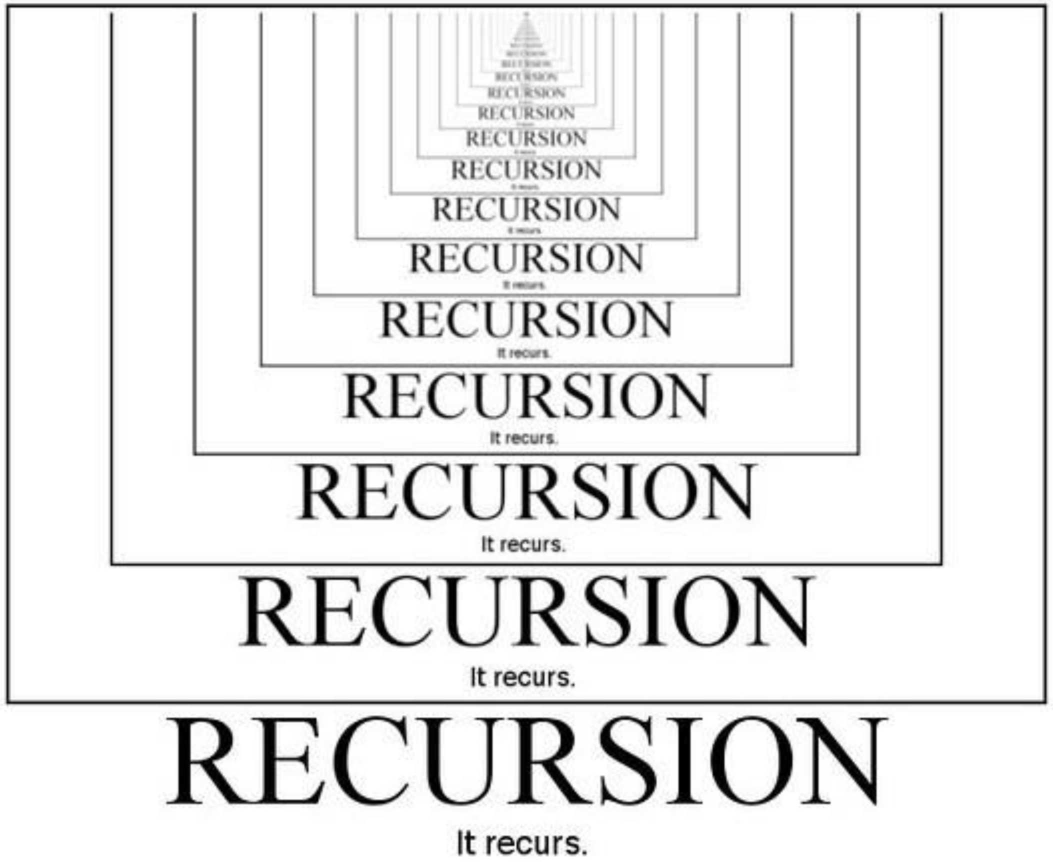
遮迴

課程大綱

1. 遞迴的重要性
2. 經典的遞迴函數
3. 遞迴解還是迭代解

遞迴的重要性

遞迴，一種常見表達解決問題流程的方法



遞迴，一種常見表達解決問題流程的方法

我們會需要遞迴的原因有兩個：

1. 易於理解
2. 易於簡化問題

遞迴，一種常見表達解決問題流程的方法

我們會需要遞迴的原因有兩個：

1. 易於理解
2. 易於簡化問題

首先我們舉個例

數學的階乘 $n!$

其定義為：

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1$$

其中 $0! = 1$

數學的階乘 $n!$

其定義為：

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1$$

其中 $0! = 1$

因為我們知道階乘有某種「重複」的性質

所以能直覺得感覺階乘演算法是遞迴關係

數學的階乘 $n!$

其定義為：

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1$$

其中 $0! = 1$

於是我們開始探討該演算法的遞迴細節

我們想要知道它的終止條件跟遞迴複雜度

數學的階乘 $n!$

其定義為：

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1$$

其中 $0! = 1$

於是我們開始探討該演算法的遞迴細節

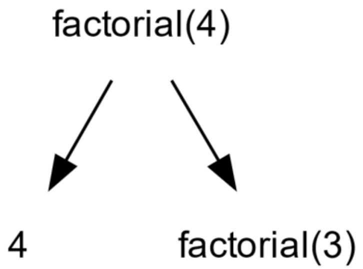
我們想要知道它的終止條件跟遞迴複雜度

為了簡化工人智慧的計算成本，我們以實例探討

數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

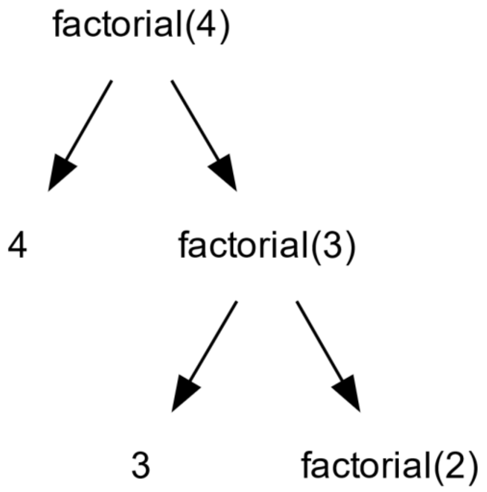


數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$



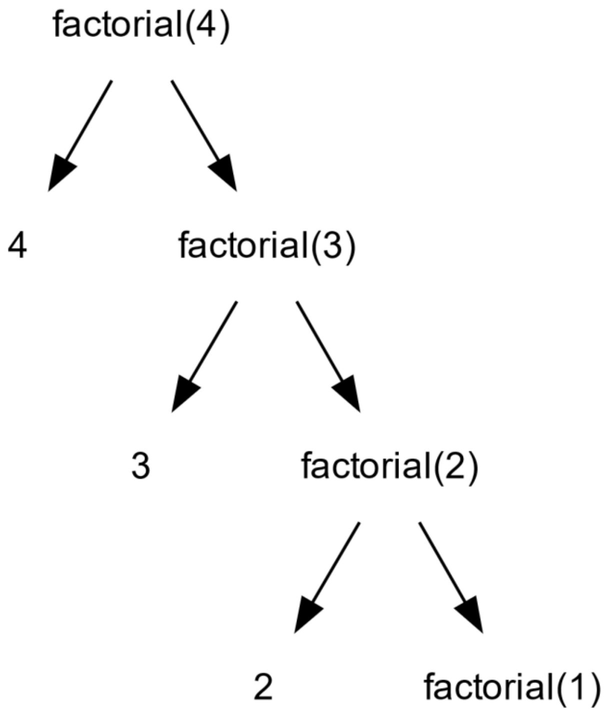
數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$



數學的階乘 4!

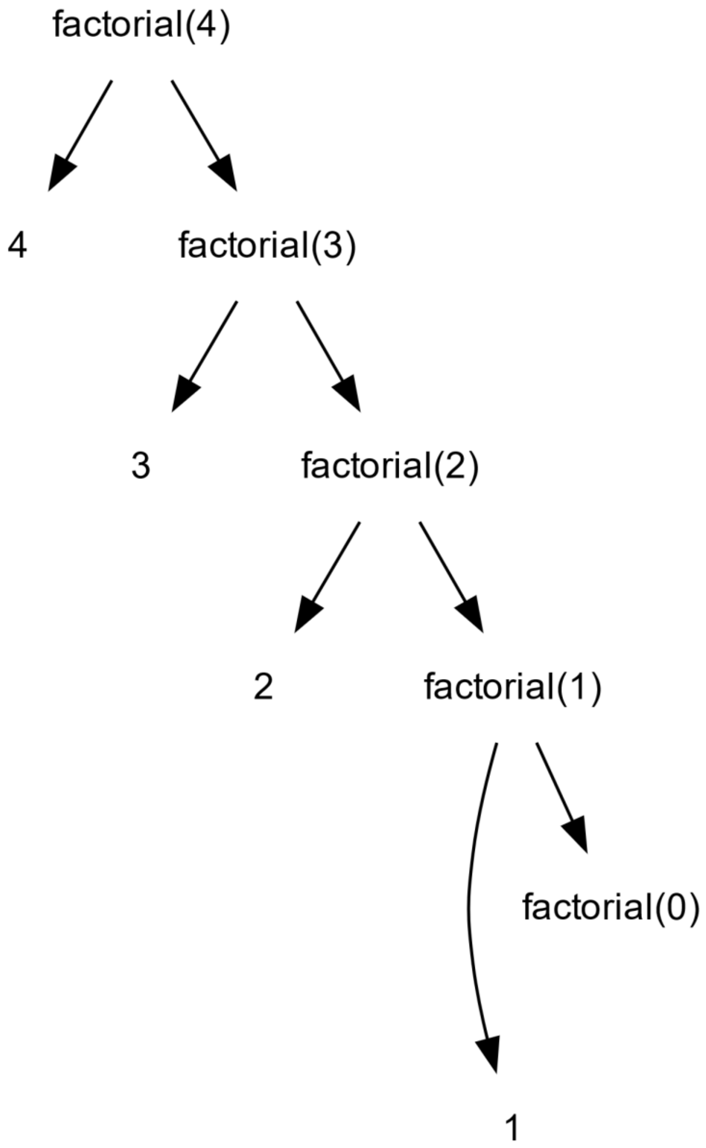
計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$

$= 4 \times 3 \times 2 \times 1 \times 0!$



數學的階乘 4!

計算過程為：

$4! = 4 \times 3!$

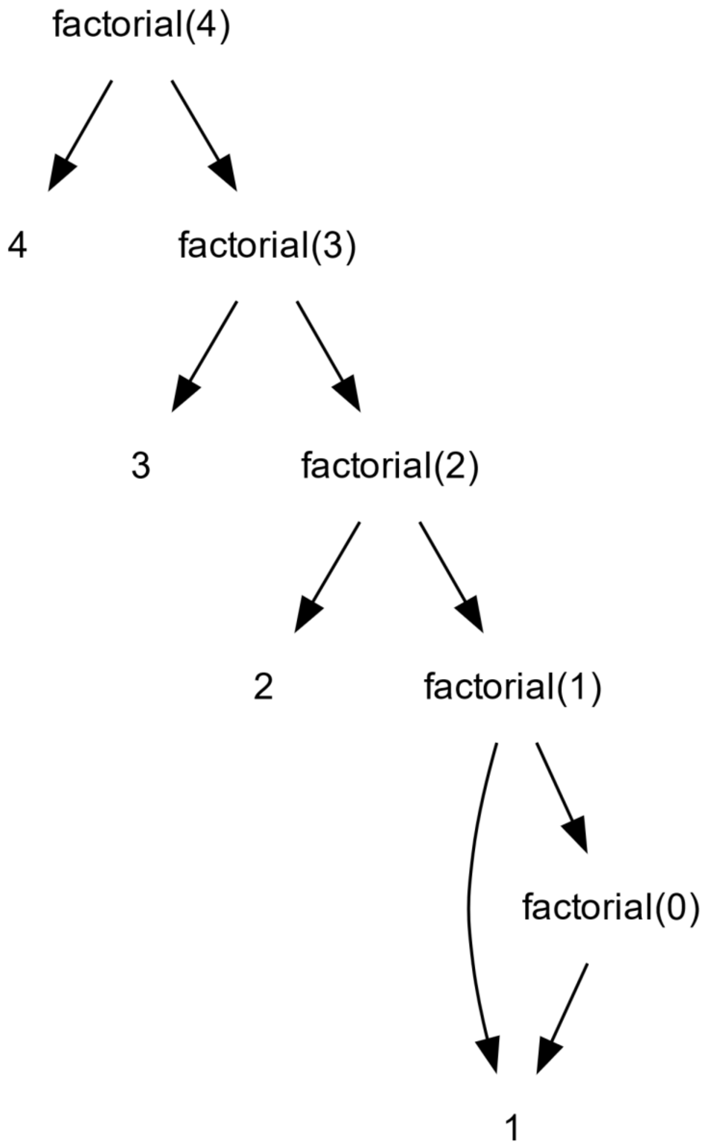
$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$

$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

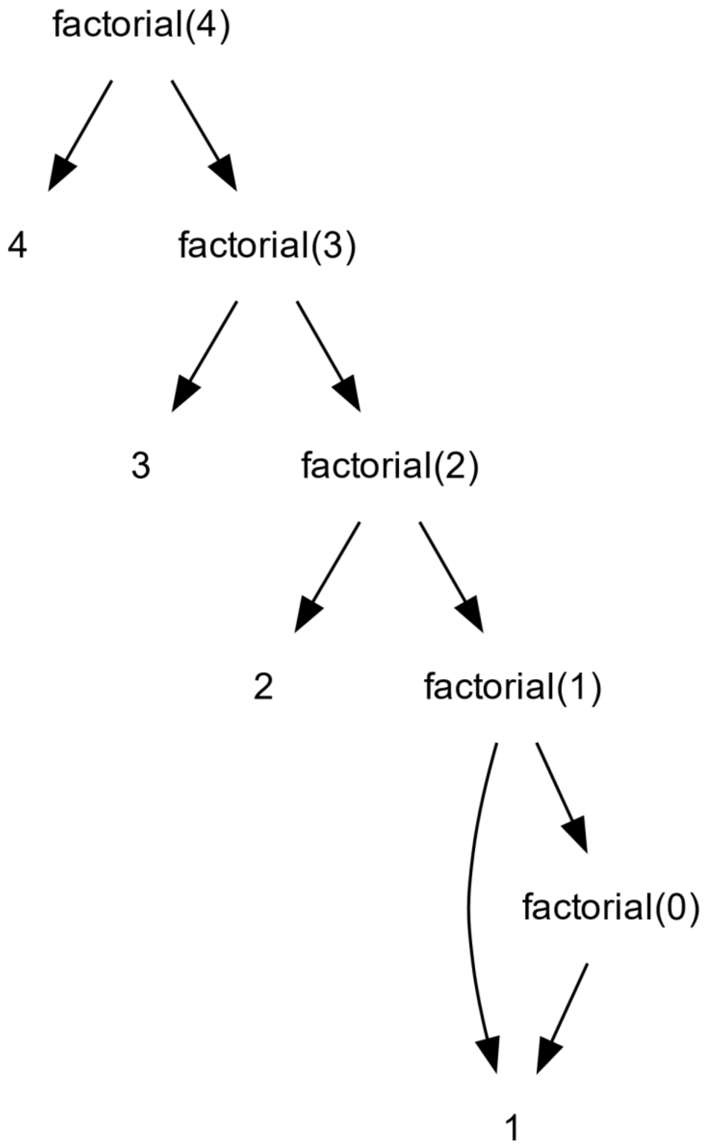


數學的階乘 $4!$

計算過程為：

$$
\begin{aligned}
4! &= 4 \times 3! \\
&= 4 \times 3 \times 2! \\
&= 4 \times 3 \times 2 \times 1! \\
&= 4 \times 3 \times 2 \times 1 \times 0! \\
&= 4 \times 3 \times 2 \times 1 \\
&= 24
\end{aligned}
$$

照著這個方向來看
我們知道了…



數學的階乘 4!

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$

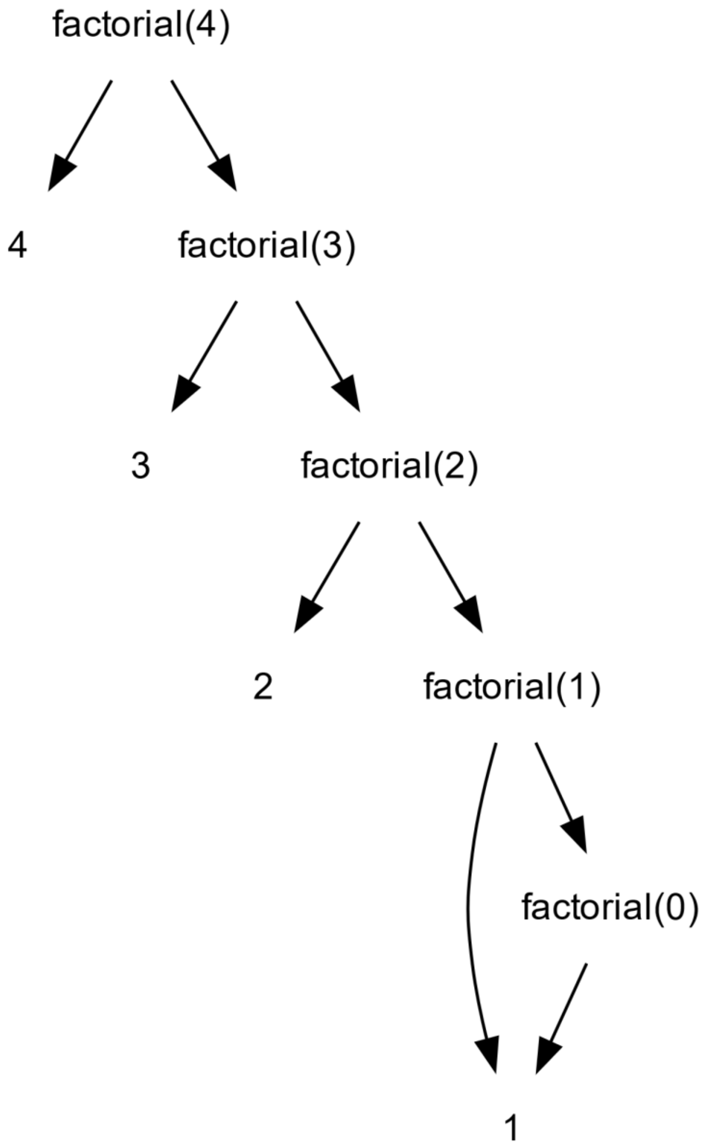
$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

照著這個方向來看
我們知道了…

n



數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$

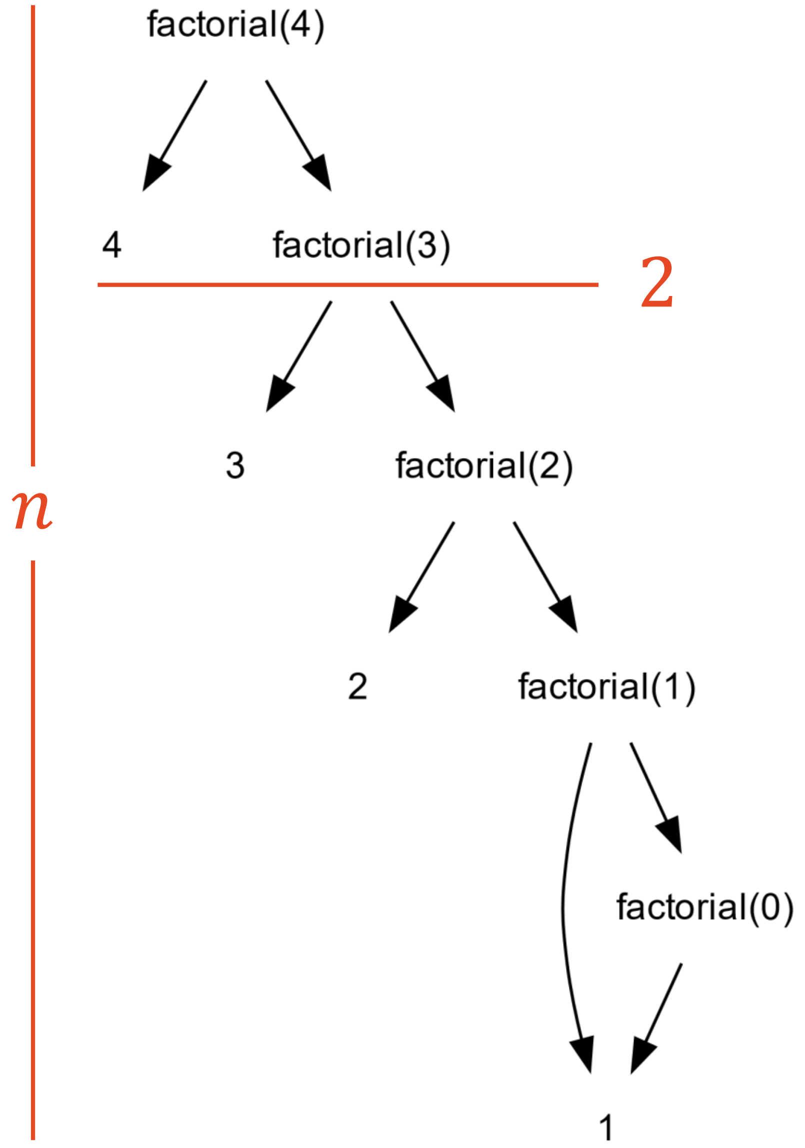
$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

照著這個方向來看

我們知道了…



數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

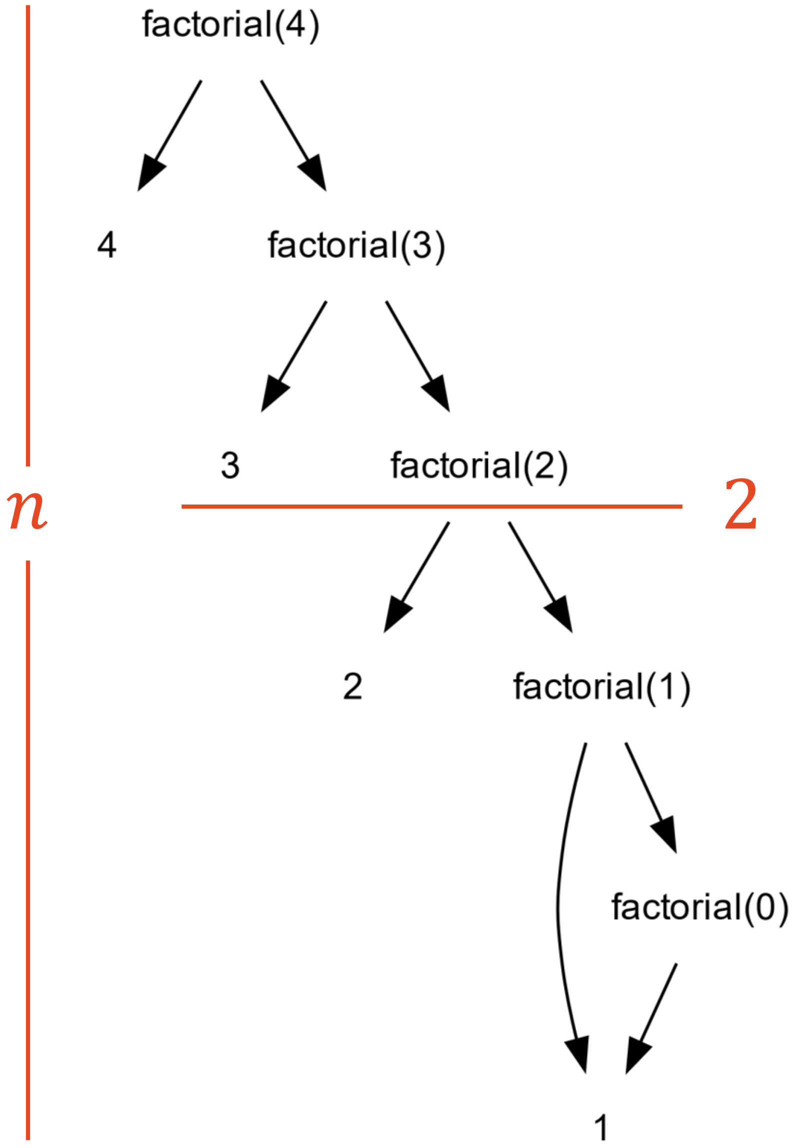
$= 4 \times 3 \times 2 \times 1!$

$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

照著這個方向來看
我們知道了…



數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

$= 4 \times 3 \times 2 \times 1!$

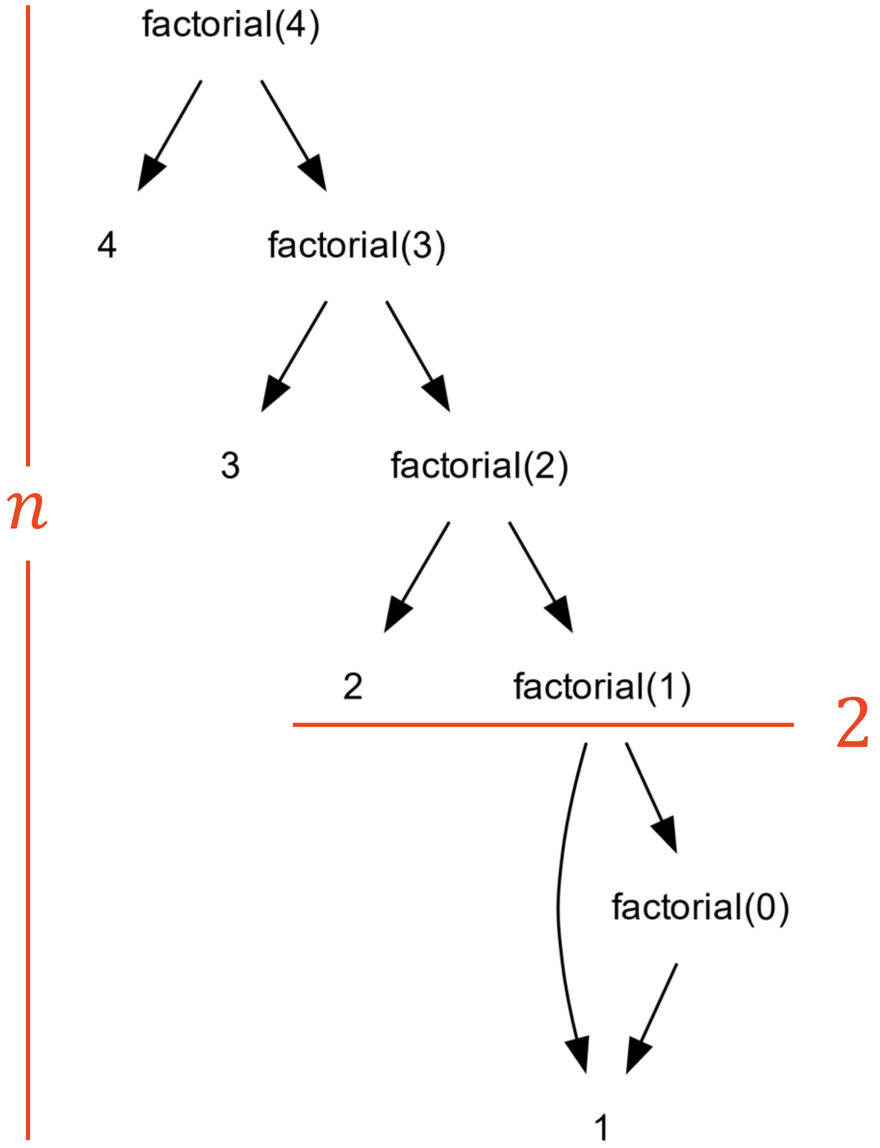
$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

照著這個方向來看

我們知道了…



數學的階乘 $4!$

計算過程為：

$4! = 4 \times 3!$

$= 4 \times 3 \times 2!$

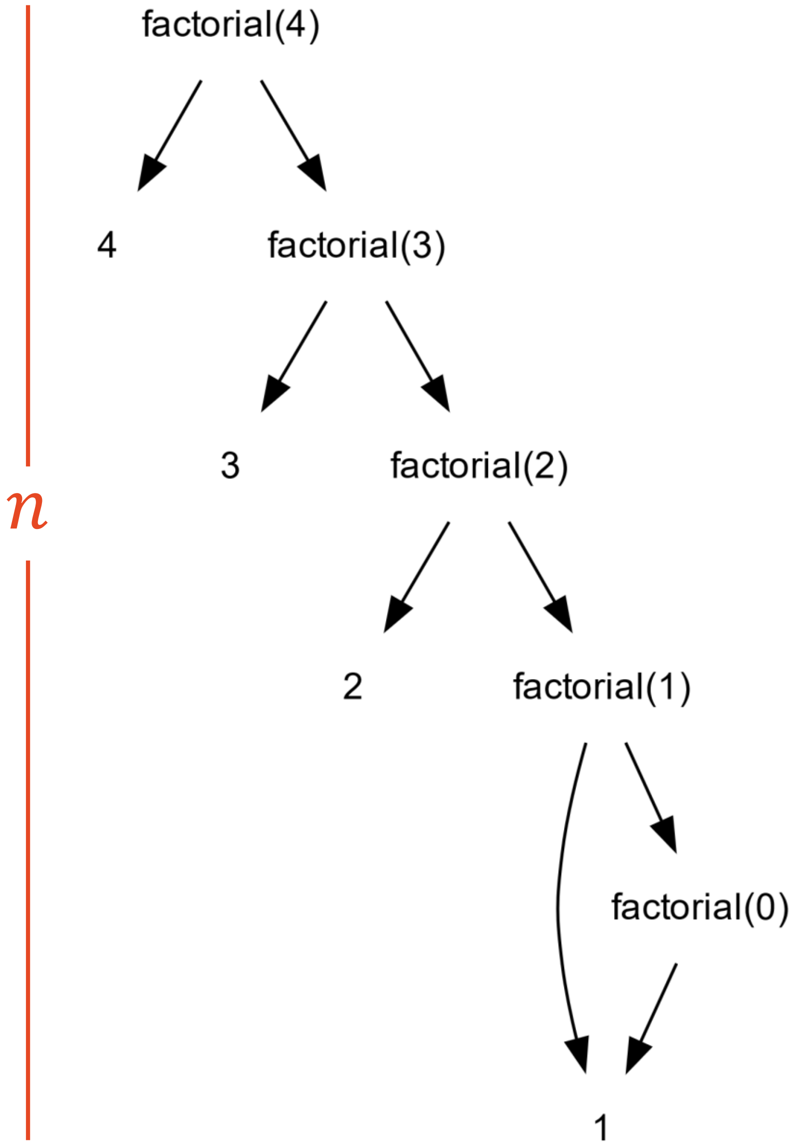
$= 4 \times 3 \times 2 \times 1!$

$= 4 \times 3 \times 2 \times 1 \times 0!$

$= 4 \times 3 \times 2 \times 1$

$= 24$

照著這個方向來看
我們知道了⋯



複雜度：$O(n)$

數學的階乘 $n!$

其定義為：

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 2 \times 1$$

其中 $0! = 1$

我們能只讓別人記住階乘的兩個性質

「持續連乘自身 $n-1$ 直到 $n=0$ 時結束」

就能映證「易於理解」的優勢

經典的遞迴函數

二元搜尋

一個常用在搜尋工作的演算法

二元搜尋

一個常用在搜尋工作的演算法

比起循序搜尋，它有著更高效率的演算流程

二元搜尋

一個常用在搜尋工作的演算法

比起循序搜尋，它有著更高效率的演算流程

我們記得一句話：

「持續平分陣列，直到指針碰到搜尋目標」

就能理解二分搜尋的核心想法

二元搜尋

一個常用在搜尋工作的演算法

在二元搜尋中會使用三個指針協助搜尋工作

1. Left 與 Right 指針
2. Mid 指針，用來尋找 $\mathrm{arr}\left[\frac{\mathrm{Left}}{\mathrm{Right}}\right]$ 是否為搜尋目標

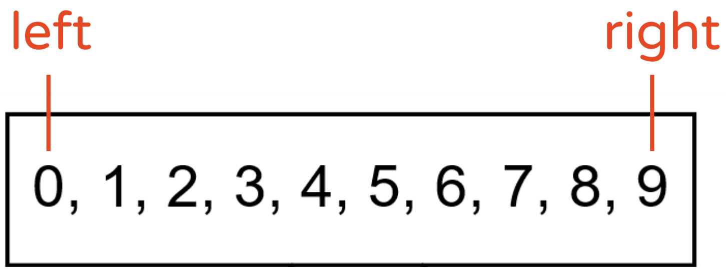
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

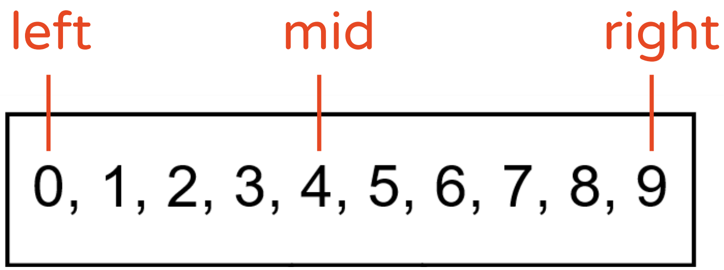
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



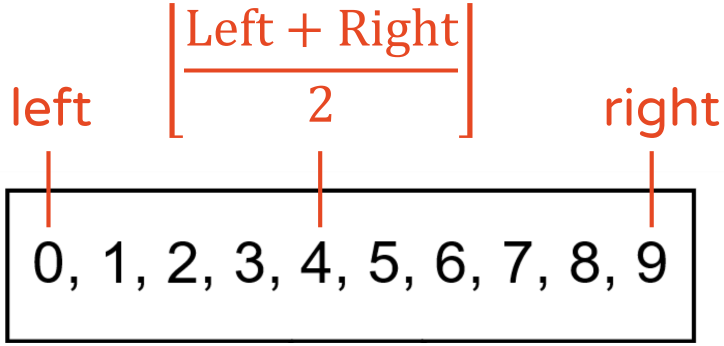
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



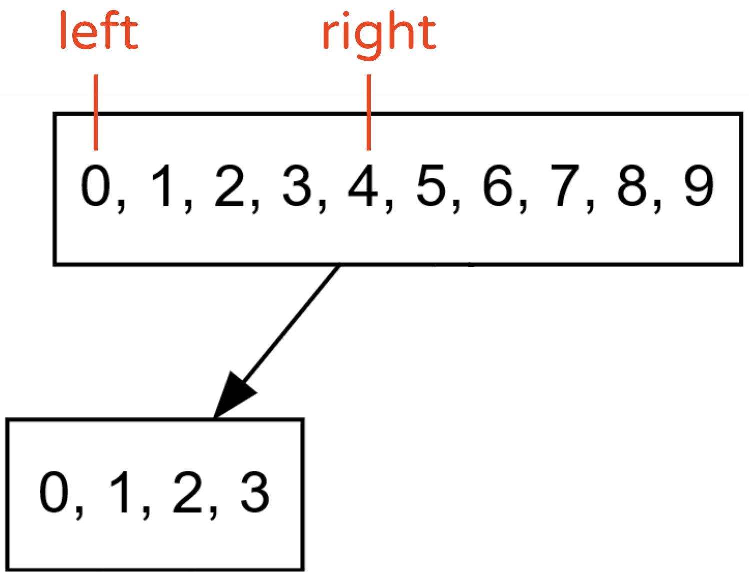
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



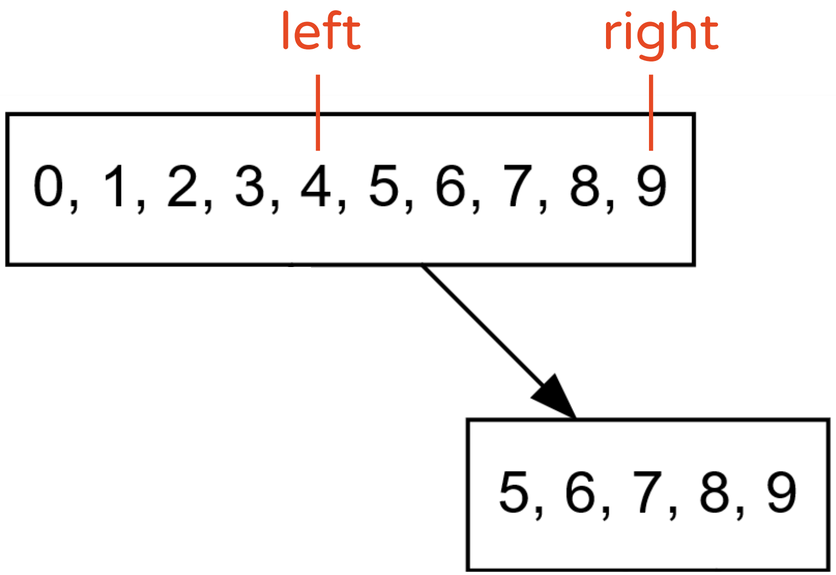
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



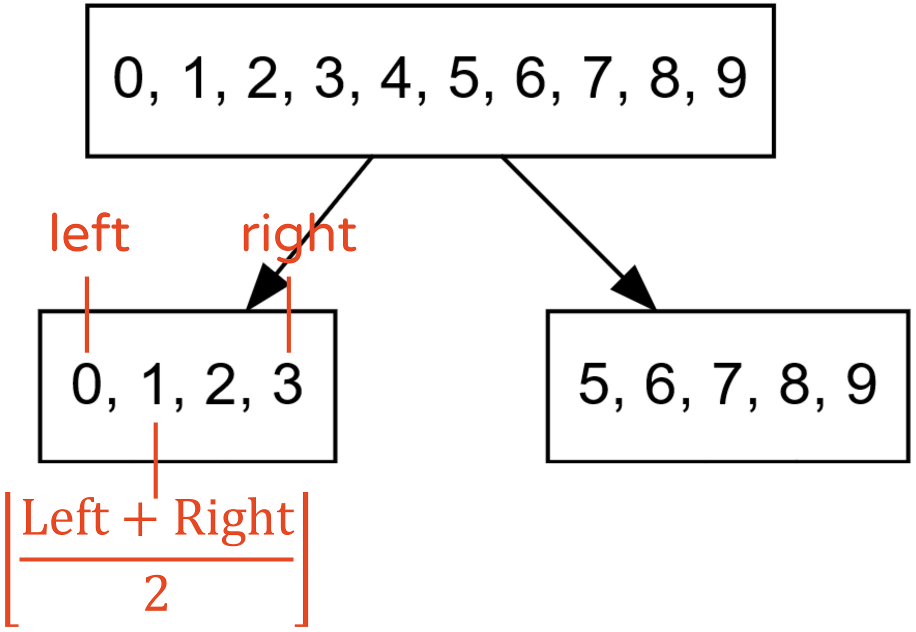
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



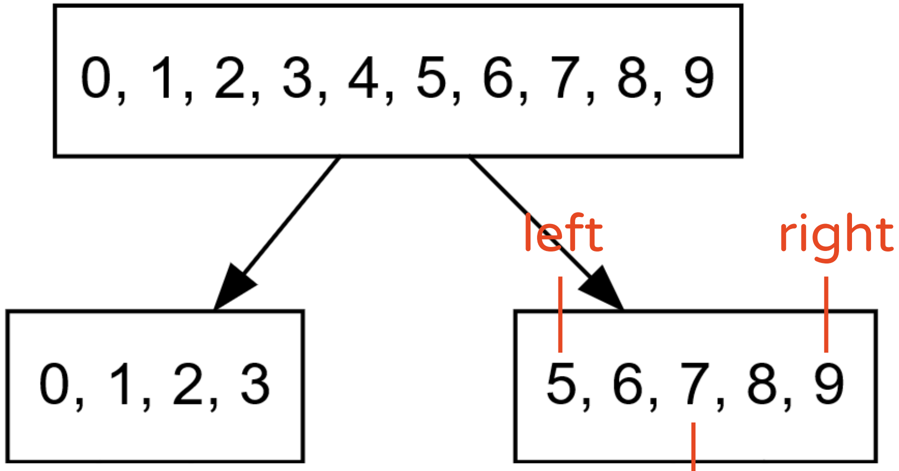
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



「持續平分陣列，直到指針碰到搜尋目標」

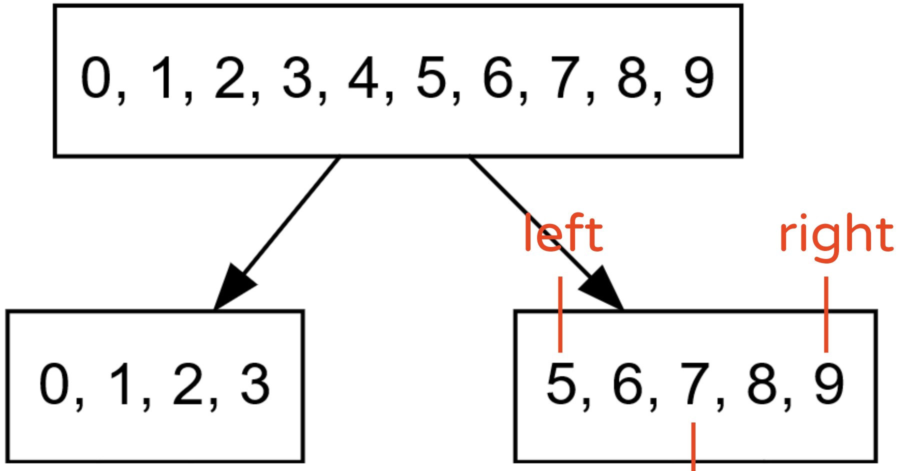
1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



$$\left\lfloor \frac{Left + Right}{2} \right\rfloor$$

「持續平分陣列，直到指針碰到搜尋目標」

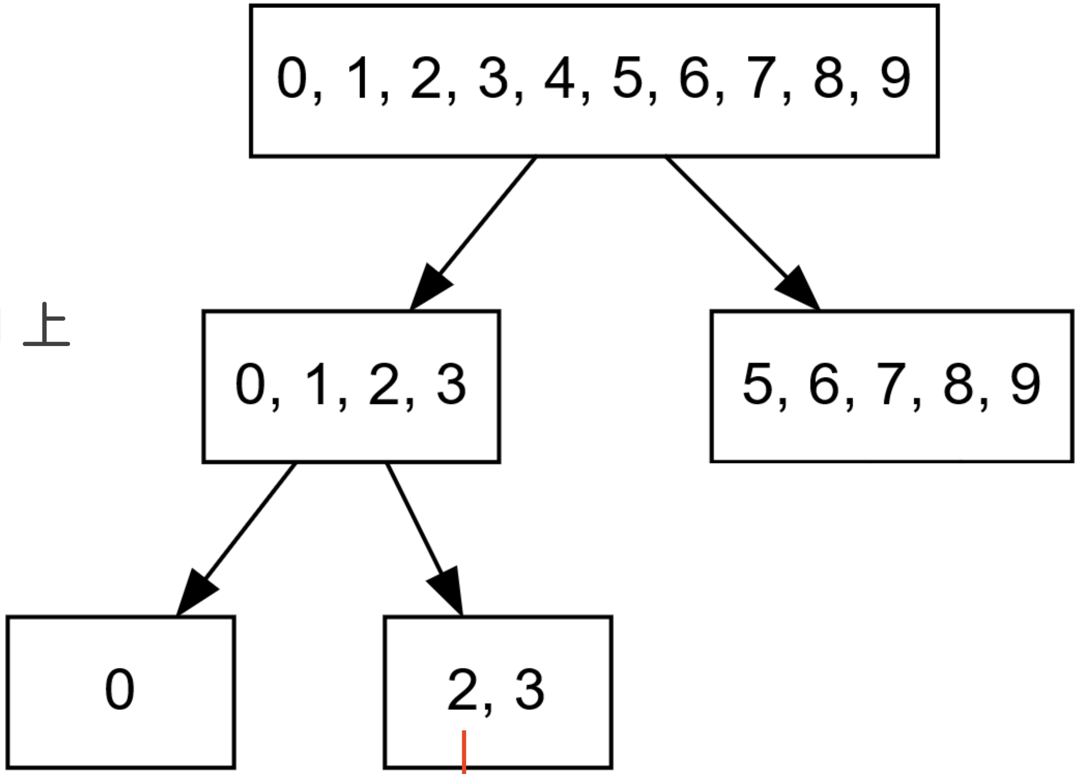
1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



$$\left\lfloor \frac{\text{Left} + \text{Right}}{2} \right\rfloor$$

「持續平分陣列，直到指針碰到搜尋目標」

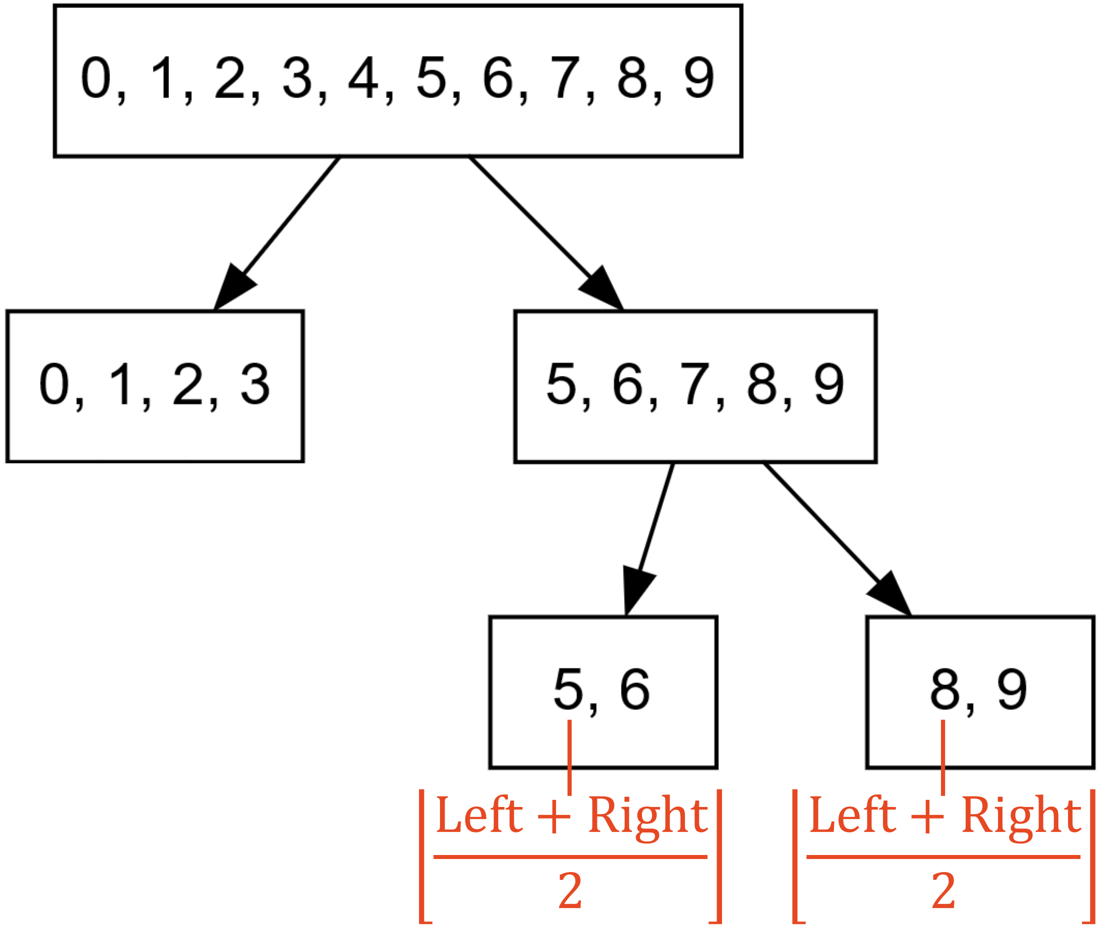
1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



$$\left\lfloor \frac{Left + Right}{2} \right\rfloor$$

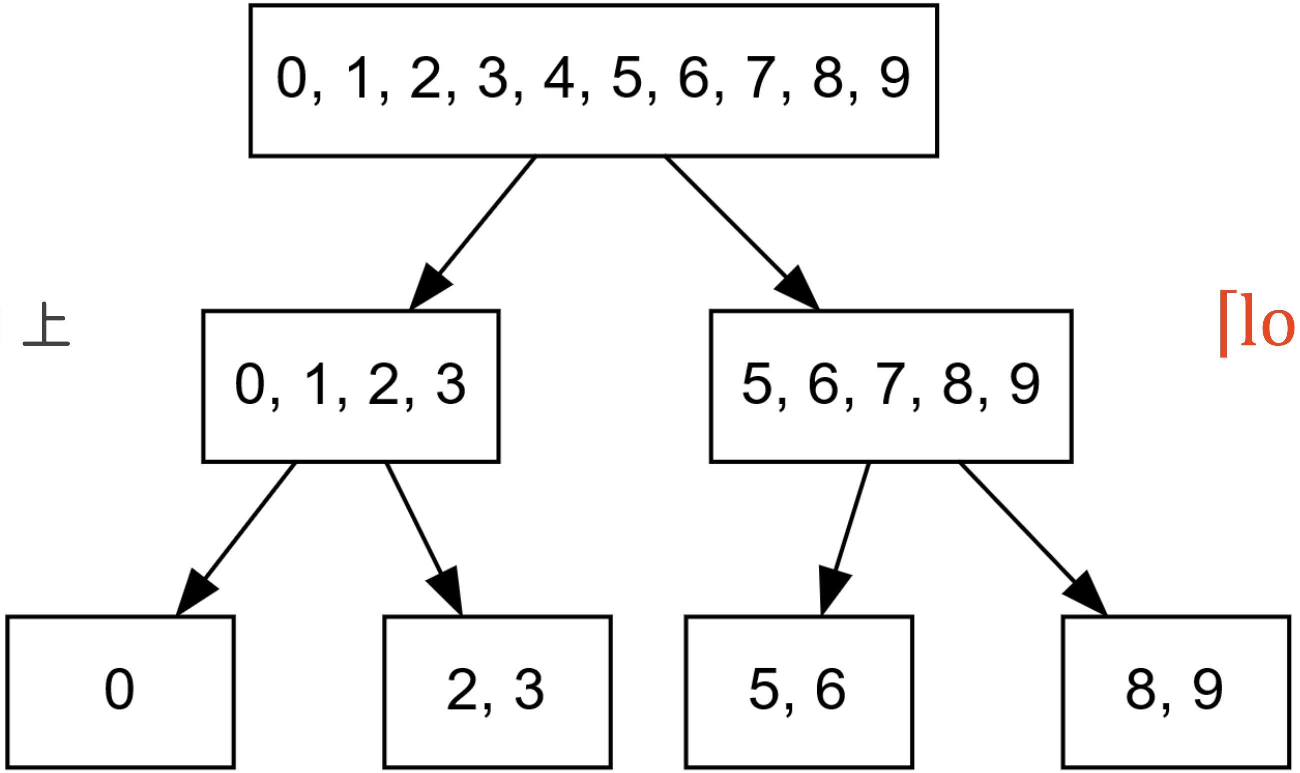
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



「持續平分陣列，直到指針碰到搜尋目標」

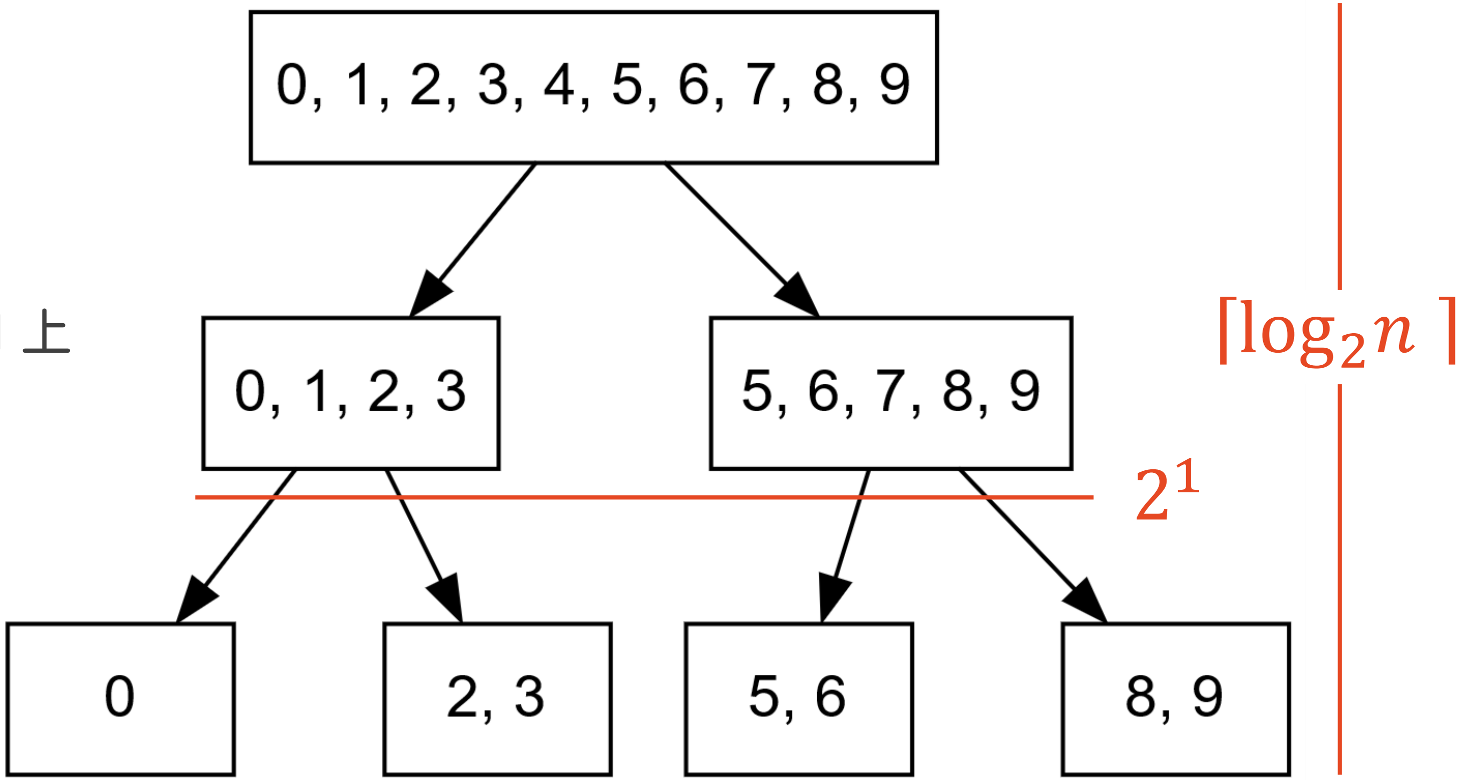
1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



$\lceil \log_2 n \rceil$

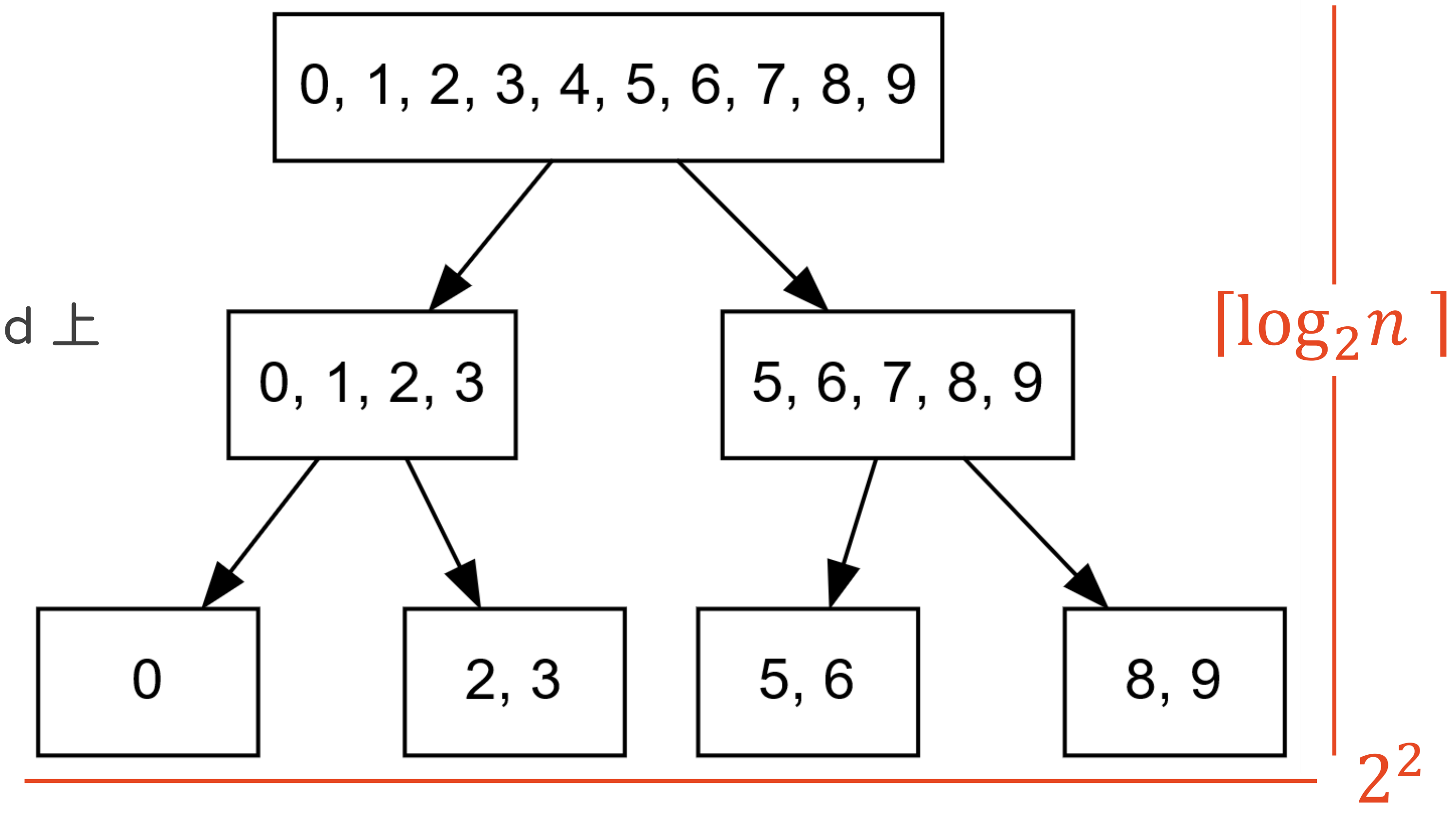
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



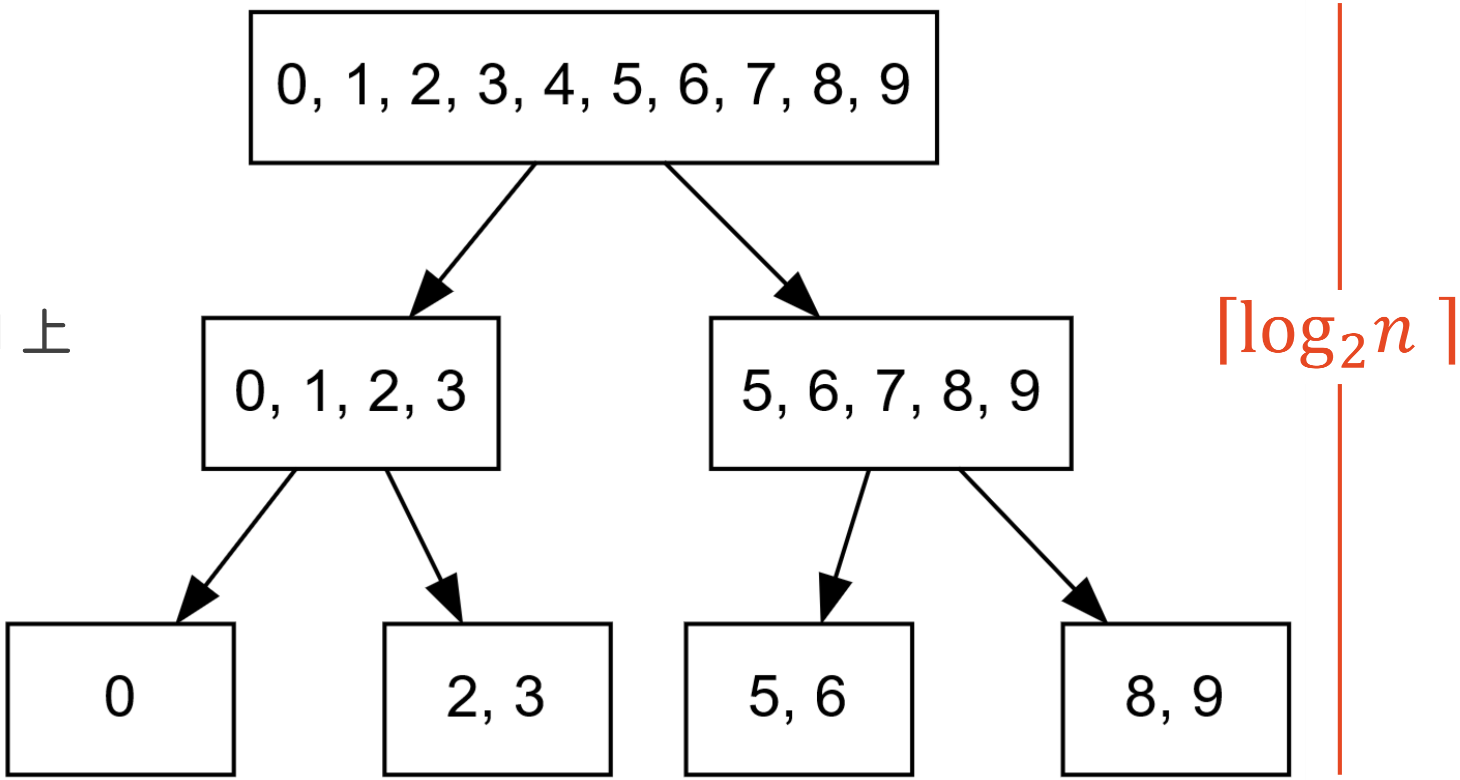
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



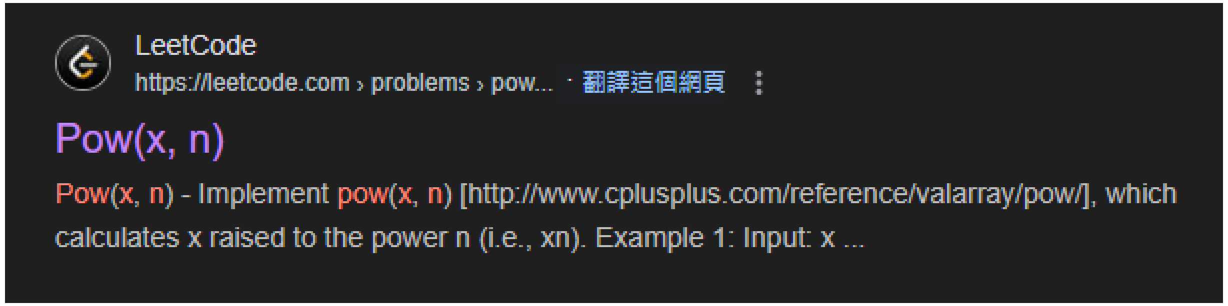
「持續平分陣列，直到指針碰到搜尋目標」

1. 給定一個陣列從 0 到 9
2. 找出每個節點的指針
3. 如果中間元素是目標就結束
4. 否則判斷大小，移動指針到 Mid 上



複雜度：$O(\log_2 n)$

leetcode Pow(x, n)



遞迴解還是迭代解

Pow(x, n)

數學式

$x^n = x \times x \times \cdots \times x$ (共 n 次)

$O(n)$

Pow(x, n)

數學式

$x^n = x \times x \times \cdots \times x$ (共 n 次)

$O(n)$

$x^{\text{INT\_MAX}} = ?$

Pow(x, n)

數學式

$x^n = x \times x \times \cdots \times x$ (共 n 次)

If $a + b = n$, then $x^n = x^{a+b}$

Pow(x, n)

數學式

$x^n = x \times x \times \cdots \times x$ (共 n 次)

If $a + b = n$, then $x^n = x^{a+b}$

e.g.,

$7^{11} = ?$

Pow(x, n)

數學式

$x^n = x \times x \times \cdots \times x$ (共 n 次)

If $a + b = n$, then $x^n = x^{a+b}$

e.g.,

$7^{11} = 7^{8+2+1}$

$11_{10} = 1011_2$

1. 如果你重視簡潔的 code：
2. 如果你重視易於理解的程式碼：
3. 如果你想算複雜度：
4. 如果你追求執行速度：

1. 如果你重視簡潔的 code：用遞迴解
2. 如果你重視易於理解的程式碼：
3. 如果你想算複雜度：
4. 如果你追求執行速度：

1. 如果你重視簡潔的 code：用遞迴解
2. 如果你重視易於理解的程式碼：用遞迴解
3. 如果你想算複雜度：
4. 如果你追求執行速度：

1. 如果你重視簡潔的 code：用遞迴解
2. 如果你重視易於理解的程式碼：用遞迴解
3. 如果你想算複雜度：都可以
4. 如果你追求執行速度：

1. 如果你重視簡潔的 code：用遞迴解
2. 如果你重視易於理解的程式碼：用遞迴解
3. 如果你想算複雜度：都可以
4. 如果你追求執行速度：用迭代解或DP（進階班）

謝謝聆聽

推薦題目會跟簡報一起發在 Discord 上