Design of Algorithms by Induction-2

Design of Algorithms

- Using some well-know strategies
  - greedy method
  - dynamic programming
  - divide and conquer
  - …
- Using induction

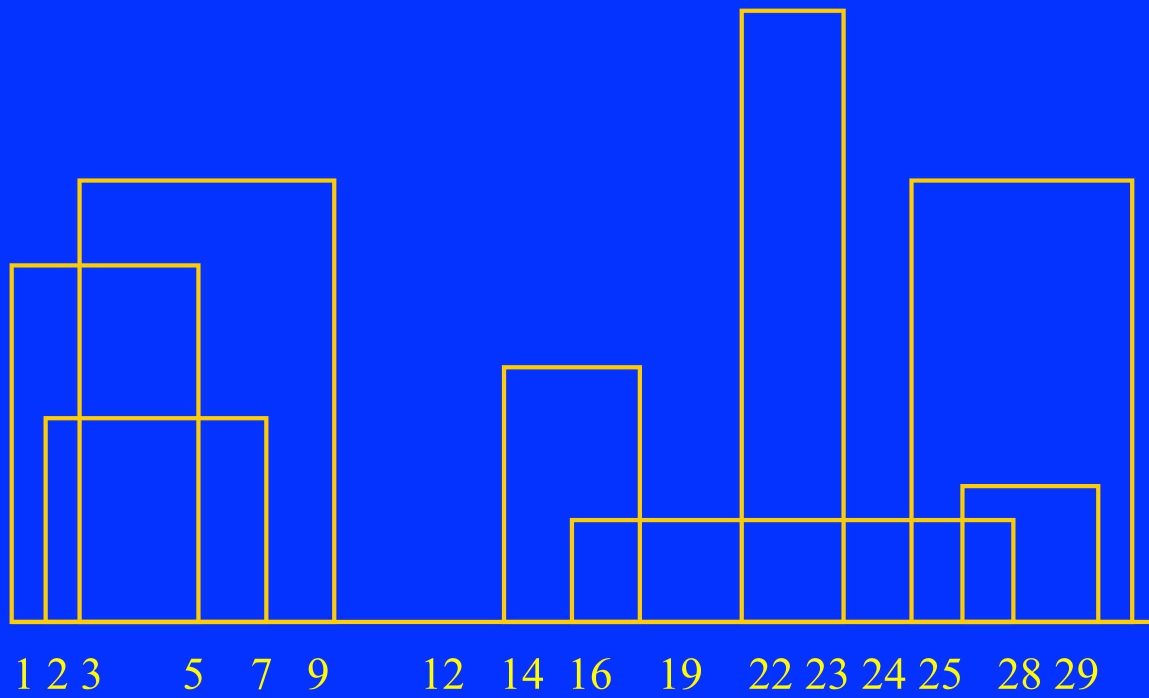
  base: solve a small instance of problem

  induction: a solution to every problem
  can be constructed from
  solutions of smaller problems.

The Skyline Problem

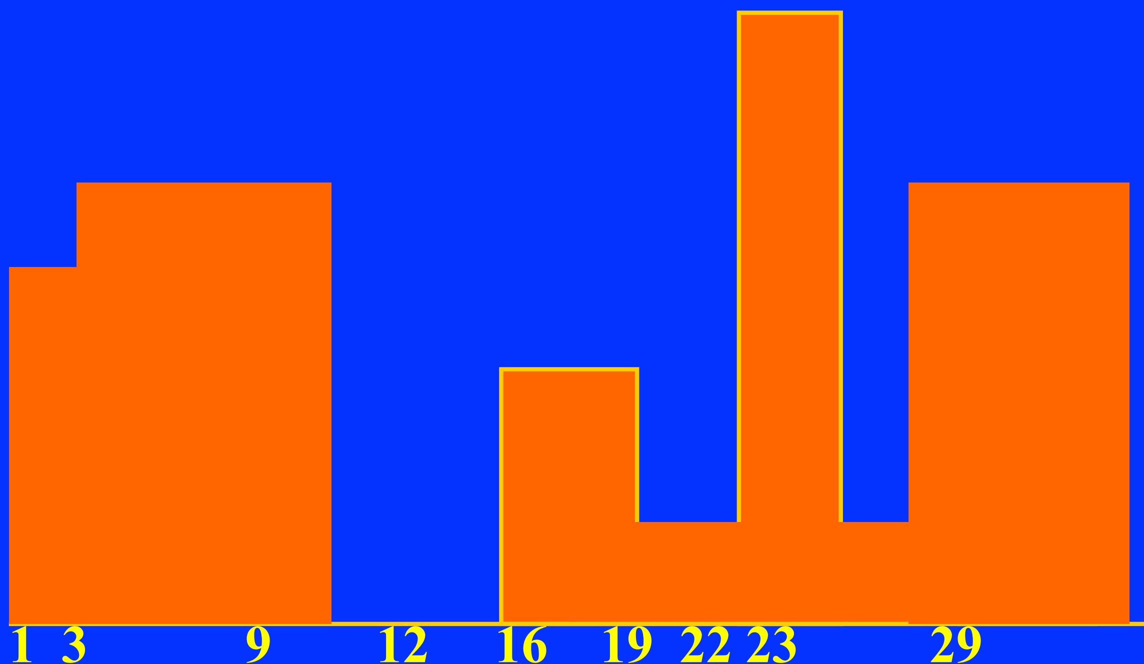
The Skyline Problem

- Given exact locations & shape of rectangular buildings
  Draw skyline of these building, eliminating hidden lines



(左端,	高度,	右端)
(1,	11,	5)
(2,	6,	7)
(3,	13,	9)
(12,	7,	16)
(14,	3,	25)
(19,	18,	22)
(23,	13,	29)
(24,	4,	28)

(1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29,0)



直接透過數字(而不用畫圖)，求解？
(2,10, 9), (3, 15, 7), (5 12 12), (15, 10, 20), (19, 8, 24)
的Skyline？

Thinking

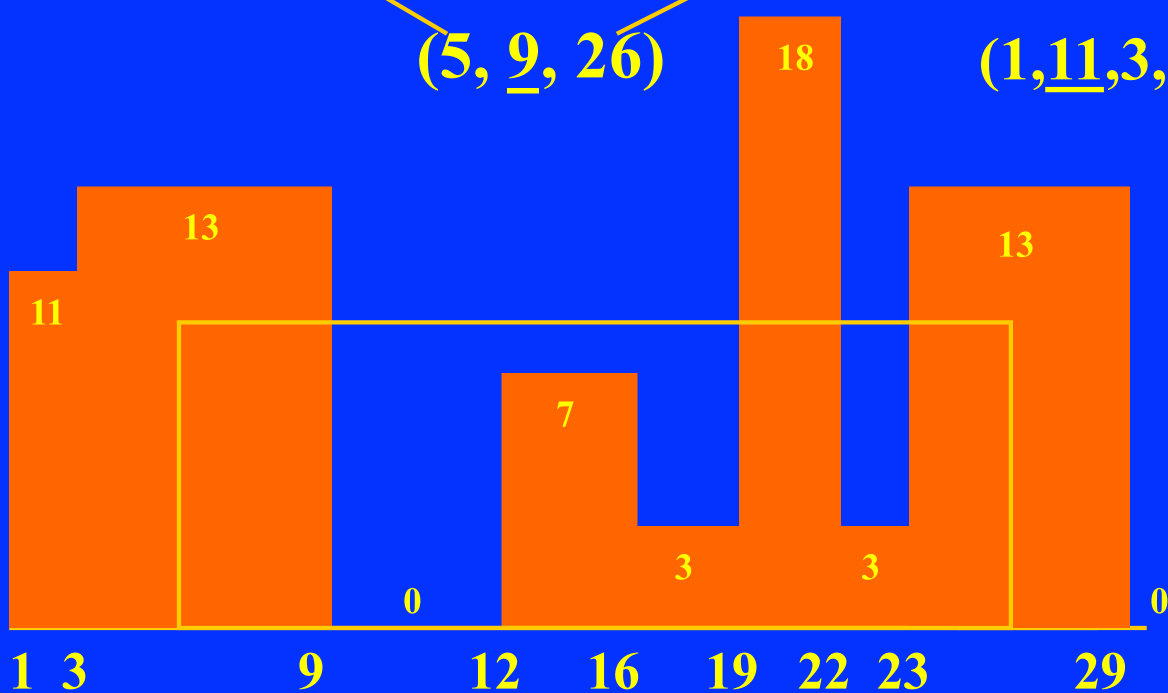
- Hypothesis: we know how to solve for n-1 building
- Induction: add the nth building (O(n)), totally O(n^2)

(1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29,0)

+

(5, 9, 26)

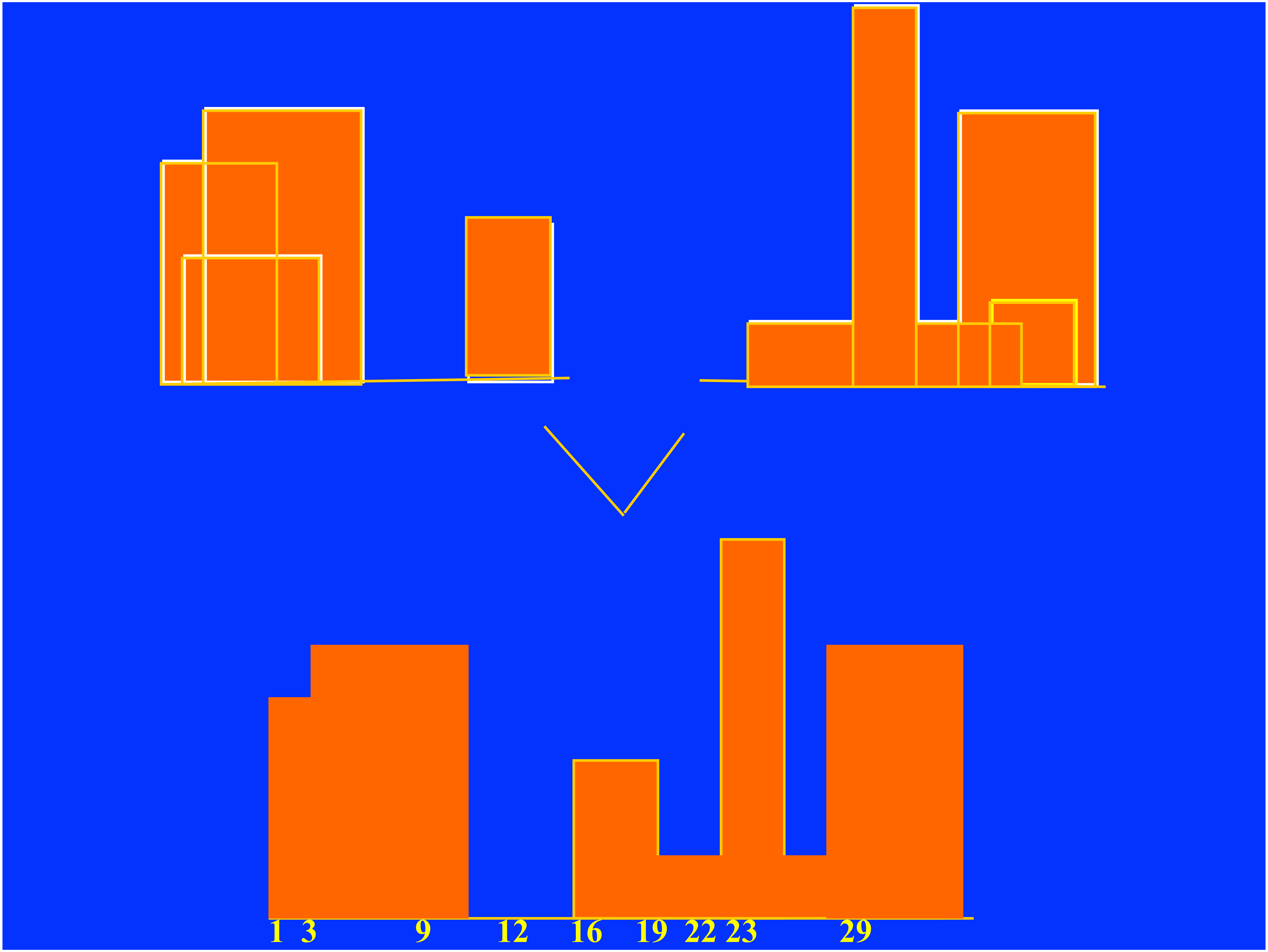
(1,11,3,13,9, 9,19,18,22,9,23,13,29,0)

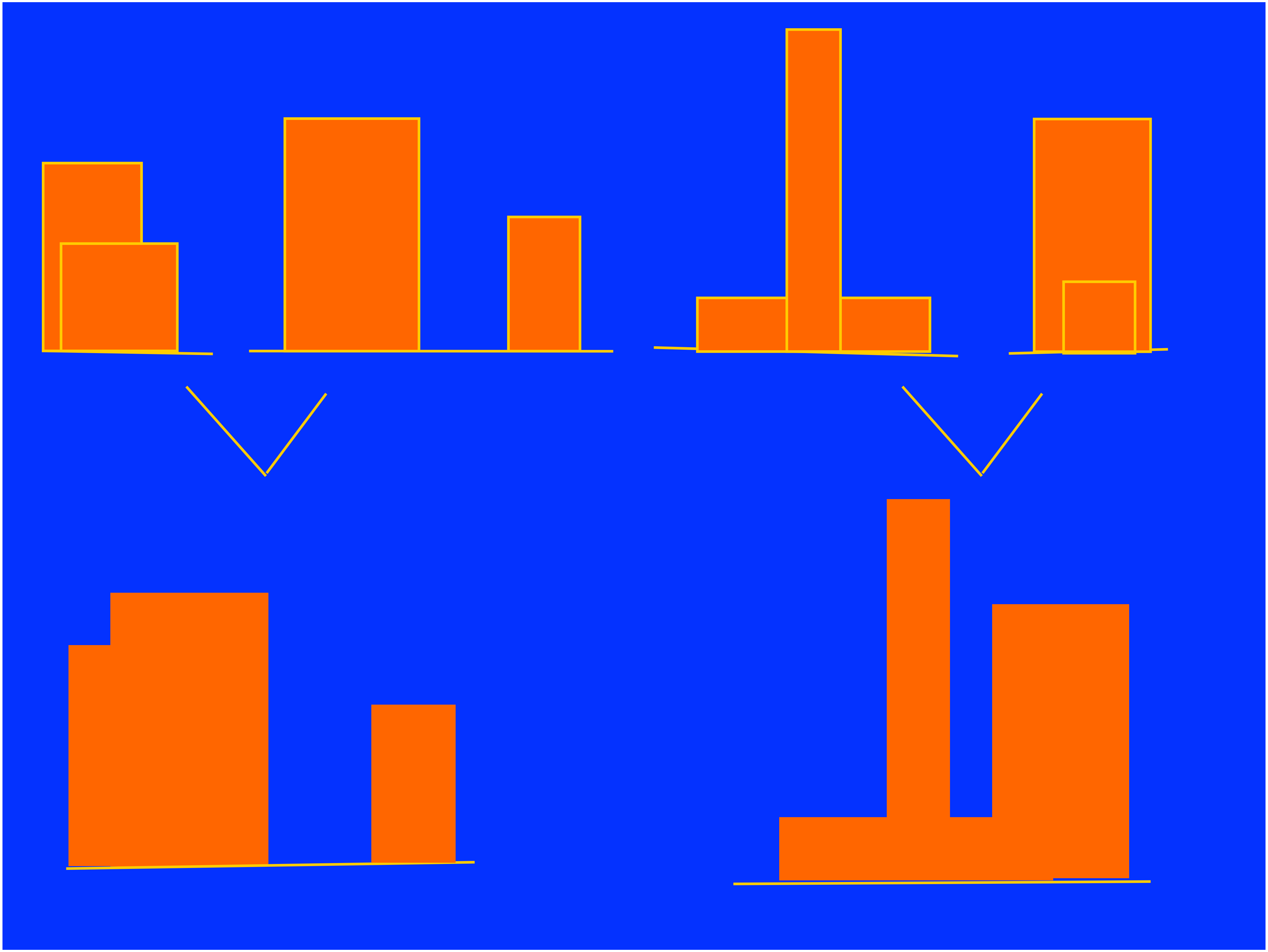


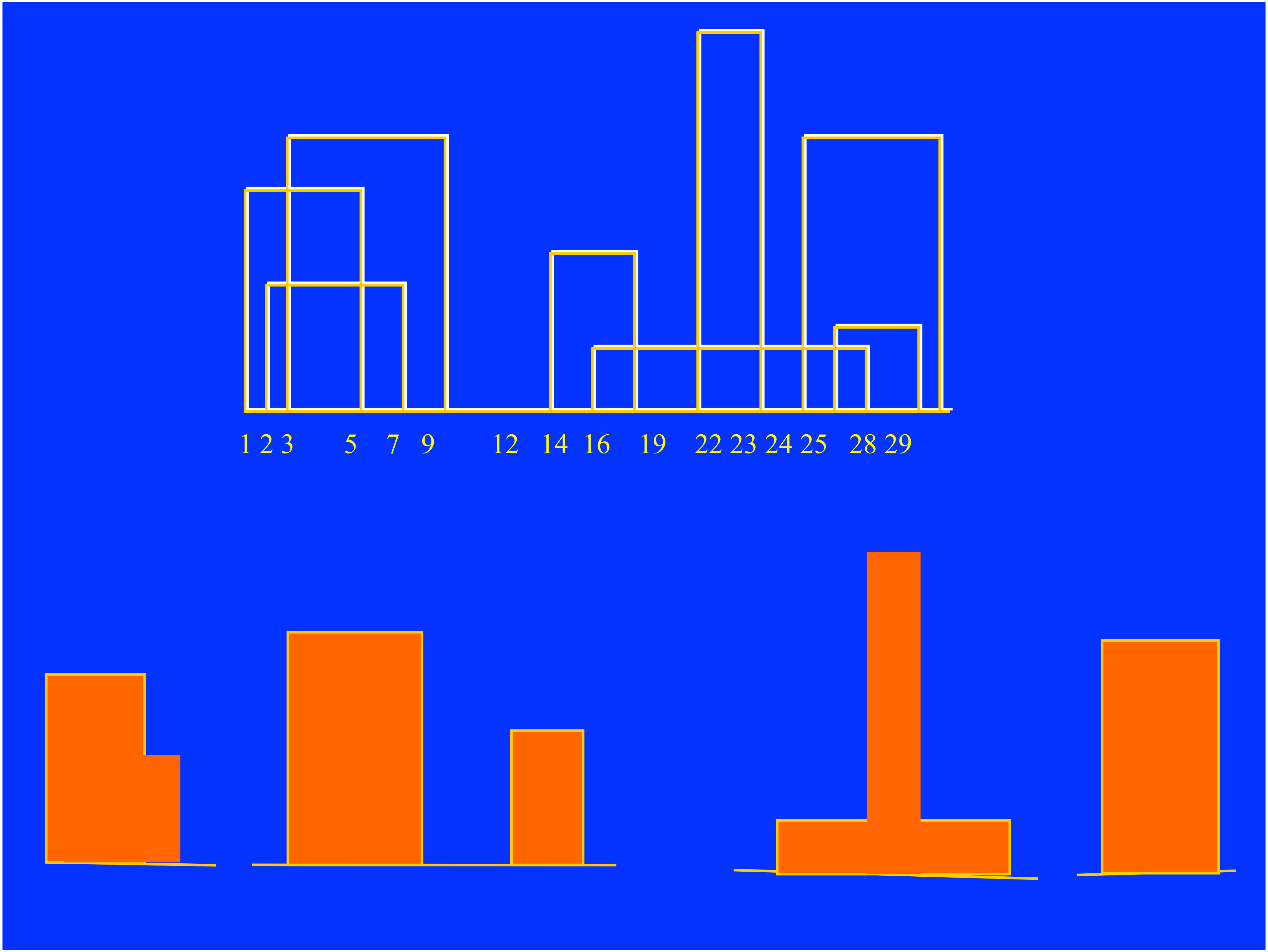
O(nlogn)的演算法來找出Skyline?

Induction

- Hypothesis: we know how to solve for n/2 buildings
- Induction: from n/2 to n
  - merge two n/2 skylines: similar to add one building
  - merge: O(n)
- Algorithm
  - similar to merge sort
  - complexity: $T(n)=2*T(n/2)+O(n) = O(nlogn)$


1 3 9 12 16 19 22 23 29




1 2 3 5 7 9 12 14 16 19 22 23 24 25 28 29

Time Comlexity Analysis

- **Recurrence Relation: T(1)=1, T(N)=2T(N/2)+N**
- **Solution 1**

$$\frac{T(N)}{N} = \frac{T(N/2)}{N/2} + 1$$

$$\frac{T(N/2)}{N/2} = \frac{T(N/4)}{N/4} + 1$$

$$\frac{T(N/4)}{N/4} = \frac{T(N/8)}{N/8} + 1$$

$$\cdots$$

$$\frac{T(2)}{2} = \frac{T(1)}{1} + 1$$

$$\Rightarrow \frac{T(N)}{N} = \frac{T(1)}{1} + \log N$$

$$\boxed{\Rightarrow T(N) = N \log N + N = O(N \log N)}$$

Time Comlexity Analysis

- **Recurrence Relation: T(1)=1, T(N)=2T(N/2)+N**
- **Solution 2**

$$T(N) = 2T(N/2) + N$$

$$\because 2T(N/2) = 2(2(T(N/4)) + N/2) = 4T(N/4) + N$$

$$\Rightarrow T(N) = 4T(N/4) + 2N$$

$$\because 4T(N/4) = 4(2(T(N/8)) + N/4) = 8T(N/8) + N$$

$$\Rightarrow T(N) = 8T(N/8) + 3N$$

$$\ldots$$

$$\Rightarrow T(N) = 2^k T(N/2^k) + k * N$$

Using k =log$N$

$$\Rightarrow \boldsymbol{T(N) = N T(1) + N \log N = N \log N + N = O(N \log N)}$$

Skyline Divide & Conquer的Merge
為何是 O(n)？

Comment

- divide and conquer
- merging two skylines is more general than adding one building into a skyline
- If the algorithm includes a step that is more general than required, consider applying this step to a more complicated part.

Skyline Divide & Conquer的Algorithm
需要先將input building依照座標排序嗎？

Computing Balance Factors

In a Binary Tree

Computing Balance Factors in Binary Tree

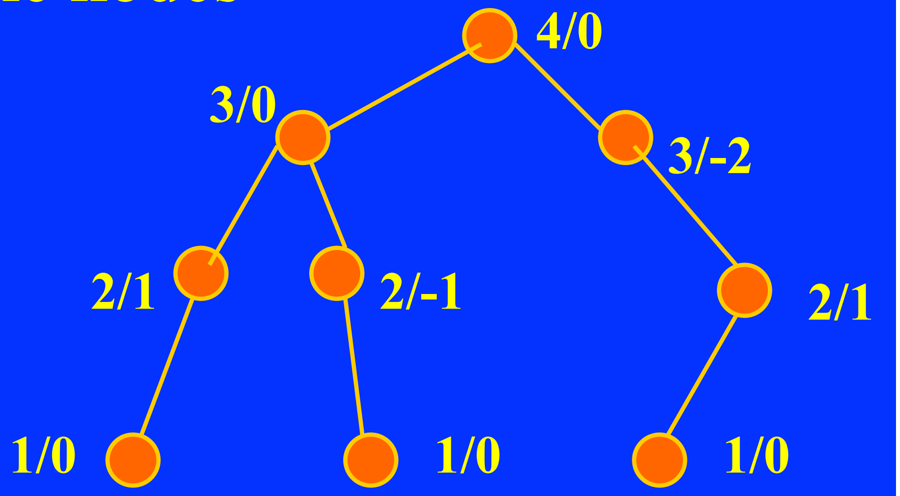
- Height: H(v)
- Balance factor: B(v)=|H(vl)-H(vr)|,

  * vl, vr: left, right children of v

  * AVL tree: balance factors of -1, 0, 1
- Problem

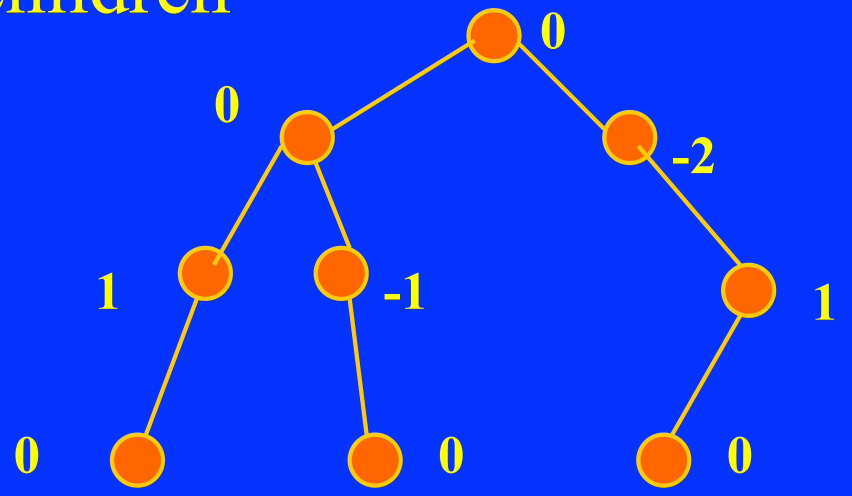
  Given a binary tree T with n nodes

  Compute the balance factors of all the nodes



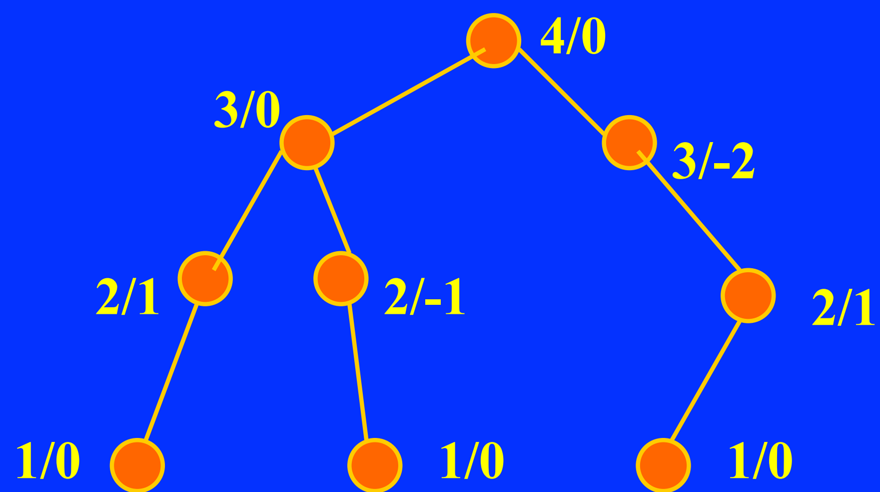
Thinking

- Hypothesis: we know how to balance factors of all nodes in trees that have < n nodes
- Induction: from < n nodes to nodes tree
  - remove root nodes to < nodes
  - But, root's balance factor depends not on the balance factors of root's children but rather on their height



Induction

- Hypothesis: we know how to compute balance factors & heights of all nodes in trees that have < n nodes
- Induction: from < n nodes to n nodes
  - Base
  - root:
    - calculate difference between heights of children
    - height: maximal height of two children + 1



Comment

- Comment
  - sometimes, solving a stronger problem is easier
    (stronger problem: balance factor + height)
  - if solution is broader, induction step may be easier

Find Maximum Consecutive Subsequence

Finding Maximum Consecutive Subsequence

- **Subsequence (of consecutive elements)**

  e.g. (3, -2, -3) is a subsequence of (2, -3, 1.5, -1, 3, -2, -3, 3)

- **Maximum subsequence: maximum sum of subsequence**

  e.g. maximum subsequence of (2, -3, <u>1.5, -1, 3</u>, -2, -3, 3)=(1.5, -1, 3)

- **Problem**

  <u>Given</u> a sequence $x_1, x_2, \ldots, x_n$ of real numbers

  <u>find</u> a subsequence $x_i, x_{i+1}, \ldots, x_j$

  <u>such that</u> sum of the numbers in it is maximum over all subsequence of consecutive elements

$(-2, 1, -3, 4, -1, 2, 1, -5, 4)$

的Maximum Consecutive Subsequence？

Brute Force Approach

- (2, -3, 1.5, -1, 3, -2, -3, 3)
- generate all subsequences & check the sums
  - size 1: (2), (-3), (1.5), (-1), (3), (-2), (-3), (3)
  - size 2: (2,-3), (-3,1.5), (1.5,-1), (-1,3), (3,-2), (-2,-3), (-3,3)
  - size 3: (2,-3,1.5), (-3,1.5,-1), (1.5,-1,3), (-1,3,-2), (3,-2,-3),(-2,-3,3)
  - size 4: (2,-3,1.5,-1), (-3,1.5,-1,3), (1.5,-1,3,-2), (-1,3,-2,-3),(3,-2,-3,3)
  - size 5: (2, -3, 1.5, -1, 3, -2, -3), (-3, 1.5, -1, 3, -2, -3, 3)
  - size 6: (2, -3, 1.5, -1, 3, -2, -3, 3)

<u>Improvement of</u>
<u>Brute Force Approach</u>

- **(2, -3, <u>1.5, -1, 3</u>, -2, -3, 3)**
- **generate all subsequences with positive boundaries & check the sums**
  - **size 1: (2), (1.5), (3), (3)**
  - **size 2:**
  - **size 3: (2,-3,1.5), (1.5,-1,3),**
  - **size 4: (3,-2,-3,3)**
  - **size 5:**
  - **size 6: (2, -3, 1.5, -1, 3, -2, -3, 3)**

Thinking

- Hypothesis: we know how to find maximum subsequence in sequence of size (n-1)
- Induction: from $S' = (x_1, x_2, \ldots, x_{n-1})$ to $(x_1, x_2, \ldots, x_n)$
  - Case

(1) maximum subsequence in S' is empty: consider x_n only

(2) assume solution in S' is $S'_M = (x_i, x_{i+1}, \ldots, x_j)$

if $j = n-1$ (i.e., S'_M is a suffix of S'_M)

if x_n is positive $S_M = (x_i, x_{i+1}, \ldots, x_{n-1}, x_n)$

else negative $S_M = S'_M$

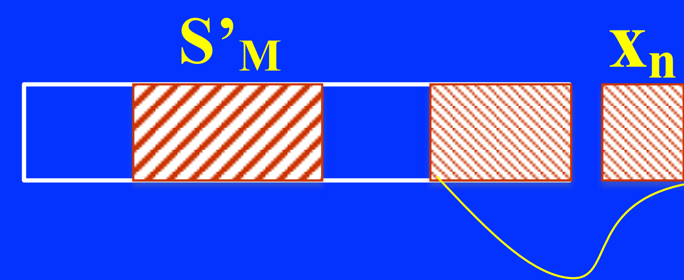
else two possibilities

(a) $S_M = S'_M$

(b) there is another subsequence,

which is not maximum in S'

but is maximum in S when x_n is added to it

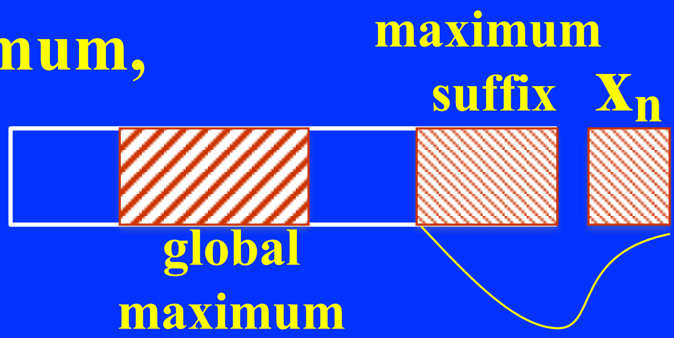


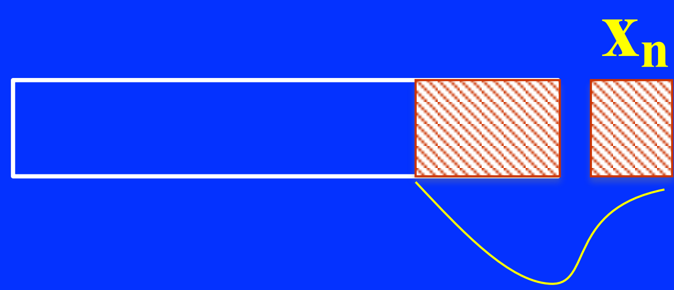
求解O(n)的演算法來找出Maximum Consecutive Subsequence？

Prefix 字首, 前綴
Suffix 字尾, 後綴

Induction

- Hypothesis: we know how to find, in sequence of size < n
  - (1) maximum subsequence overall (global maximum)
  - (2) maximum subsequence that is a suffix (maximum suffix)
- Induction
  - If (x_n + maximum suffix) > global maximum,
    - new global
  - Else retain previous global
  - maintain maximum suffix
  - If maximum suffix + x_n <= 0,
    - maximum suffix is empty
  - Else (maximum suffix + x_n > 0)
    - maximum suffix = maximum suffix + x_n





```
Algorithm Maximum_Consecutive_Subsequence(X,n)
Input: X (an array of size n)
Output: Global_max
Begin
    Global_max:=0;
    Suffix_Max:=0;
    For i:=1 to n do
        If x[i] + Suffix_Max > Global_Max then
            Suffix_Max:= Suffix_Max+ x[i];
            Global_Max := Suffix_Max;
        else if x[i]+Suffix_Max > 0 then
                Suffix_Max := x[i] + Suffix_Max
              else Suffix_Max:=0
End
```

	2,	-3,	1.5,	-1,	3,	-2,	-3,	3
max_suffix	**0**	**2**	**0**	**1.5**	**0.5**	**3.5**	**1.5**	**0**	**3**
global_max	**0**	**2**	**2**	**2**	**2**	**3.5**	**3.5**	**3.5**	**3.5**

Strengthening Induction Hypothesis

- $P(<n) \Rightarrow P(n)$
- Strengthening

  $[P \& Q](<n) \Rightarrow [P \& Q](n)$

  e.g. [Global_Max & Suffix_Max] $(<n) \Rightarrow$

  [Global_Max & Suffix_Max] (n)

Find Maximum Sum Consecutive Subsequence
有何應用？

(8, 6, 7, 4, 8, 7, 9, 10, 5, 9)

何時買進，何時賣出，獲利最大?

(8, 6, 7, 4, 8, 7, 9, 10, 5, 9)

$(-2, 1, -3, 4, -1, 2, 1, -5, 4)$

的Maximum Consecutive Subsequence **?**

如何求解有長度限制的
Maximum Sum
Consecutive Subsequence ?

求解O(n)的演算法來找出
Maximum Product
Consecutive Subsequence ?