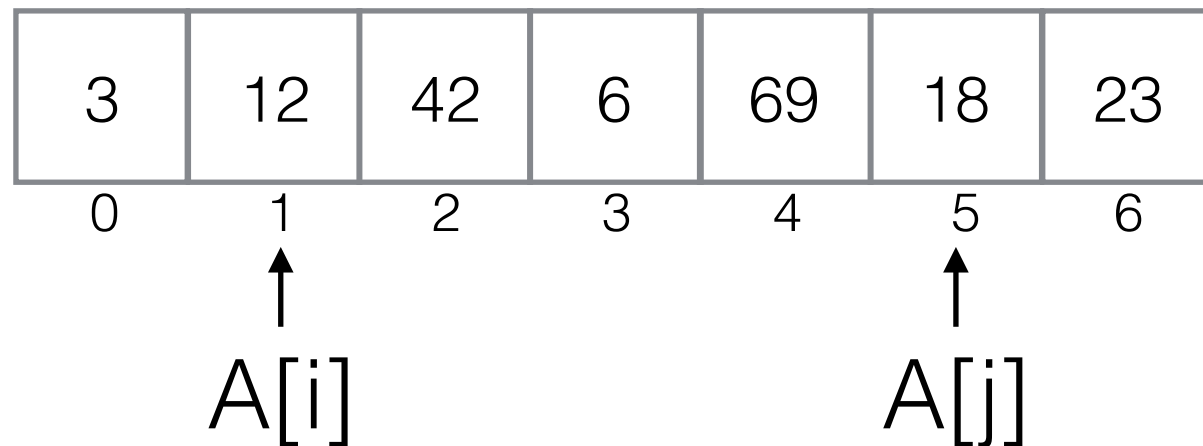


min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

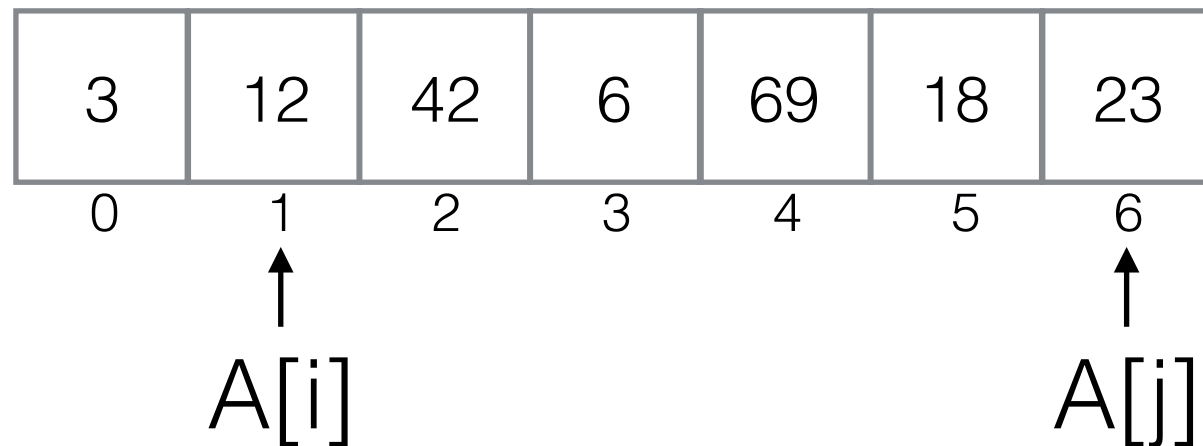


min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

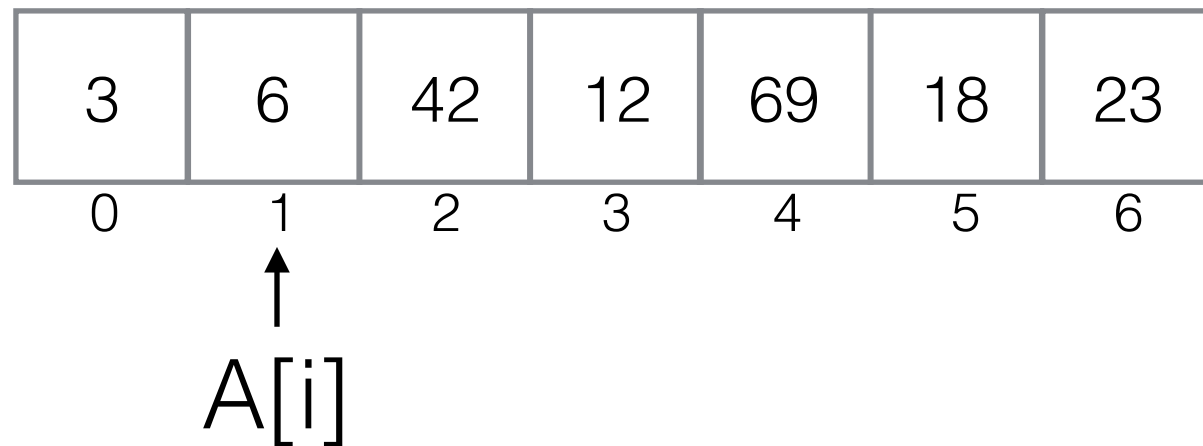


min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```



min: 3

Example: Selection Sort



```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

3	6	42	12	69	18	23
0	1	2	3	4	5	6

↑
 $A[i]$

min: 3

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

$C(n) =$

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n - 1 - i)$$

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n - 1 - i) = \sum_{i=0}^{n-2} (n - 1) - \sum_{i=0}^{n-2} i$$

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

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

Example: Selection Sort

```
function SELSORT( $A[\cdot]$ ,  $n$ )  
  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  $A[min]$ 
```

Input size n : length
of the array

Basic operation:
comparison $A[j] < A[min]$

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

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix}$$

A

$$\begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix}$$

B

$$\begin{bmatrix} & \\ & \end{bmatrix}$$

C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} \begin{bmatrix} & \\ & \end{bmatrix}$$

$A \qquad \qquad B \qquad \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 0$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} \begin{bmatrix} & \\ & \end{bmatrix}$$

$A \qquad \qquad B \qquad \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 0$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} \quad \begin{bmatrix} 0 \end{bmatrix}$$

$A \qquad \qquad B \qquad \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 0$

$k: 0$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} \quad \begin{bmatrix} 0 \end{bmatrix}$$

$A \qquad \qquad B \qquad \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 0$

$k: 0$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 40 \end{bmatrix}$$

$A \qquad B \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 0$

$k: 1$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 40 \end{bmatrix}$$

$A \qquad B \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

i: 0

j: 0

k: 1

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 103 \end{bmatrix}$$

A B C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 1$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 103 \end{bmatrix}$$

$A \qquad B \qquad C$

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

i: 0

j: 1

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 103 & 0 \end{bmatrix}$$

A B C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

i: 0

j: 1

k: 0

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix}$$

A

$$\begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix}$$

B

$$\begin{bmatrix} 103 & 0 \end{bmatrix}$$

C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

i: 0

j: 1

k: 0

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix}$$

A

$$\begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix}$$

B

$$\begin{bmatrix} 103 & 10 \end{bmatrix}$$

C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

i: 0

j: 1

k: 1

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 103 & 10 \end{bmatrix}$$

A B C

Example: Matrix Multiplication



```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

$i: 0$

$j: 1$

$k: 1$

$$\begin{bmatrix} 5 & 7 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 8 & 2 \\ 9 & 6 \end{bmatrix} = \begin{bmatrix} 103 & 52 \end{bmatrix}$$

$A \qquad B \qquad C$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

$$M(n) =$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

$$M(n) = 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

$$M(n) = \sum_{k=0}^{n-1} 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

$$M(n) = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

```
function MATRIXMULT( $A[\cdot, \cdot]$ ,  $B[\cdot, \cdot]$ ,  $n$ )    ▷ For  $n \times n$  matrices
  for  $i \leftarrow 0$  to  $n - 1$  do
    for  $j \leftarrow 0$  to  $n - 1$  do
       $C[i, j] \leftarrow 0.0$ 
      for  $k \leftarrow 0$  to  $n - 1$  do
         $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$ 
  return  $C$ 
```

Basic operation: multiplication $A[i, k] * B[k, j]$

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

Example: Matrix Multiplication



THE UNIVERSITY OF
MELBOURNE

$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1)$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n - 1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

$$\sum_{i=l}^u ca_i = c \sum_{i=l}^u a_i$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

$$= \sum_{i=0}^{n-1} \left(n \cdot \sum_{j=0}^{n-1} 1 \right)$$

$$\sum_{i=l}^u ca_i = c \sum_{i=l}^u a_i$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

$$= \sum_{i=0}^{n-1} \left(n \cdot \sum_{j=0}^{n-1} 1 \right) = \sum_{i=0}^{n-1} n^2$$

$$\sum_{i=l}^u ca_i = c \sum_{i=l}^u a_i$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \cdots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

$$= \sum_{i=0}^{n-1} \left(n \cdot \sum_{j=0}^{n-1} 1 \right) = \sum_{i=0}^{n-1} n^2$$

$$= n^3$$

$$\sum_{i=l}^u c a_i = c \sum_{i=l}^u a_i$$

Example: Matrix Multiplication



$$M(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$\sum_{i=l}^u 1 = \underbrace{1 + 1 + \dots + 1}_{u-l+1 \text{ times}} = u - l + 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} ((n-1) - 0 + 1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (n \cdot 1)$$

$$= \sum_{i=0}^{n-1} \left(n \cdot \sum_{j=0}^{n-1} 1 \right) = \sum_{i=0}^{n-1} n^2$$

$$\sum_{i=l}^u ca_i = c \sum_{i=l}^u a_i$$

$$= n^3$$

$$\in \Theta(n^3)$$

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$F(5)$

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$F(5) = F(4) \cdot 5$$

Analysing Recursive Algorithms

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \end{aligned}$$

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \\ &= ((F(2) \cdot 3) \cdot 4) \cdot 5 \end{aligned}$$

Analysing Recursive Algorithms

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \\ &= ((F(2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((F(1) \cdot 2) \cdot 3) \cdot 4) \cdot 5 \end{aligned}$$

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \\ &= ((F(2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((F(1) \cdot 2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((((F(0) \cdot 1) \cdot 2) \cdot 3) \cdot 4) \cdot 5) \end{aligned}$$

Analysing Recursive Algorithms

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \\ &= ((F(2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((F(1) \cdot 2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((((F(0) \cdot 1) \cdot 2) \cdot 3) \cdot 4) \cdot 5) \\ &= (((((1 \cdot 1) \cdot 2) \cdot 3) \cdot 4) \cdot 5 \end{aligned}$$

Analysing Recursive Algorithms

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

$$\begin{aligned} F(5) &= F(4) \cdot 5 \\ &= (F(3) \cdot 4) \cdot 5 \\ &= ((F(2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((F(1) \cdot 2) \cdot 3) \cdot 4) \cdot 5 \\ &= (((((F(0) \cdot 1) \cdot 2) \cdot 3) \cdot 4) \cdot 5) \\ &= (((((1 \cdot 1) \cdot 2) \cdot 3) \cdot 4) \cdot 5) \\ &= 5! \end{aligned}$$

Analysing Recursive Algorithms



THE UNIVERSITY OF
MELBOURNE

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

Analysing Recursive Algorithms

```
function F( $n$ )  
  if  $n = 0$  then return 1  
  else return F( $n - 1$ ) ·  $n$ 
```

Basic operation:
multiplication

Analysing Recursive Algorithms

```
function  $F(n)$   
  if  $n = 0$  then return 1  
  else return  $F(n - 1) \cdot n$ 
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) =$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) =$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) = 1$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) = \quad + 1$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) = M(n - 1) + 1$$

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) = M(n - 1) + 1$$

Need to express $M(n)$ in **closed form** (i.e. non-recursively)

Analysing Recursive Algorithms

```
function F(n)  
  if n = 0 then return 1  
  else return F(n − 1) · n
```

Basic operation:
multiplication

We express the cost **recursively** (as a **recurrence relation**)

$$M(0) = 0$$

$$M(n) = M(n - 1) + 1$$

Need to express $M(n)$ in **closed form** (i.e. non-recursively)

Try: “**telescoping**” aka “**backward substitution**”

Telescoping (aka Backward Substitution)



THE UNIVERSITY OF
MELBOURNE

$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

Telescoping (aka Backward Substitution)



THE UNIVERSITY OF
MELBOURNE

$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

Telescoping (aka Backward Substitution)



THE UNIVERSITY OF
MELBOURNE

$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$M(n - 1) = M((n - 1) - 1) + 1$$

Telescoping (aka Backward Substitution)



THE UNIVERSITY OF
MELBOURNE

$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$M(n) =$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$M(n) = (M(n - 2) + 1) + 1$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \\ &= n \end{aligned}$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \\ &= n \end{aligned}$$

Closed form:

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \\ &= n \end{aligned}$$

Closed form:

$$M(n) = n$$

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \\ &= n \end{aligned}$$

Closed form:

$$M(n) = n$$

Complexity:

Telescoping (aka Backward Substitution)



$$M(n) = M(n - 1) + 1$$

$$M(0) = 0$$

What is $M(n-1)$?

$$\begin{aligned} M(n - 1) &= M((n - 1) - 1) + 1 \\ &= M(n - 2) + 1 \end{aligned}$$

$$\begin{aligned} M(n) &= (M(n - 2) + 1) + 1 \\ &= M(n - 2) + 2 \\ &= (M(n - 3) + 1) + 2 \\ &= M(n - 3) + 3 \\ &\dots \\ &= M(n - n) + n \\ &= M(0) + n \\ &= n \end{aligned}$$

Closed form:

$$M(n) = n$$

Complexity:

$$M(n) \in \Theta(n)$$

Example:

Binary Search in Sorted Array



```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 0$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

Example:

Binary Search in Sorted Array

function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 0$

$hi: 6$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 0$

$hi: 6$

$key: 41$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 0$

$hi: 6$

$key: 41$

$mid: 3$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 0$

$hi: 6$

$key: 41$

$mid: 3$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 0, 6, 41$)

BinSearch($A, 4, 6, 41$)

Example:

Binary Search in Sorted Array



```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 6, 41$)

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 6, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 6$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 6, 41$)

Example:

Binary Search in Sorted Array



THE UNIVERSITY OF
MELBOURNE

function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

lo : 4

hi : 6

key : 41

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch(A , 4, 6, 41)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

lo : 4

hi : 6

key : 41

mid : 5

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch(A , 4, 6, 41)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 6$

$key: 41$

$mid: 5$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 6, 41$)

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array



```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array

function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 4$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 4$

$key: 41$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array



function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 4$

$key: 41$

$mid: 4$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

Example:

Binary Search in Sorted Array

function BINSEARCH($A[\cdot]$, lo , hi , key)

if $lo > hi$ **then return** -1

$mid \leftarrow lo + (hi - lo)/2$

if $A[mid] = key$ **then return** mid

else

if $A[mid] > key$ **then**

return BINSEARCH(A , lo , $mid - 1$, key)

else return BINSEARCH(A , $mid + 1$, hi , key)

$lo: 4$

$hi: 4$

$key: 41$

$mid: 4$

A:

4	9	13	22	41	83	96
0	1	2	3	4	5	6

BinSearch($A, 4, 4, 41$)

returns 4

Example:

Binary Search in Sorted Array



THE UNIVERSITY OF
MELBOURNE

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Example:

Binary Search in Sorted Array



THE UNIVERSITY OF
MELBOURNE

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Basic operation: key comparison $A[mid] = key$

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Basic operation: key comparison $A[mid] = key$

$C(0) =$

Example:

Binary Search in Sorted Array



THE UNIVERSITY OF
MELBOURNE

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Basic operation: key comparison $A[mid] = key$

$$C(0) = 0$$

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Basic operation: key comparison $A[mid] = key$

$$C(0) = 0 \qquad C(n) = \qquad + 1$$

Example:

Binary Search in Sorted Array

```
function BINSEARCH( $A[\cdot]$ ,  $lo$ ,  $hi$ ,  $key$ )  
  if  $lo > hi$  then return  $-1$   
   $mid \leftarrow lo + (hi - lo)/2$   
  if  $A[mid] = key$  then return  $mid$   
  else  
    if  $A[mid] > key$  then  
      return BINSEARCH( $A$ ,  $lo$ ,  $mid - 1$ ,  $key$ )  
    else return BINSEARCH( $A$ ,  $mid + 1$ ,  $hi$ ,  $key$ )
```

Basic operation: key comparison $A[mid] = key$

$$C(0) = 0 \qquad C(n) = C(n/2) + 1$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$C(n) = C(n/2) + 1$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \\ &\dots \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \\ &\dots \\ &= C(n/n) + \log_2 n \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \\ &\dots \\ &= C(n/n) + \log_2 n \\ &= (C(0) + 1) + \log_2 n \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \\ &\dots \\ &= C(n/n) + \log_2 n \\ &= (C(0) + 1) + \log_2 n \\ &= 1 + \log_2 n \end{aligned}$$

Telescoping

A **smoothness rule** allows us to assume
that n is a power of 2

$$C(0) = 0$$

$$C(n) = C(n/2) + 1$$

$$\begin{aligned} C(n) &= C(n/2) + 1 \\ &= (C(n/4) + 1) + 1 \\ &= ((C(n/8) + 1) + 1) + 1 \\ &\dots \\ &= C(n/n) + \log_2 n \\ &= (C(0) + 1) + \log_2 n \\ &= 1 + \log_2 n \end{aligned}$$

$$C(n) \in \Theta(\log n)$$

Logarithmic Functions Have Same Rate of Growth



In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

So, e.g.:

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

So, e.g.: $O(\log_a n) = O(\log_b n)$

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

So, e.g.: $O(\log_a n) = O(\log_b n)$

Also, since

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

So, e.g.: $O(\log_a n) = O(\log_b n)$

Also, since $\log n^c = c \cdot \log n$

Logarithmic Functions Have Same Rate of Growth

In O , Ω , Θ , expressions we can just write “log” for any logarithmic function no matter what the base is

Asymptotically, all logarithmic behaviour is the same, since

$$\log_a x = (\log_a b)(\log_b x)$$

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{\log_b n} = \lim_{n \rightarrow \infty} \frac{(\log_a b)(\log_b n)}{\log_b n} = (\log_a b) \lim_{n \rightarrow \infty} 1 = \log_a b$$

So, e.g.: $O(\log_a n) = O(\log_b n)$

Also, since $\log n^c = c \cdot \log n$

$$\log n^c \in O(\log n)$$

Back to Euclid



```
function EUCLID( $m, n$ )  
  while  $n \neq 0$  do  
     $r \leftarrow m \bmod n$   
     $m \leftarrow n$   
     $n \leftarrow r$   
  return  $m$ 
```

```
function EUCLID( $m, n$ )  
  if  $n = 0$  then  
    return  $m$   
  return EUCLID( $n, m \bmod n$ )
```

Back to Euclid



```
function EUCLID( $m, n$ )  
  while  $n \neq 0$  do  
     $r \leftarrow m \bmod n$   
     $m \leftarrow n$   
     $n \leftarrow r$   
  return  $m$ 
```

```
function EUCLID( $m, n$ )  
  if  $n = 0$  then  
    return  $m$   
  return EUCLID( $n, m \bmod n$ )
```

Running time is **linear** in size (in bits) of input, i.e.
 $O(\log(m + n))$

Back to Euclid



```
function EUCLID( $m, n$ )  
  while  $n \neq 0$  do  
     $r \leftarrow m \bmod n$   
     $m \leftarrow n$   
     $n \leftarrow r$   
  return  $m$ 
```

```
function EUCLID( $m, n$ )  
  if  $n = 0$  then  
    return  $m$   
  return EUCLID( $n, m \bmod n$ )
```

Running time is **linear** in size (in bits) of input, i.e.

$$O(\log(m + n))$$

since the value of m (and n) is at least halved in every two iterations.

Back to Euclid

```
function EUCLID( $m, n$ )  
  while  $n \neq 0$  do  
     $r \leftarrow m \bmod n$   
     $m \leftarrow n$   
     $n \leftarrow r$   
  return  $m$ 
```

```
function EUCLID( $m, n$ )  
  if  $n = 0$  then  
    return  $m$   
  return EUCLID( $n, m \bmod n$ )
```

Running time is **linear** in size (in bits) of input, i.e.

$$O(\log(m + n))$$

since the value of m (and n) is at least halved in every two iterations.

Why? After two iterations, m becomes $m \bmod n$; also

$$1 < n < m \implies m \bmod n < m/2$$

Summarising Reasoning with Big-Oh



THE UNIVERSITY OF
MELBOURNE

Summarising Reasoning with Big-Oh



THE UNIVERSITY OF
MELBOURNE

Suppose

Summarising Reasoning with Big-Oh



THE UNIVERSITY OF
MELBOURNE

Suppose $t_1(n) \in O(g_1(n))$

$t_2(n) \in O(g_2(n))$

Summarising Reasoning with Big-Oh



THE UNIVERSITY OF
MELBOURNE

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

Summarising Reasoning with Big-Oh



THE UNIVERSITY OF
MELBOURNE

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Summarising Reasoning with Big-Oh



Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

(we can throw away smaller summands)

Summarising Reasoning with Big-Oh

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

(we can throw away smaller summands)

$$c \cdot t_1(n) \in O(g_1(n))$$

Summarising Reasoning with Big-Oh

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

(we can throw away smaller summands)

$$c \cdot t_1(n) \in O(g_1(n))$$

(constants can be thrown away too)

Summarising Reasoning with Big-Oh

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

(we can throw away smaller summands)

$$c \cdot t_1(n) \in O(g_1(n))$$

(constants can be thrown away too)

$$t_1(n) \cdot t_2(n) \in O(g_1(n) \cdot g_2(n))$$

Summarising Reasoning with Big-Oh

Suppose $t_1(n) \in O(g_1(n))$ $t_2(n) \in O(g_2(n))$

Then the following hold:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

(we can throw away smaller summands)

$$c \cdot t_1(n) \in O(g_1(n))$$

(constants can be thrown away too)

$$t_1(n) \cdot t_2(n) \in O(g_1(n) \cdot g_2(n))$$

(for nested loops: count number of times outer loop
is executed, multiply by cost of inner loop)

Some Useful Formulas



From Stirling's formula: $n! \in O(n^{n+\frac{1}{2}})$

Some Useful Formulas

From Stirling's formula: $n! \in O(n^{n+\frac{1}{2}})$

this is not the time complexity of an
algorithm to compute n -factorial

Some Useful Formulas

From Stirling's formula: $n! \in O(n^{n+\frac{1}{2}})$

this is not the time complexity of an
algorithm to compute n -factorial

$$\sum_{i=0}^n i^2 = \frac{n}{3}(n + \frac{1}{2})(n + 1)$$

$$\sum_{i=0}^n (2i + 1) = (n + 1)^2$$

$$\sum_{i=1}^n 1/i = O(\log n)$$

Some Useful Formulas

From Stirling's formula: $n! \in O(n^{n+\frac{1}{2}})$

this is not the time complexity of an
algorithm to compute n -factorial

$$\sum_{i=0}^n i^2 = \frac{n}{3}(n + \frac{1}{2})(n + 1)$$

$$\sum_{i=0}^n (2i + 1) = (n + 1)^2$$

$$\sum_{i=1}^n 1/i = O(\log n)$$

See also Cormen's Appendix A or Levitin's Appendix A.

Levitin's Appendix B is a tutorial on recurrence relations.

The Road Ahead

- You'll get much more familiar with asymptotic analysis as we use it on algorithms we meet in this course.
- Next week we begin our study of algorithms by looking at **brute force** approaches