Efficient Algorithm

郭至軒（KuoE0）

KuoE0.tw@gmail.com
KuoE0.ch

Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)

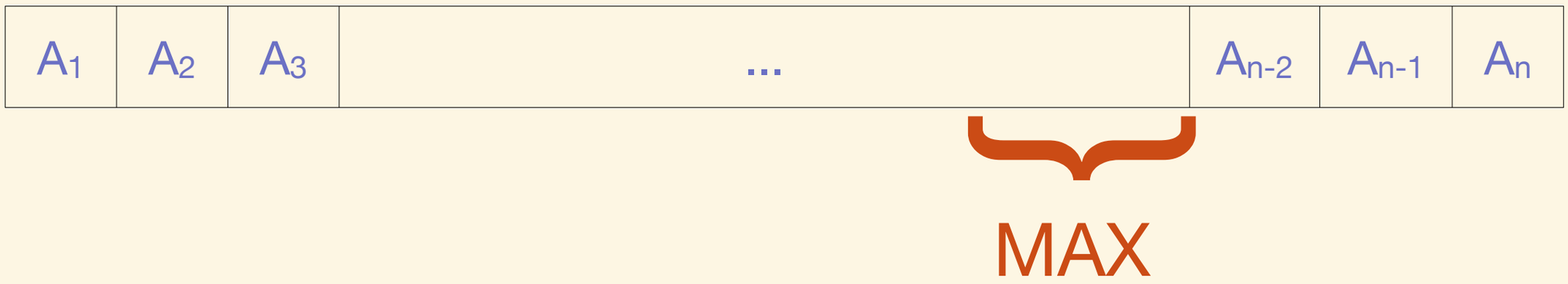
Latest update: Feb 27, 2013

Time Complexity

最大連續和

有一長度為 n 的數列 A_1, A_2, A_3, ...,

A_{n-1}, A_n, 求其最大連續區間和。



```
int MAX = A[ 1 ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    int sum = 0;
    for ( int k = i; k <= j; ++k )
      sum += A[ k ];
    MAX = max( MAX, sum );
  }
```

```
int MAX = A[ 1 ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    int sum = 0;
    for ( int k = i; k <= j; ++k )
      sum += A[ k ];
    MAX = max( MAX, sum );
  }
```

核心運算：加法

```
int MAX = A[ 1 ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    int sum = 0;
    for ( int k = i; k <= j; ++k )
      sum += A[ k ];
    MAX = max( MAX, sum );
  }
```

```
int MAX = A[ 1 ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    int sum = 0;
    for ( int k = i; k <= j; ++k )
      sum += A[ k ];
    MAX = max( MAX, sum );
  }
```

加法次數： $T(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} j - i + 1 = \frac{n(n+1)(n+2)}{6}$

$$T(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} j - i + 1 = \frac{n(n+1)(n+2)}{6}$$

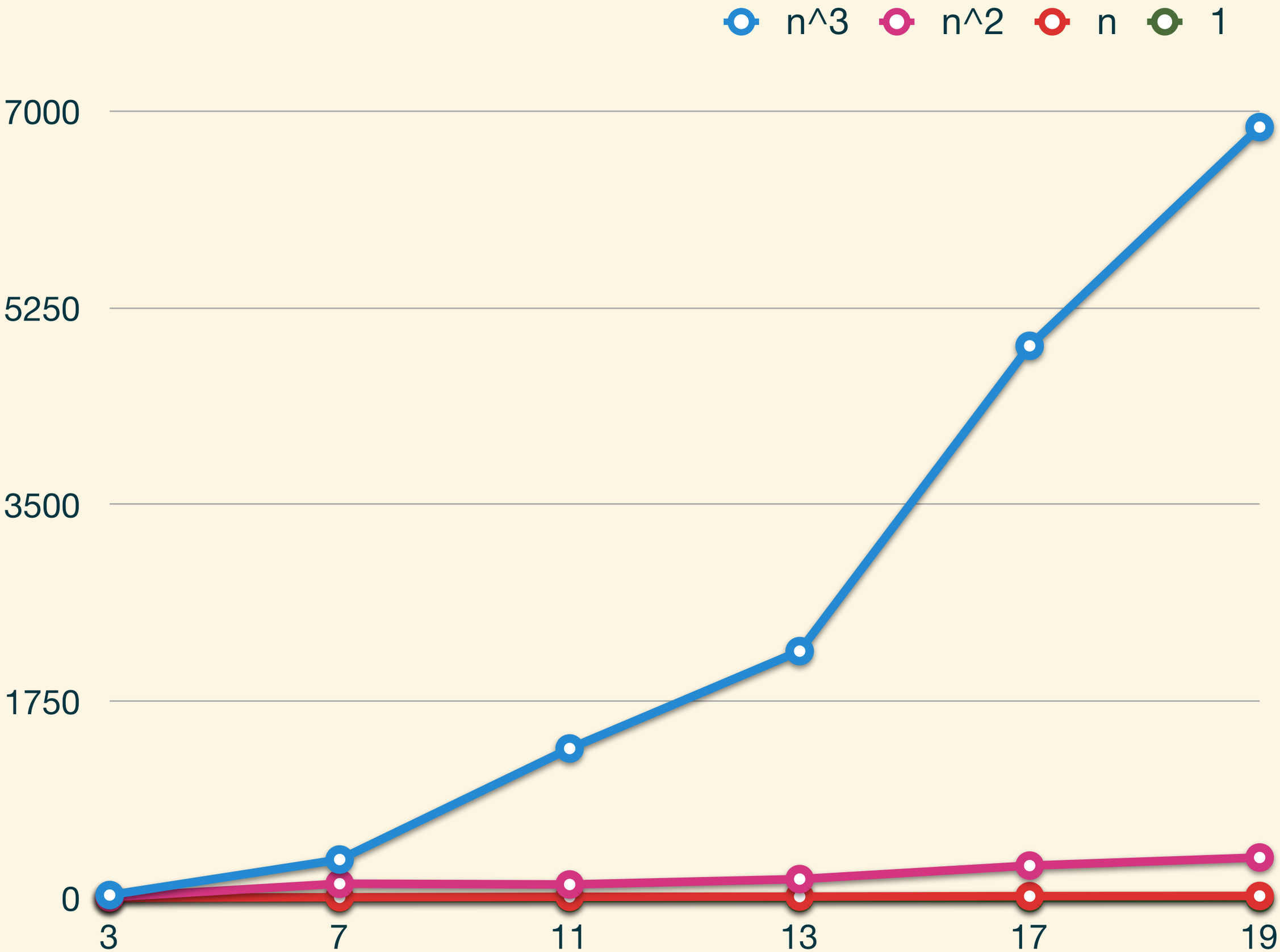
$$T(n)=\sum_{i=1}^{n}\sum_{j=i}^{n} j-i+1=\frac{n(n+1)(n+2)}{6}$$

三次式

$$T(n)=\sum_{i=1}^{n}\sum_{j=i}^{n} j-i+1=\frac{n(n+1)(n+2)}{6}$$

三次式

	n^3	n^2	n	1
$n=10$	10^3	10^2	10^1	1
$n=10^2$	10^6	10^4	10^2	1
$n=10^3$	10^9	10^6	10^3	1


n^3
n^2
n
1
7000
5250
3500
1750
0
3
7
11
13
17
19

最大指數次項影響最大

Time Complexity: $O(n^3)$

不需精確分析

進行精確分析較耗時間！

不需精確分析

進行精確分析較耗時間！

↓

略估迴圈次數

```
int MAX = A[ 1 ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    int sum = 0;
    for ( int k = i; k <= j; ++k )
      sum += A[ k ];
    MAX = max( MAX, sum );
  }
```

	最多次數
第 1 層迴圈	n
第 2 層迴圈	n
第 3 層迴圈	n

$$n \times n \times n = n^3$$

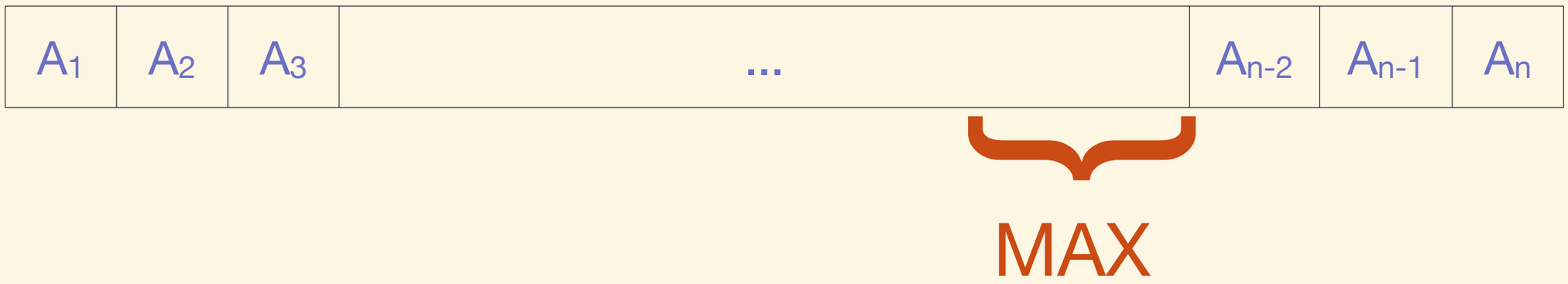
Time Complexity: $O(n^3)$

前處理優化

最大連續和

有一長度為 n 的數列 A_1, A_2, A_3, ...,

A_{n-1}, A_n, 求其最大連續區間和。



設數列 S 為數列 A 的累積和,

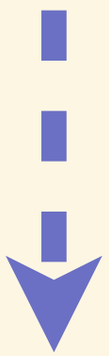
因此 $S_i = A_1 + A_2 + \ldots + A_i$, 則

$A_i + A_{i+1} + \ldots + A_j = S_j - S_{i-1}$。

設數列 S 為數列 A 的累積和,

因此 $S_i = A_1 + A_2 + \ldots + A_i$, 則

$A_i + A_{i+1} + \ldots + A_j = S_j - S_{i-1}$。



求區間 [i,j] 的連續和僅需要 O(1) 的時間。

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	
S_1	3	7	-1	-3	9	-10	1	2	3
S_2	3	7	-1	-3	9	-10	1	2	10
S_3	3	7	-1	-3	9	-10	1	2	9
S_4	3	7	-1	-3	9	-10	1	2	6
S_5	3	7	-1	-3	9	-10	1	2	15
S_6	3	7	-1	-3	9	-10	1	2	5
S_7	3	7	-1	-3	9	-10	1	2	6
S_8	3	7	-1	-3	9	-10	1	2	8

```
int MAX = A[ 1 ];
S[ 0 ] = 0;
for ( int i = 1; i <= n; ++i )
  S[ i ] = S[ i - 1 ] + A[ i ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    MAX = max( MAX, S[ j ] - S[ i - 1 ] );
  }
```

```
int MAX = A[ 1 ];
S[ 0 ] = 0;
for ( int i = 1; i <= n; ++i )
  S[ i ] = S[ i - 1 ] + A[ i ];
for ( int i = 1; i <= n; ++i )
  for ( int j = i; j <= n; ++j ) {
    MAX = max( MAX, S[ j ] - S[ i - 1 ] );
  }
```

O(n)

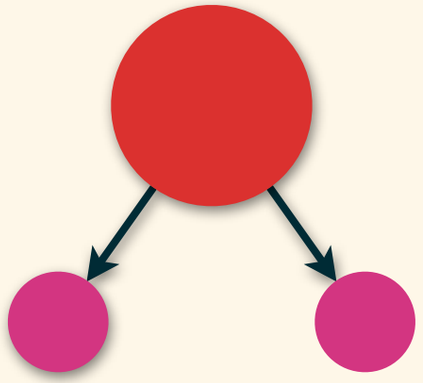
O(n²)

最大指數次項：O(n²)

Divide & Conquer

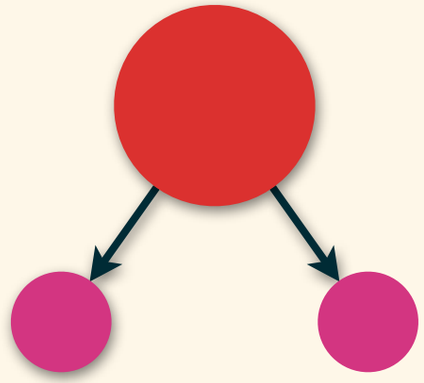
分而治之

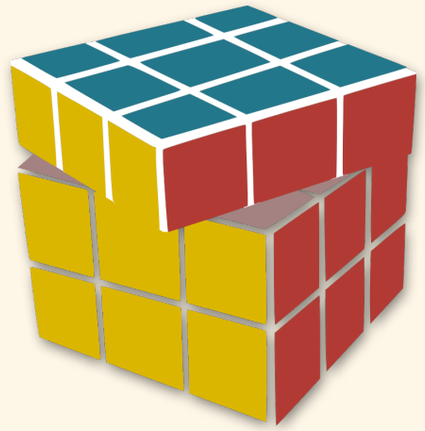
劃分為相同的子問題



劃分為相同的子問題

遞迴求解子問題

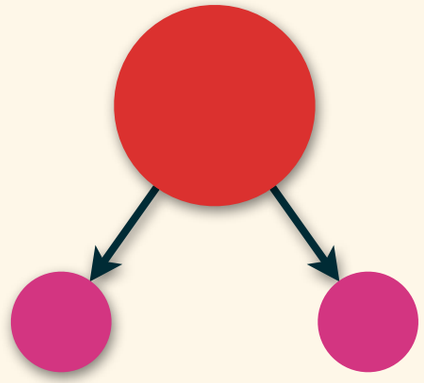


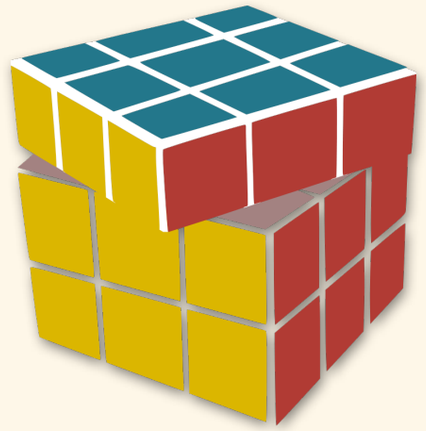


劃分為相同的子問題

遞迴求解子問題

將子問題的解合併

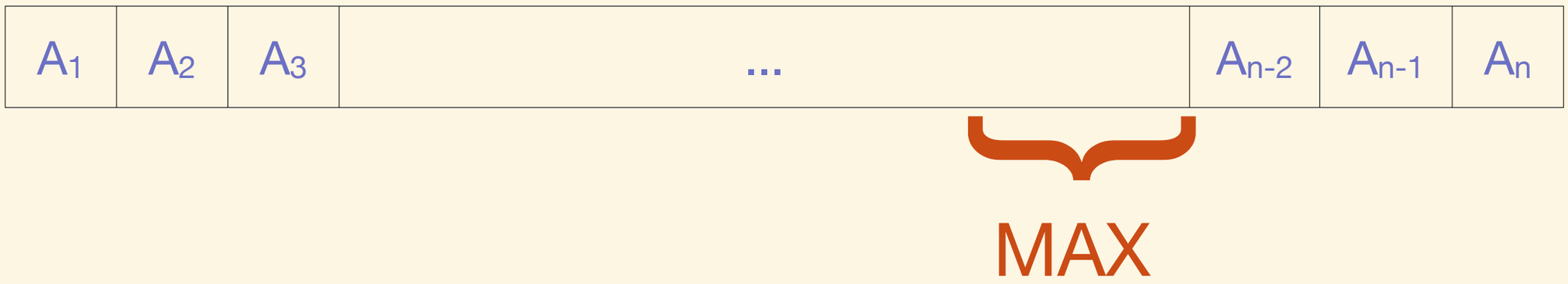




最大連續和

有一長度為 n 的數列 A_1, A_2, A_3, ...,

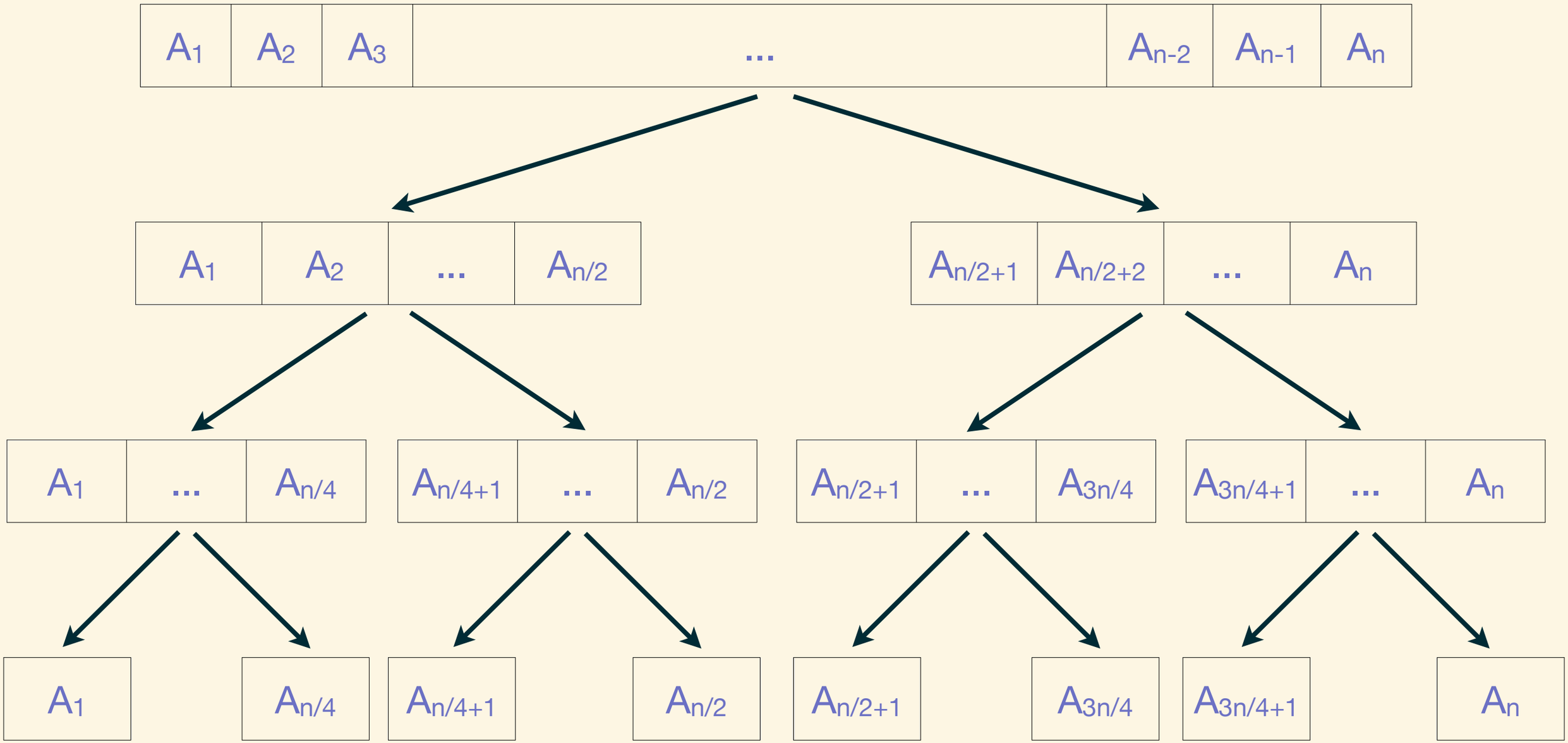
A_{n-1}, A_n, 求其最大連續區間和。



切割問題

A_1	A_2	A_3	...	A_{n-2}	A_{n-1}	A_n

切割問題


A1 A2 A3 ... An-2 An-1 An
A1 A2 ... An/2
An/2+1 An/2+2 ... An
A1 ... An/4
An/4+1 ... An/2
An/2+1 ... A3n/4
A3n/4+1 ... An
A1
An/4
An/4+1
An/2
An/2+1
A3n/4
A3n/4+1
An

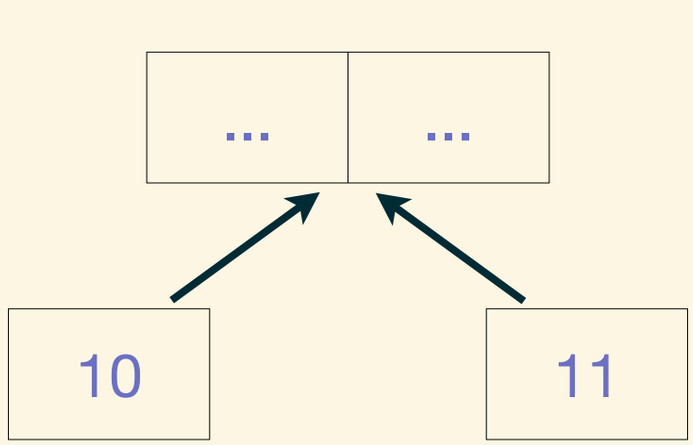
求解問題

遞迴直到數列僅剩一元素，直接回傳該數值。

僅有一元素時，該數值為最大連續和。

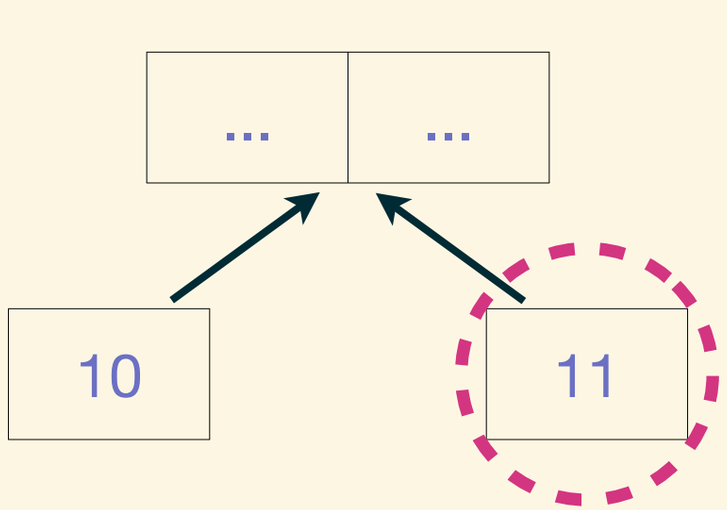
合併子問題解

選擇子問題中最優解！

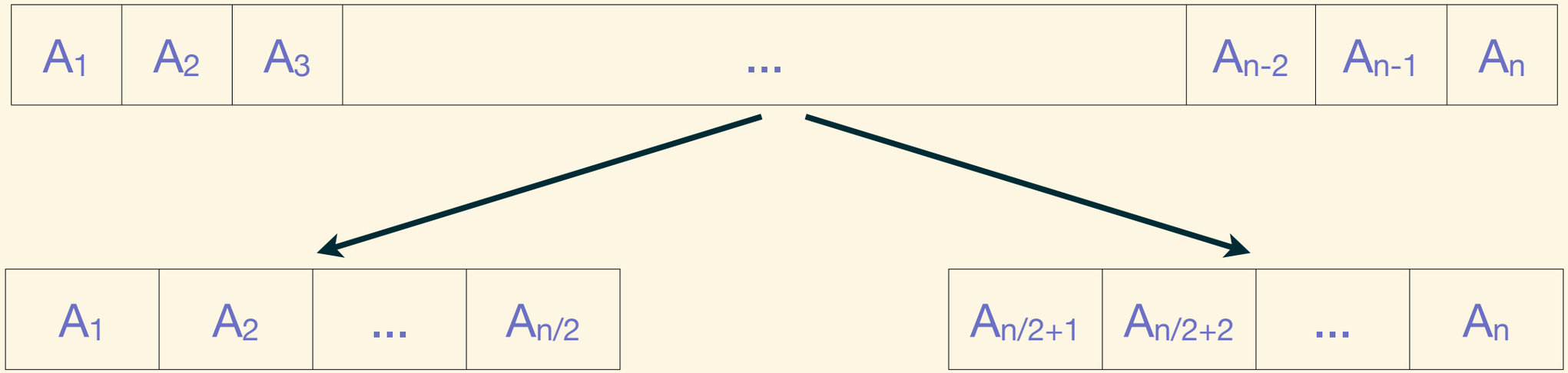


合併子問題解

選擇子問題中最優解！

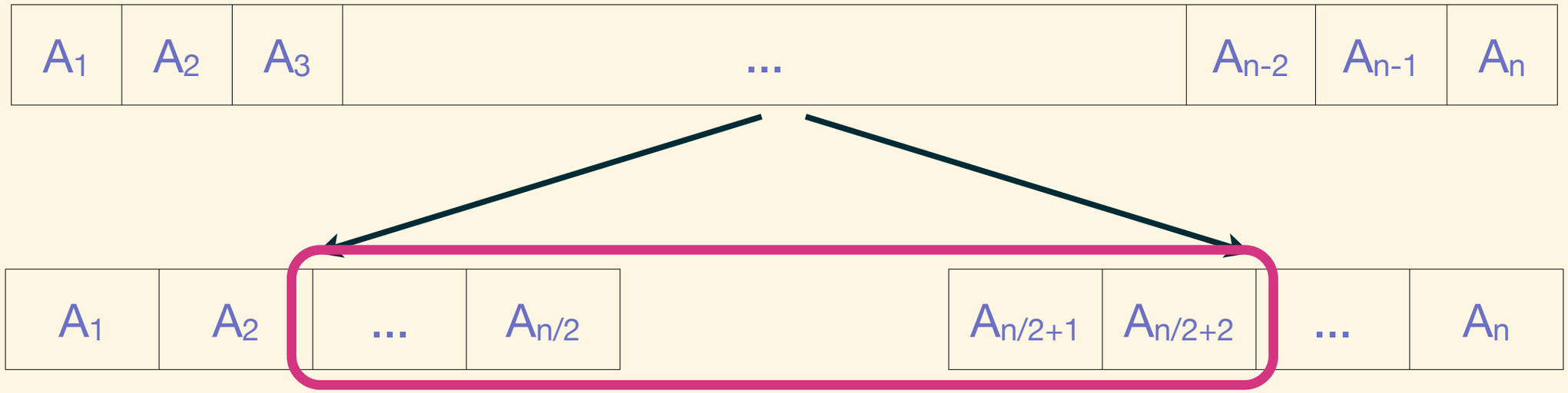


But....



But....

答案可能橫跨兩個區間！



How-to

5	-9	3	1

3	-10	3	3

How-to

5	-9	3	1

3	-10	3	3

最大連續和：5

How-to

5	-9	3	1

最大連續和：5

3	-10	3	3

最大連續和：6

How-to

5	-9	3	1

最大連續和：5

3	-10	3	3

最大連續和：6

當前最大連續和：6

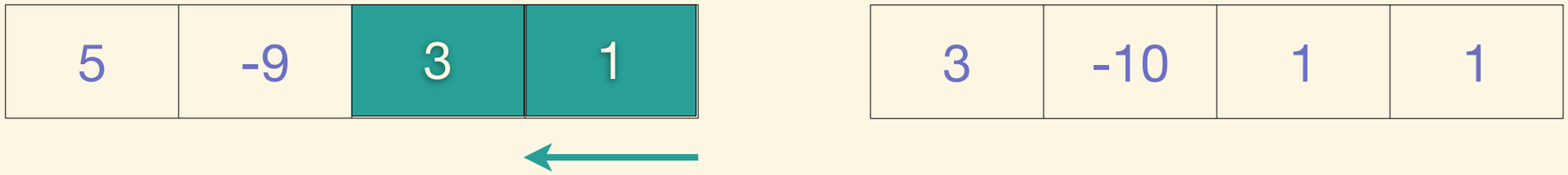
How-to

5	-9	3	1

3	-10	1	1

當前最大連續和：6

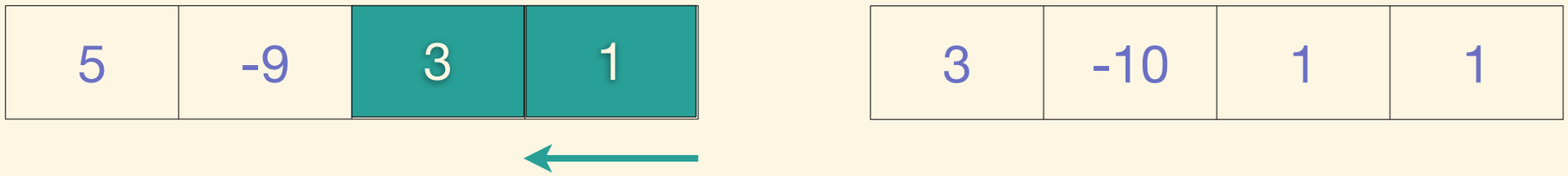
How-to



自右而左找最大連續和

當前最大連續和：6

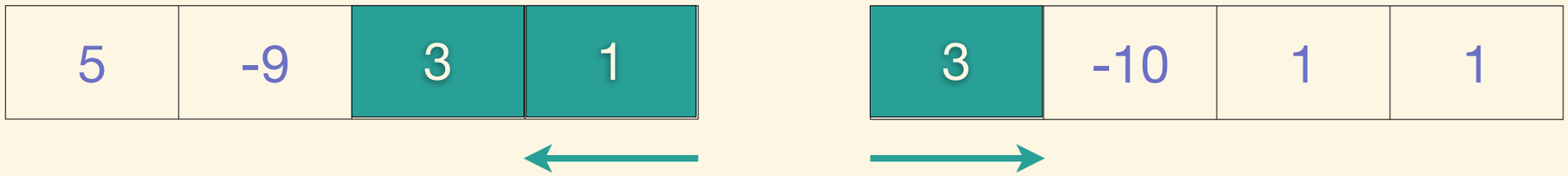
How-to



連續和：4

當前最大連續和：6

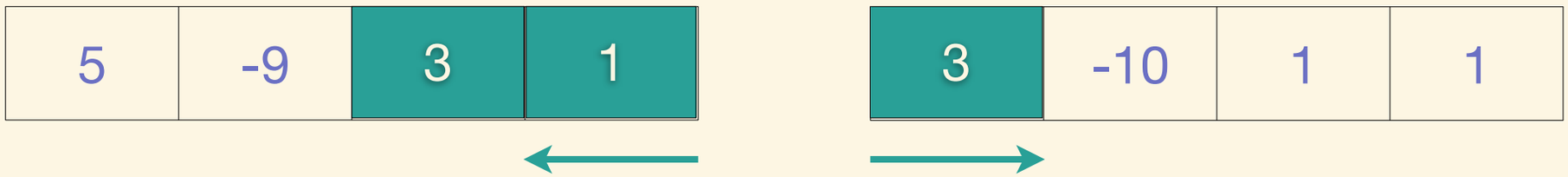
How-to



連續和：4　　自左而右找最大連續和

當前最大連續和：6

How-to



當前最大連續和：6

How-to

5	-9	3	1	3	-10	1	1

連續和：7

當前最大連續和：6

How-to

5	-9	3	1	3	-10	1	1

連續和：7

最終最大連續和：7

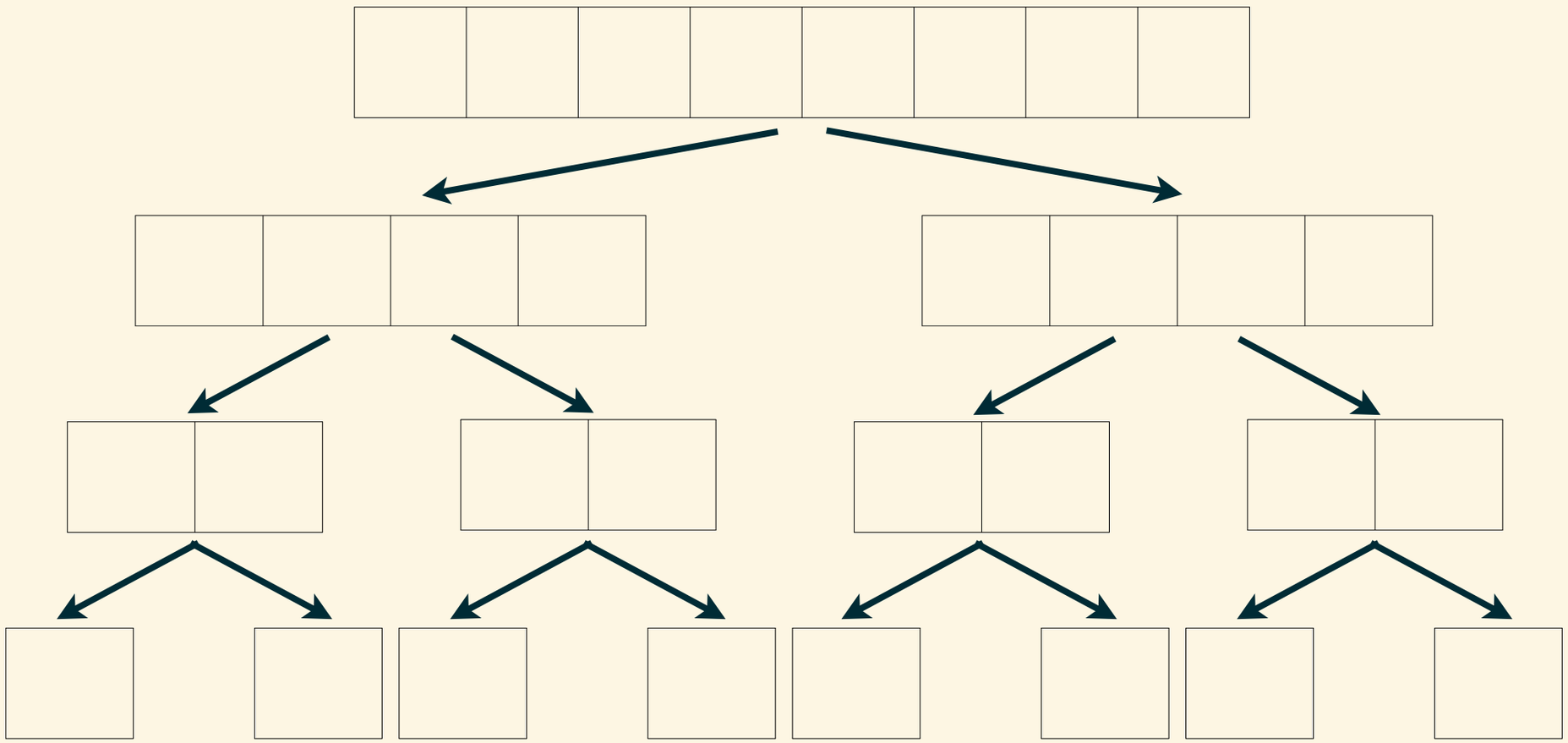
```
int maxsum( int L, int R ) {
  if ( R - L == 1 )
    return A[ L ];
  int mid = ( L + R ) / 2;
  int MAX = max( maxsum( L, mid ), maxsum( mid,
  R ) );
  int tempL = 0, tempR = 0;
  for ( int i = mid - 1, sum = 0; i >= L; --i )
    tempL = max( tempL, sum += A[ i ] );
  for ( int i = mid, sum = 0; i < R; ++i )
    tempR = max( tempR, sum += A[ i ] );
  return max( MAX, tempL + tempR );
}
```

Time Complexity

total time

Time Complexity

total time



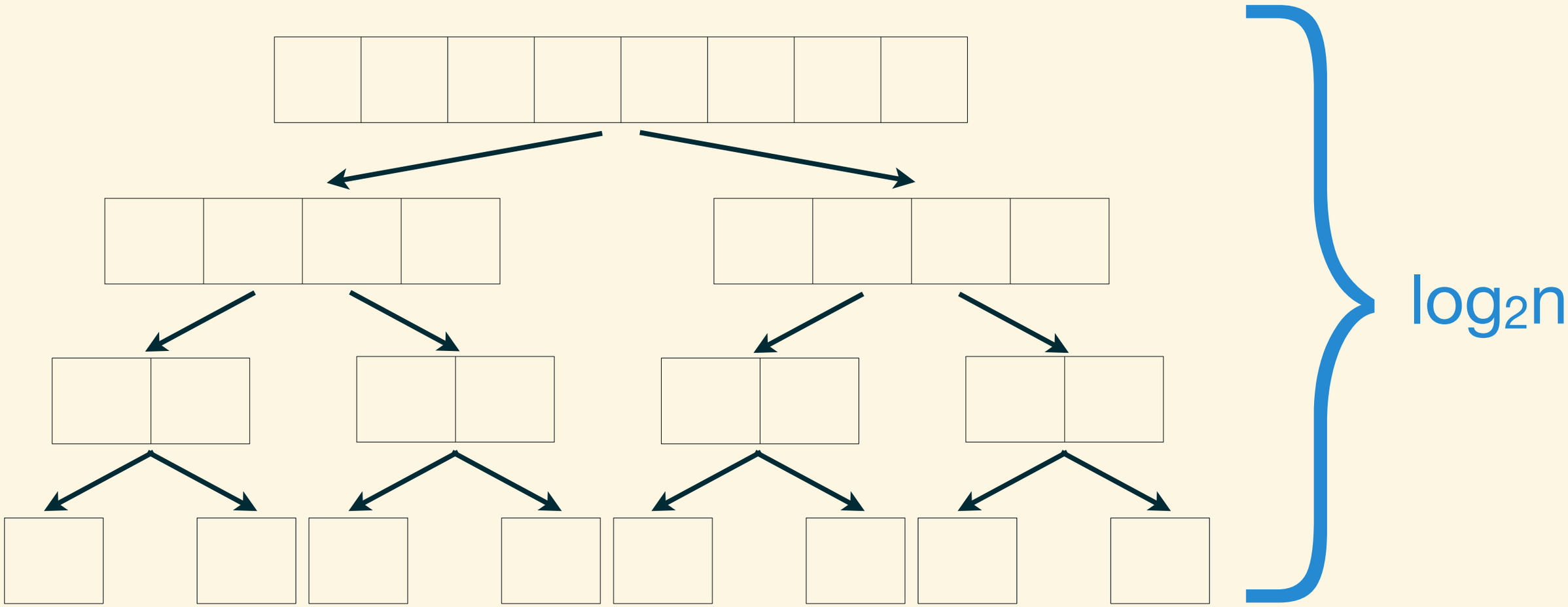
n

2 * (n/2) = n

...

n * (n/n) = n

The height



Time Complexity: $O(n \times \log_2 n)$

Time Complexity of Divide & Conquer

子問題扣除遞迴呼叫的時間複雜度　C(n)

遞迴呼叫層數　H(n)

Time Complexity: C(n) × H(n)

Time Comparison

	n^3	n^2	$n\log_2 n$
100	0.15	0.02	0.01
1000	**123.86**	1.36	**0.14**

單位：秒

從測試資料判斷演算法

假設機器 1 秒可以執行 10^6 個指令，...

時間複雜度	$n!$	2^n	n^3	n^2	$n\log_2 n$	n
最大規模	9	20	10^2	10^3	7.5×10^4	10^6

另一種說法

假設機器 1 秒可以執行 10^8 個指令，...

時間複雜度	$n!$	2^n	n^3	n^2	$n\log_2 n$	n
最大規模	11	26	464	10^4	4.5×10^6	10^8

Practice Now

10245 - The Closest Pair Problem

Reference

- 提升程式設計的邏輯思考力 (link)

Thank You for Your Listening.