Divide and Conquer I

Michael Tsai

2013/9/12

Algorithm Design Strategy

- 不是教你"某種演算法"
- 而是怎麼用"某些策略"來"設計演算法"
- 第一課: 各個擊破=Divide & Conquer

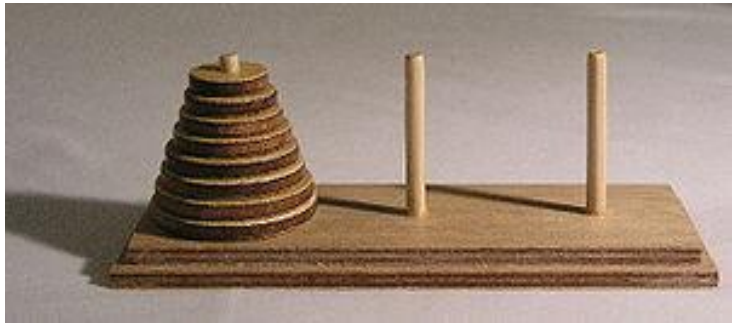
什麼是Divide-and-Conquer

- 當碰到一個問題的時候:
  1. 把問題分解(Divide)成一些比較小的同樣問題
  2. if 問題小到可以直接解決(Conquer),
     - then 直接解決 Base case
     - else 遞迴地呼叫自己的分身解決較小的這些問題 Recursive case
  3. 把已解決的小問題解答結合(Combine)起來變成原來的問題的解答

Divide and Conquer的好處

- 容易能解決困難的問題
  - 思考模式: 解決最簡單的case + 整合小問題的答案變成大問題的答案
- 通常也容易因此想出更有效率的演算法
  - 執行時間的複雜度比較低
- 適合平行運算 (Multi-core systems!)
- 更有效的記憶體存取
  - 小的subprogram與它的subprogram們的資料都可以放在在CPU的cache裡面, 而不需要存取速度比較慢的主記憶體

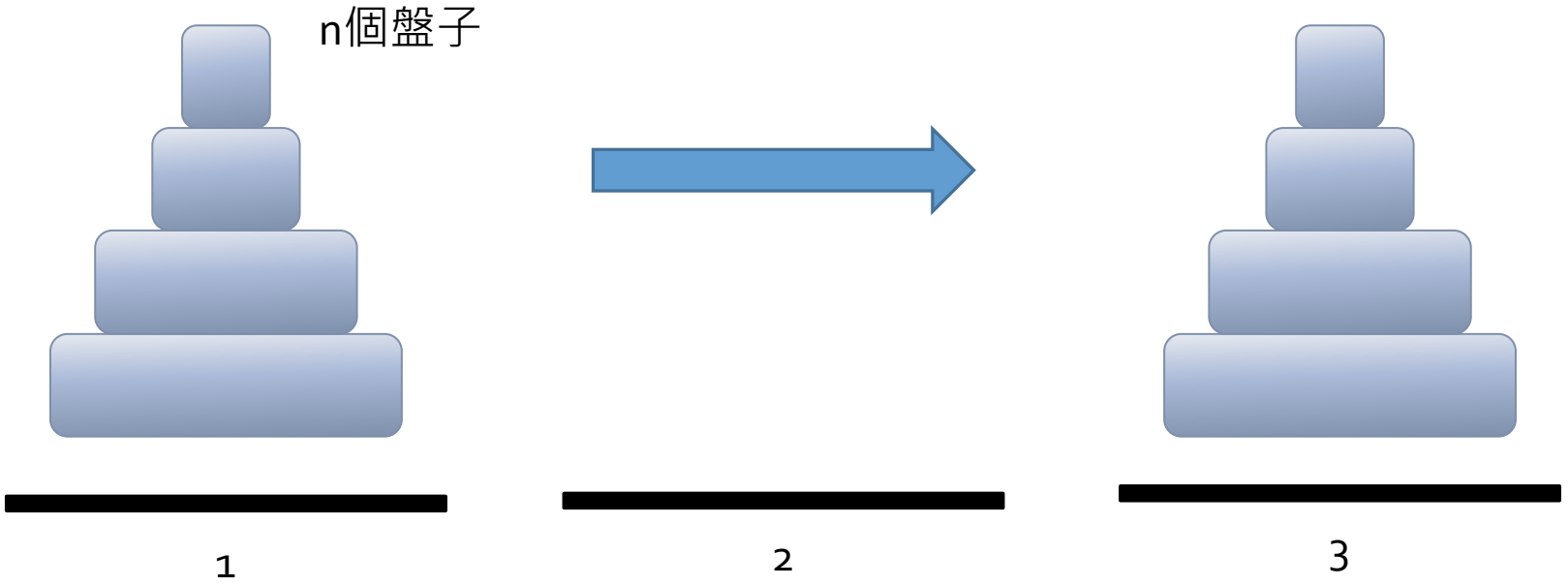
例子1：河內塔



- 規則:
  1. 每次可以移動每根棍子上最上面的盤子, 到其他棍子已有的盤子上.
  2. 大盤子不能放在小盤子上面
  3. 一次只能移動一個盤子

例子1：河内塔

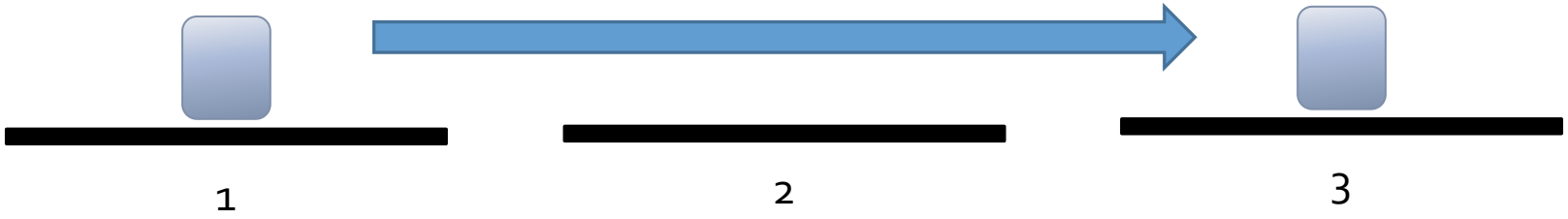
目標: n個盤子, 柱子1移到柱子3



例子1：河内塔

Base Case:

n=1時, 直接可以把盘子移過去



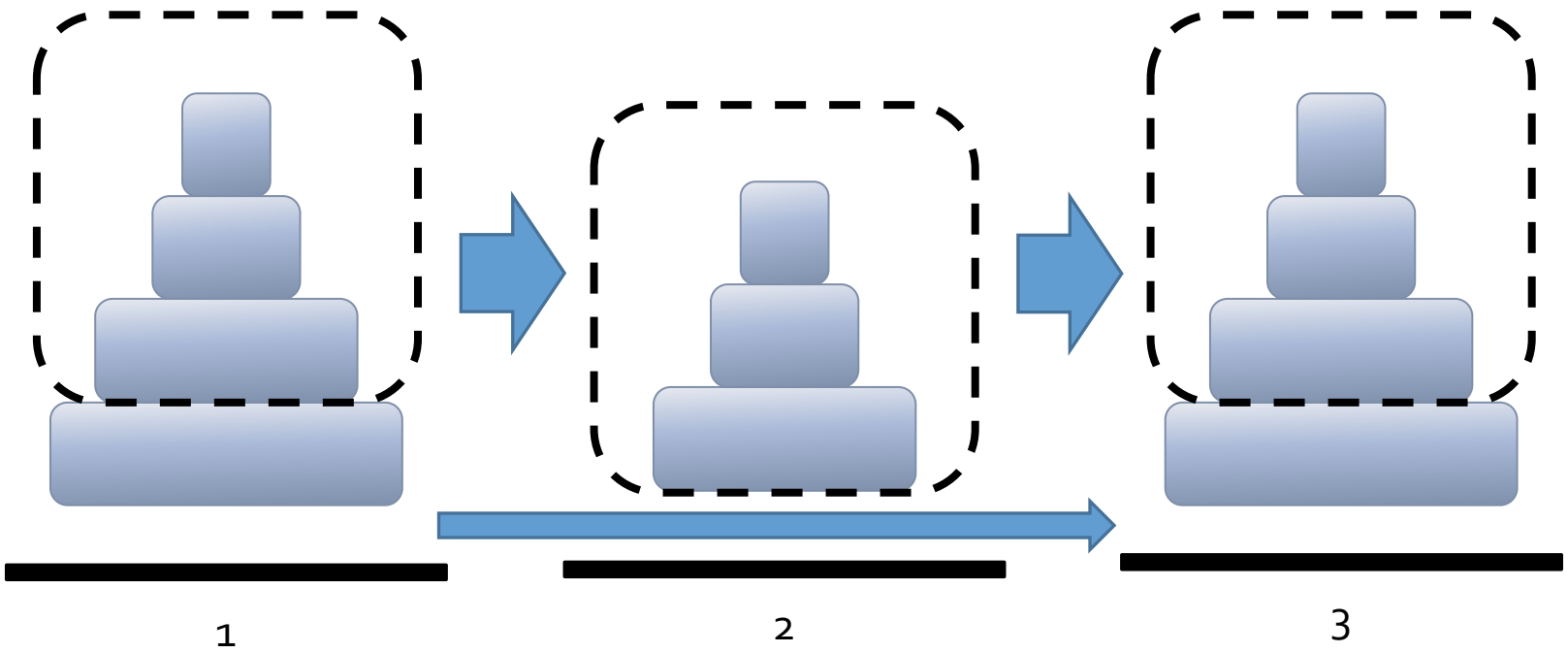
例子1：河內塔

Recursive Case:

n>1時:

1. n-1個盤子, 從柱子1移到柱子2

3. n-1個盤子, 從柱子2移到柱子3



2. 把最大的盤子從柱子1移到柱子3

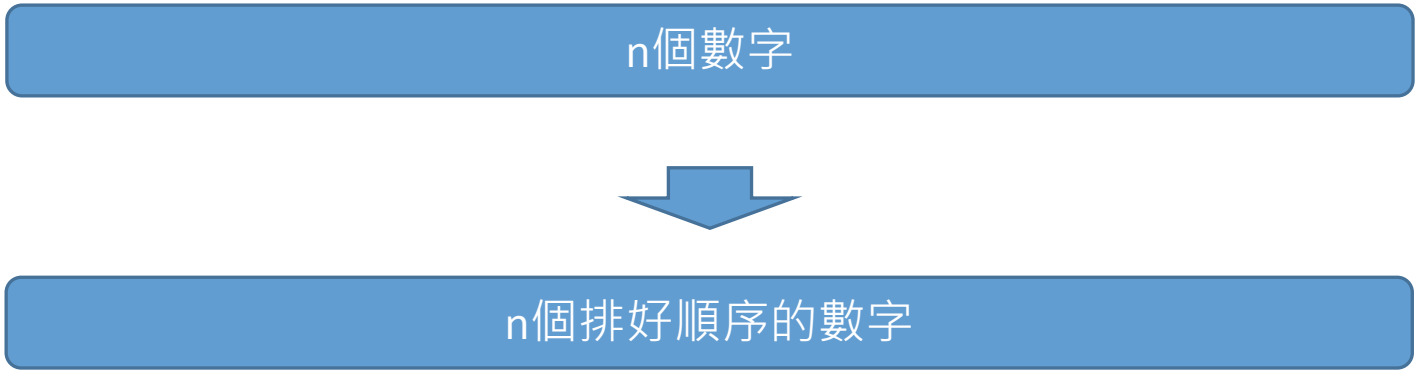
例子1：河內塔

- Divide在哪裡?
- 原本: n個盤子從柱子1移到柱子3
- 分成:
  1. n-1個盤子從柱子1移到柱子2
  2. 1個盤子從柱子1移到柱子3
  3. n-1個盤子從柱子2移到柱子3

- Combine在哪裡?
  - 這個例子不需要額外combine

例子2: Merge Sort

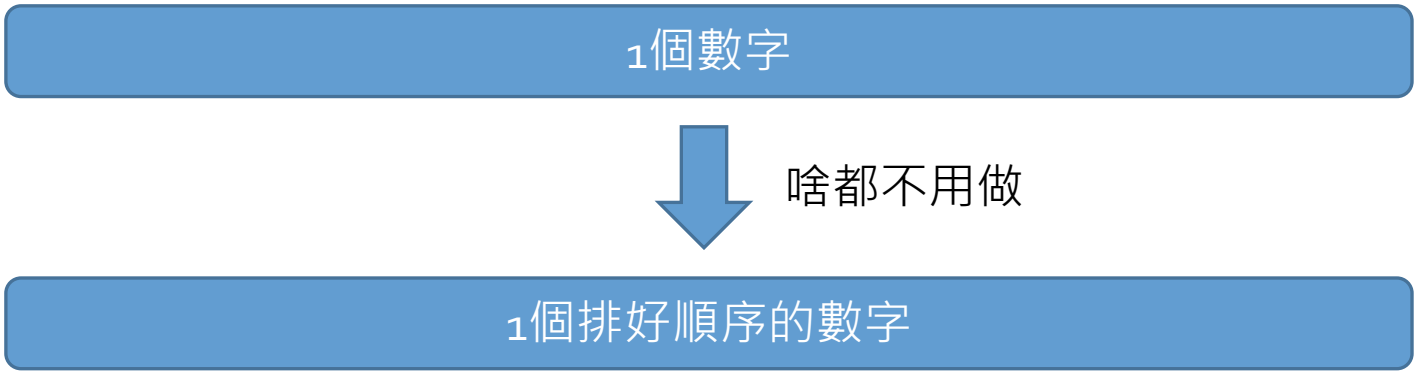
- Input: n個數字
- Output: 照順序由大排到小



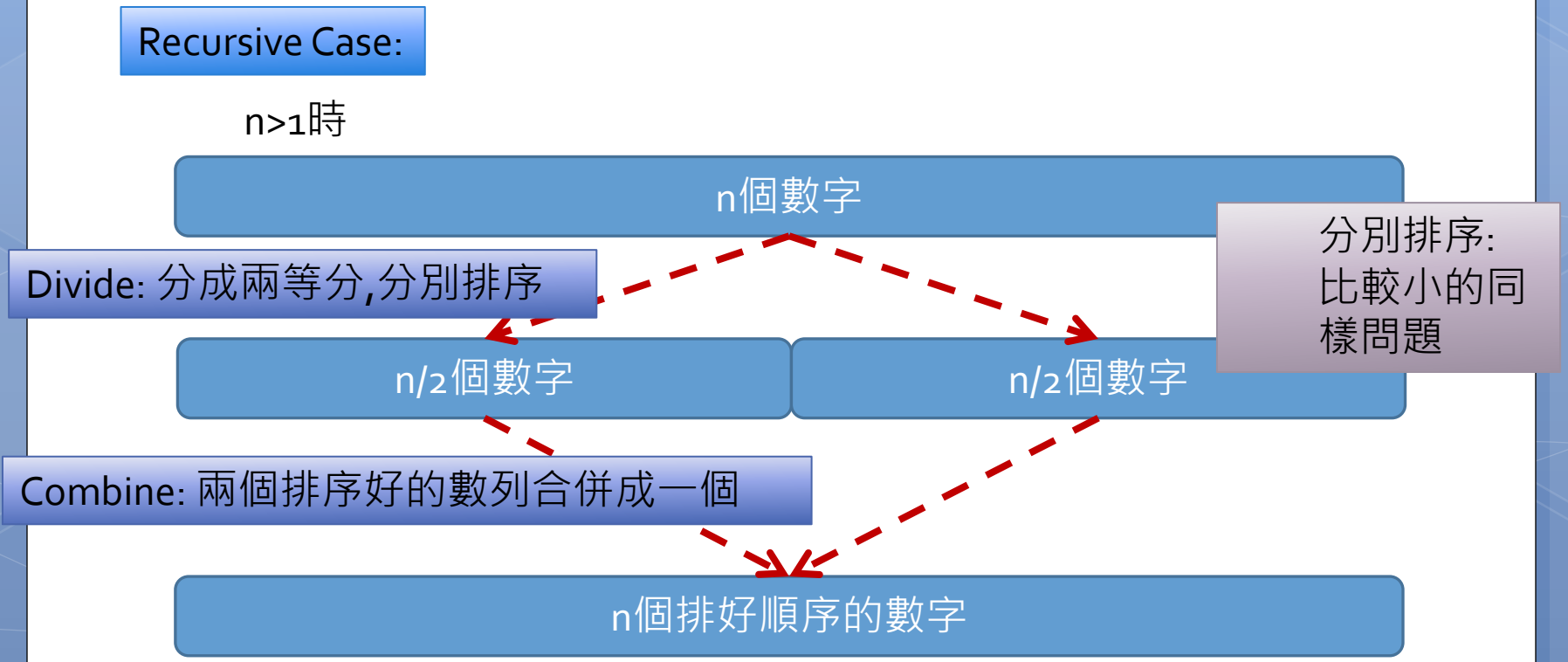
例子2: Merge Sort

Base Case:

n=1時, 沒有排序的問題, 直接輸出



例子2: Merge Sort



例子2: Merge Sort

- Divide在哪裡?
- 原本: 把n個數字排序
- 分成:
  - 2個 (n/2個數字排序)
- Combine在哪裡?
  - 把兩個排好的數列合併成一個數列

Recurrences

- 計算divide-and-conquer的演算法執行時間
  →用遞迴式最自然

- 例1. 河內塔, 移動n個盤子: $T(n)$
- 如果$n = 1$的話, 直接移動過去. $T(1) = \Theta(1)$
- 如果$n > 1$的話,分為以下步驟:
  1. n-1個盤子從柱子1移到柱子2 $T(n-1)$
  2. 1個盤子從柱子1移到柱子3 $T(1)$
  3. n-1個盤子從柱子2移到柱子3 $T(n-1)$

$$T(n) = \begin{cases} \Theta(1) & \text{,if } n = 1 \\ 2T(n-1) + T(1) & \text{,if } n > 1 \end{cases}$$

Recurrences

- 例2. Merge Sort, n個數字排序: $T(n)$
- 如果n=1時: 直接輸出. $\Theta(1)$
- 如果n>1時:
  - 分成2個 (n/2個數字排序) $2\,T(n/2)$
  - 把兩個排好的數列合併成一個數列 $\Theta(n)$

$$T(n) = \begin{cases} \Theta(1) & \text{,if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{,if } n > 1 \end{cases}$$

細節

- 例: Merge Sort的n不是偶數時, 就會變成

$$T(n) = \begin{cases} \Theta(1) & ,\text{if } n = 1 \\ T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) & ,\text{if } n > 1 \end{cases}$$

- 通常我們卻很豪爽的使用

$$T(n) = \begin{cases} \Theta(1) & ,\text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & ,\text{if } n > 1 \end{cases}$$

- 甚至

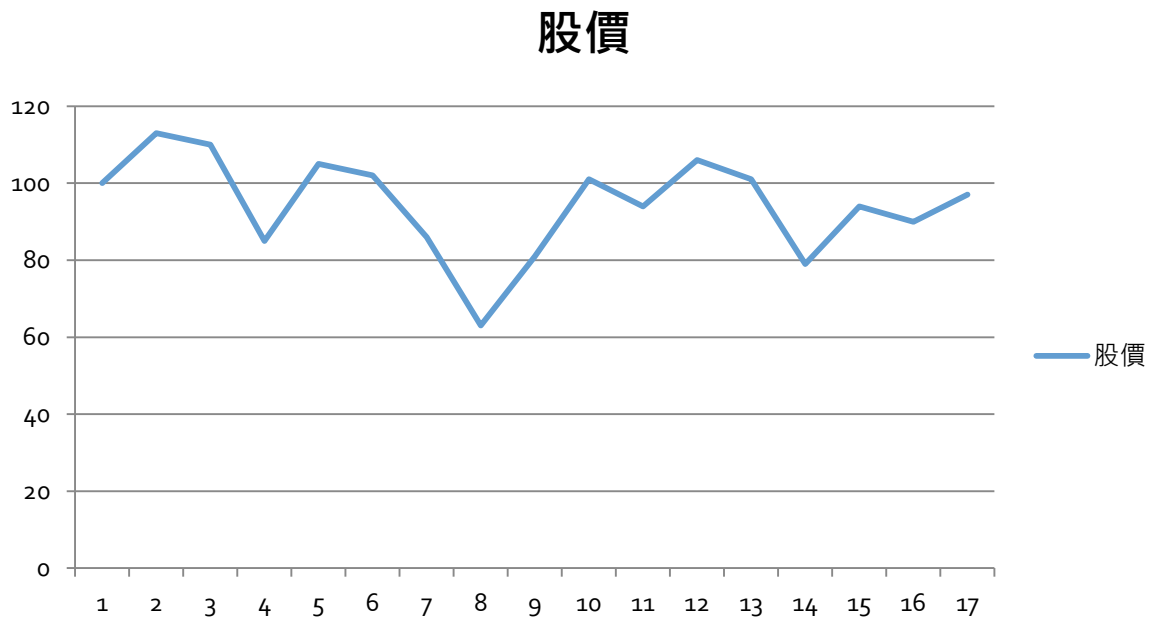
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

假設: (大部分時候都成立)

1. Ceiling & floor functions 不影響recurrence的解
2. Boundary case - n很小的時候通常 execution time=constant (不一定只是n=1時)

股市大亨

- 菜瓜布股份有限公司股票股價



- 未卜先知, 已知未來的股價走勢(內線?)
- 問: 如何找出可以使獲利最大的買進賣出時機?

股市大亨:嘗試一

- 嘗試一: 有沒有什麼絕招? (Θ(1)的方法)
- 找最低點當買入點, 往後找之後的最高點當賣出點
- 找最高點當賣出點, 往前找之前的對低點當買入點
- 以上找出的是否為正確解?

- 答:否.

股市大亨:嘗試二

- 嘗試二: 暴力法
- 不用大腦的方法
- 每種可能性都試試看 (窮舉法)
- 如此的話要花多少時間?
- 有幾種可能性: $\binom{n}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$
- 就算每種可能性都只花$O(1)$也是要$\Omega(n^2)$
- 能不能更好?

股市大亨:嘗試三

- 嘗試三: Divide-and-Conquer的方法
- 首先先把原本的問題稍微轉換

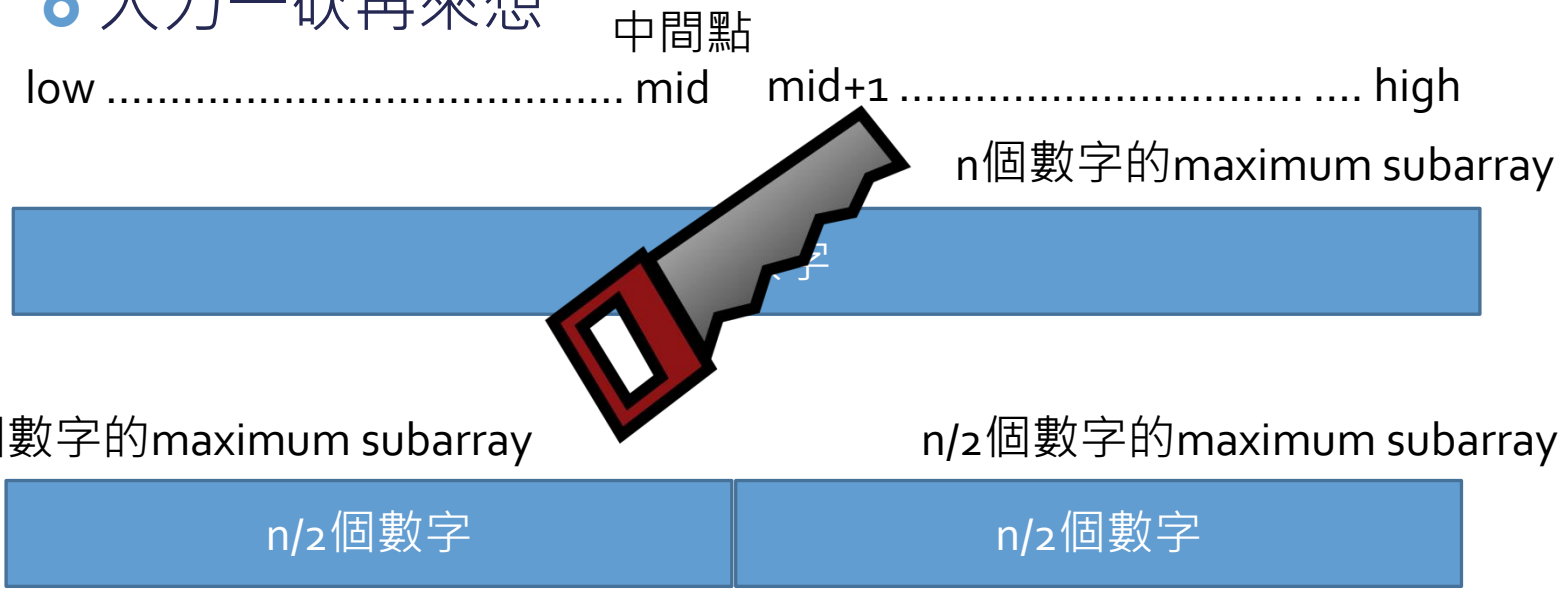
Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Δ		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

- 題目變成在Δ一列中找出此一數列的一連續子數列, 使其總合為最大, 又稱Maximum Subarray Problem

股市大亨：嘗試三

Recursive Case:

- 大刀一砍再來想



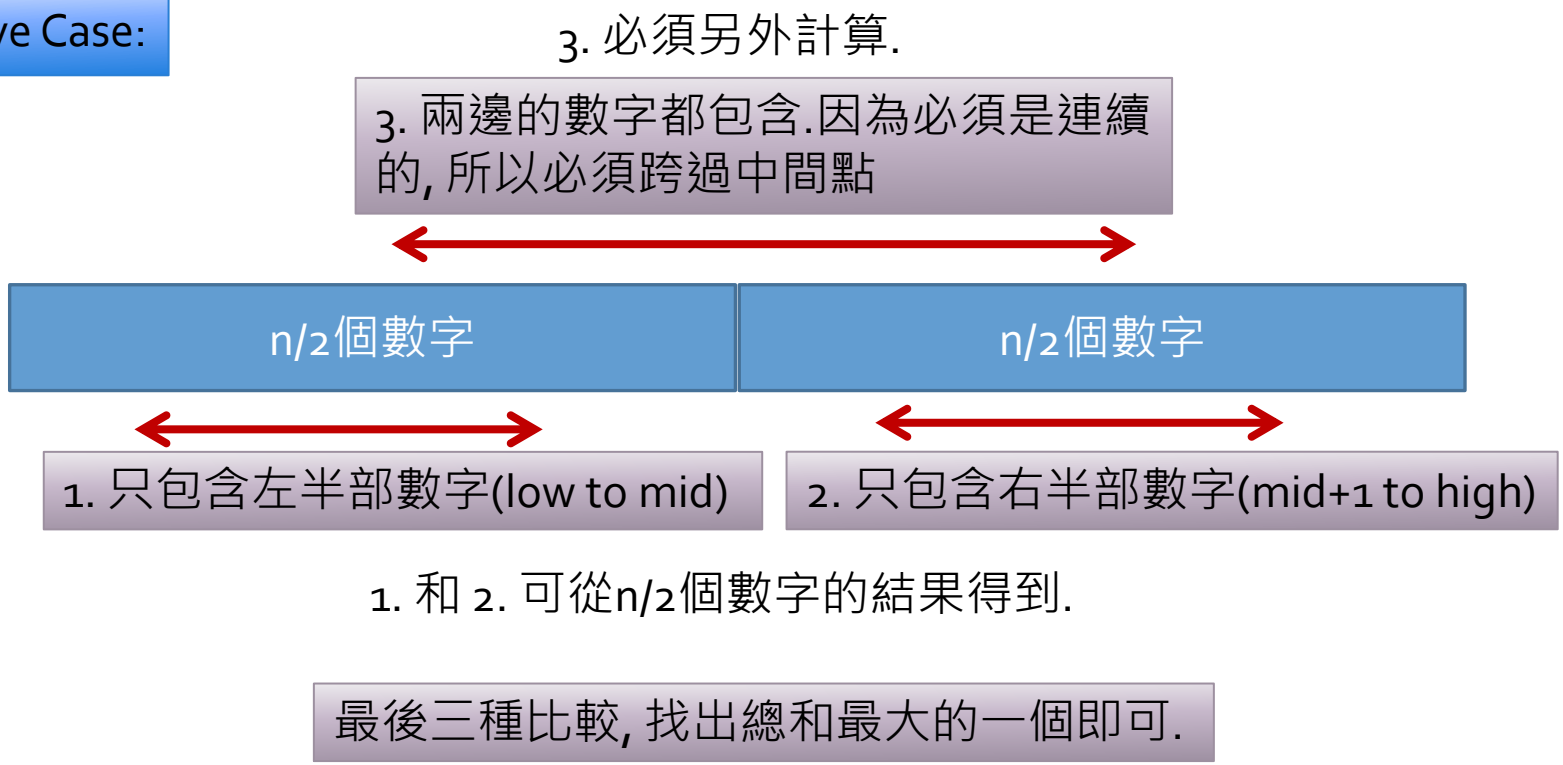
假設可以找到兩個n/2大小的maximum subarray

如何找到n個數字的maximum subarray?

股市大亨：嘗試三

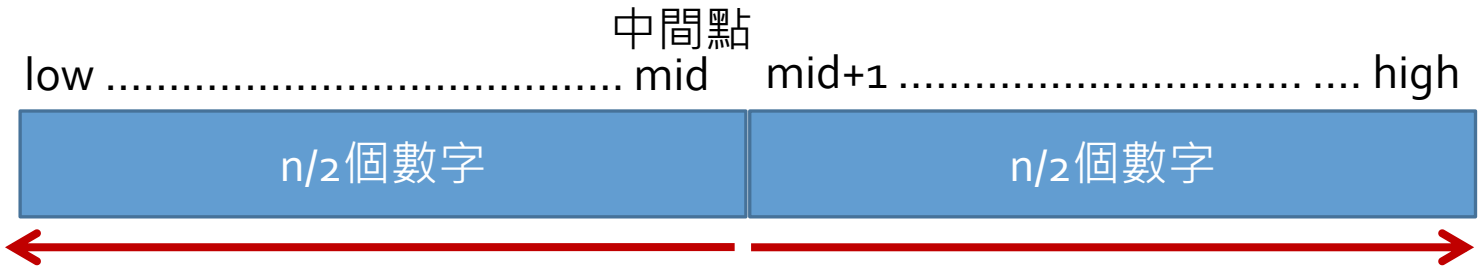
- Maximum subarray可能出現的情形:

Recursive Case:



股市大亨：嘗試三

- 3. 如何找出包含中間點的maximum subarray呢?



(1) 尋找以中點開始, 左邊的 maximum subarray

(2) 尋找以中點開始, 右邊的 maximum subarray

(3) 合併(1)和(2)即為包含中間點的maximum subarray

所花時間? $\Theta(n)$

股市大亨：嘗試三

Base Case:

- n=1的時候
- maximum subarray?
- 就是它自己.

股市大亨 之 酥多扣的(pseudo-code)

```
Find_Max_Crossing_Subarray(A, low, mid, high)
left_sum=-∞
sum=0
for i=mid downto low
        sum=sum+A[i]
        if sum>left_sum
                left_sum=sum
                max_left=i
right_sum=-∞

sum=0
for j=mid+1 to high
        sum=sum+A[i]
        if sum>right_sum
                right_sum=sum
                max_right=j
return (max_left, max_right, left_sum+right_sum)
```

A: array本身
low: array最小的index
mid: 左半部array的最大index
high: array最大的index

股市大亨 之 酥多扣的(pseudo-code)

```
Find_Maximum_Subarray(A,low,high)
if high==low
        return (low,high,A[low])
else
        mid=⌊(low + high)/2⌋
        (left_low,left_high,left_sum)=Find_Maximum_Subarray(A,low,mid)
        (right_low,right_high,right_sum)=Find_Maximum_Subarray(A,mid+1,h
igh)
        (cross_low,cross_high,cross_sum)=Find_Max_Crossing_Subarray(A,lo
w,mid,high)

if left_sum>=right_sum and left_sum>=cross_sum
        return (left_low,left_high,left_sum)
else if right_sum>=left_sum and right_sum>=cross_sum
        return (right_low,right_high,right_sum)
else
        return (cross_low,cross_high,cross_sum)
```

Base Case

Recursive Case

Conquer

Divide

Combine

執行時間分析

- n個數字找max-subarray $T(n)$
- Base case: n=1的時候直接return. $T(1) = \Theta(1)$
- Recursive case:
  - Divide: 2個 n/2個數字找max subarray $2T(n/2)$
  - Combine:
    - 1. 確認n>1, 計算中間點位置等等 $\Theta(1)$
    - 2. 尋找通過mid的max subarray $\Theta(n)$
    - 3. 比較三個max subarray的大小決定最後結果 $\Theta(1)$

$$T(n) = \Theta(1) + 2T\left(\frac{n}{2}\right) + \Theta(n) + \Theta(1) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$T(n) = \begin{cases} \Theta(1) & ,\text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & ,\text{if } n > 1 \end{cases}$$

解: $\mathrm{T(n)} = \Theta(n \log n)$

Today's Reading Assignment

- Cormen ch 4 – 4.1

下次...

- 其他Divide-and-Conquer的例子
  - 矩陣相乘
  - 找中位數
- 如何解遞迴式