Review and Terminologies

Outline

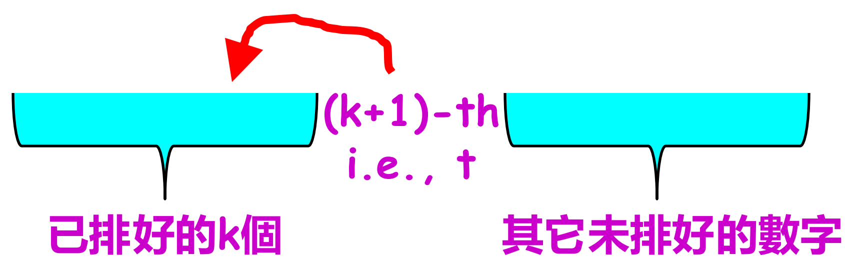
- Complexity
- Tree and search tree
- Graph

複雜度 (Complexity)

Insertion Sort
(不同的sort有不同的稱呼，請謹慎使用)

```
for (i = 1; i < n; i++)
{/* insert a[i] into a[0:i-1] */
    int t = a[i];
    int j;
    for (j = i - 1; j >= 0 && t < a[j]; j--)
        a[j + 1] = a[j];
    a[j + 1] = t;
}
```

想法：先排好前k個數字，接著決定(k+1)-th數字要插入在所有k+1個數字當中的哪裡，使完成k+1個數字的排列



Complexity的計算

- 針對程式執行的time and/or 或使用的記憶體space所做的估算測量 (在程度上)
- Time (常見)
  - 計數演算法當中一個有 “代表性” 的操作上的次數 (代表性操作往往是最耗費時間或空間的操作，它應該出現在程式最花時間或空間的地方)
  - 代表性操作視演算法不同而不同
  - Insertion sort: 比較的操作

Comparison Count or Data Movement?

```
for (j = i - 1; j >= 0 && t < a[j]; j--)
        a[j + 1] = a[j];
```

這兩個數量的數值基本相當
因此看當中一個就好了...

Comparison Count

- Worst-case count = maximum count
  - 我們最感興趣的數字之一
  - 但有時沒多大參考價值
- Best-case count = minimum count
  - 很少用
- Average count
  - Expectation
  - Variance (or deviation in general) 我們也關心
  - 或，tail probability

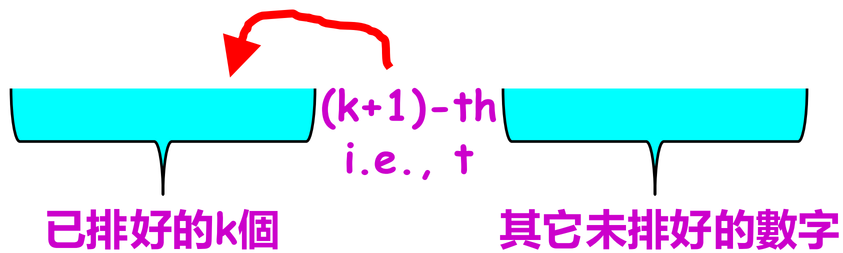
Worst-Case Comparison Count

```
for (j = i - 1; j >= 0 && t < a[j]; j--)
        a[j + 1] = a[j];
```

a = [1, 2, 3, 4] and t = 0 => 4 compares

a = [1,2,3,...,i] and t = 0 => i compares

Worst-Case Comparison Count (Here, n represents 資料量)



總共n筆資料

$$\text{total compares} = 1 + 2 + 3 + \dots + (n-1)$$
$$= (n-1)n/2 = \tfrac{1}{2}(n^2) - \tfrac{1}{2}(n)$$

A list in decreasing order

Asymptotic Complexity for Insertion Sort

- 關鍵因子，造成整體數量大小的關鍵
- $O(n^2)$
  - O: big-O (讀法) 其嚴謹的定義請上演算法的課
  - 考慮如下: $\frac{1}{2}(n^2)-\frac{1}{2}(n)$ vs n^2+3n-4
    - Both are $O(n^2)$

某一個不知名的sorting algorithm

Worst- vs Best-case Complexity for Insertion Sort

- Worst-case time is $O(n^2)$
- Best-case is $O(n)$
  - Why?
  - Increasing order

相異資料量n下
不同 Complexities的直觀

n	n	nlogn	n^2	n^3
1000	1mic	10mic	1milli	1sec
10000	10mic	130mic	100milli	17min
10^6	1milli	20milli	17min	32years

complexities

這種複雜度令人難以接受

n	n^4	n^{10}	2^n
1000	17min	3.2×10^{13} years	3.2×10^{283} years
10000	116 days	???	???
10^6	3×10^7 years	??????	??????

complexities

Solution Space (解空間)

假設一位農民有一片面積為 L 公頃的農田，可以種植小麥或大麥，或者兩者的組合。農民擁有 F 千克的肥料和 P 千克的農藥。每公頃小麥需要 F_1 千克肥料和 P_1 千克農藥，而每公頃大麥需要 F_2 千克肥料和 P_2 千克農藥。設 S_1 和 S_2 分別為每公頃小麥和大麥的售價。如果用 x_1 和 x_2 分別表示種植小麥和大麥的面積，則通過選擇 x_1 和 x_2 的最佳值可以實現利潤最大化。這個問題可以表示為以下標準型的線性規劃問題：

max:	$S_1 \cdot x_1 + S_2 \cdot x_2$	(最大化收益，即小麥總銷售額加大麥總銷售額，收益是「目標函數」)
s.t.	$x_1 + x_2 \leq L$	(總面積限制)
	$F_1 \cdot x_1 + F_2 \cdot x_2 \leq F$	(肥料限制)
	$P_1 \cdot x_1 + P_2 \cdot x_2 \leq P$	(農藥限制)
	$x_1 \geq 0, x_2 \geq 0$	(種植面積不能為負)

Linear Programming

Approximation Ratio

- Solution space
  - Valid (須是合法解)
  - 例如搜尋的問題，solution space = {insertion, quick, bubble, merge, radix, …}
  - Solution space = {A, B, C, … , optimum, …}

- appox. ratio = (所提出的演算法/最佳的演算法)

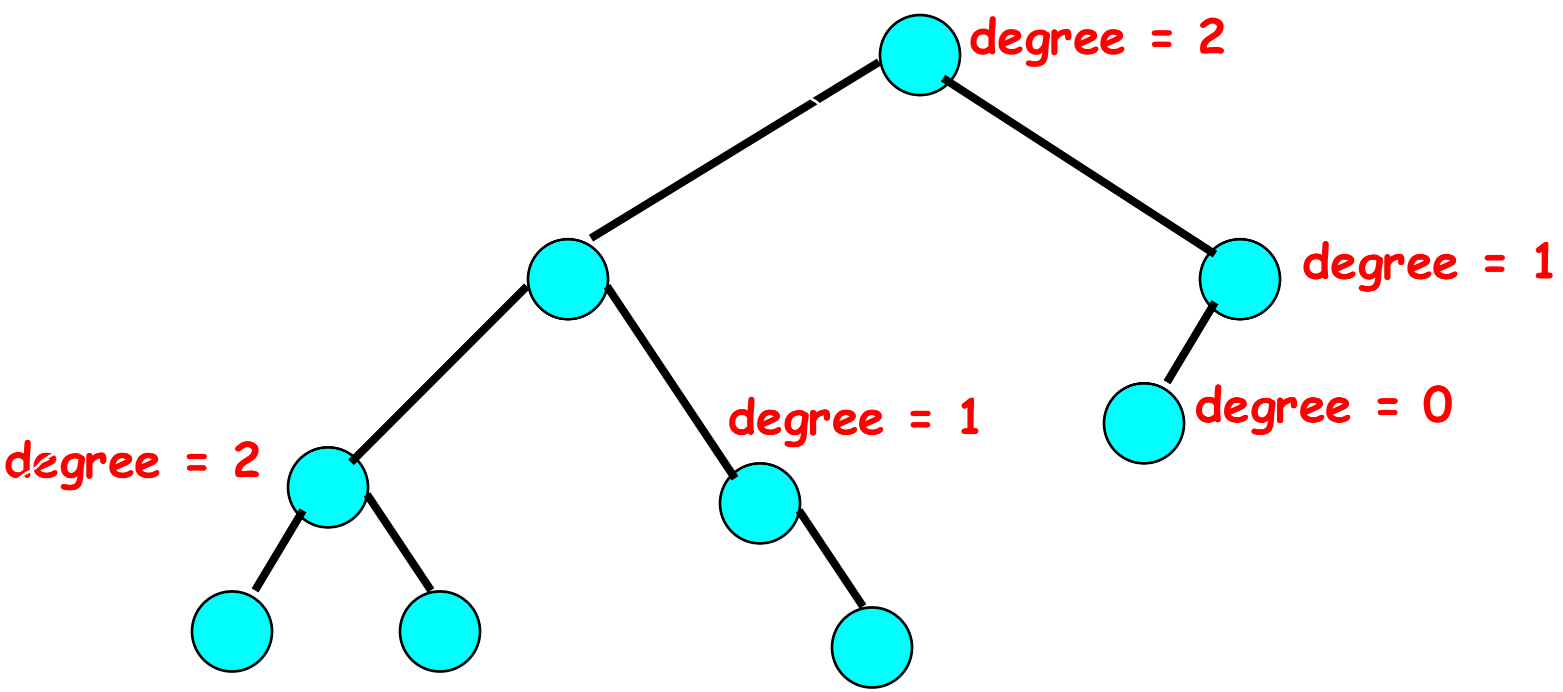
- 最佳解 “並非” 人類眼前最好的做法，也更不是一般平均水平的做法
- 最佳解往往不存在，人類無法在多項式時間 (polynomial time) 內能找出來
- Instead of optimum, we seek performance lower/upper bound

Trees and Binary Trees

Binary Trees

- Recursive definition:
  - 每個node有left sub-tree and/or right sub-tree
  - Left sub-tree (or right sub-tree) is a binary tree // 方向性!

- Root, leaf, and internal node (非root與非leaf以外的節點)
- Degree of any given node: number of sub-trees



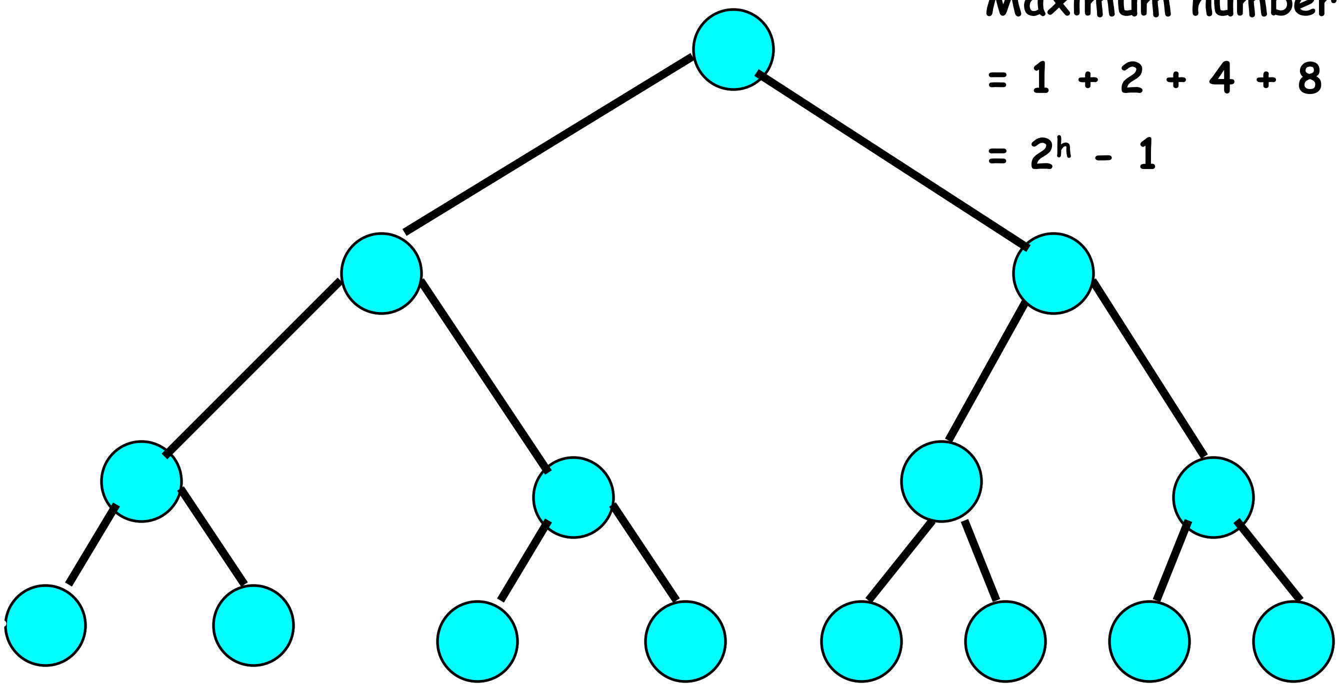
"Full" Binary Tree

- A full binary tree of a given height h has $2^h - 1$ nodes.

Maximum number of nodes

$= 1 + 2 + 4 + 8 + \dots + 2^{h-1}$

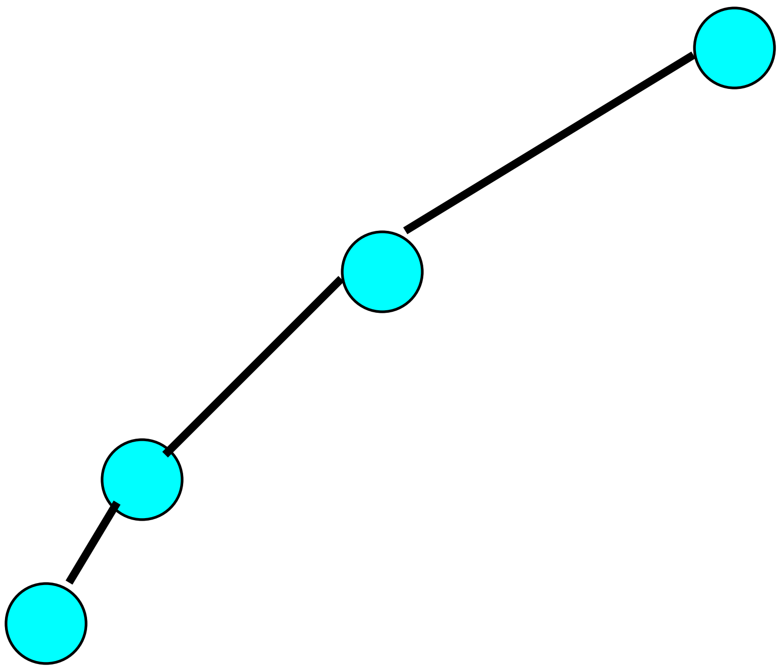
$= 2^h - 1$



Height 4 full binary tree.

Minimum Number of Nodes

- Minimum number of nodes in a binary tree whose height is h.
- At least one node at each of first h levels.



minimum number of nodes is h

Number of Nodes & Height

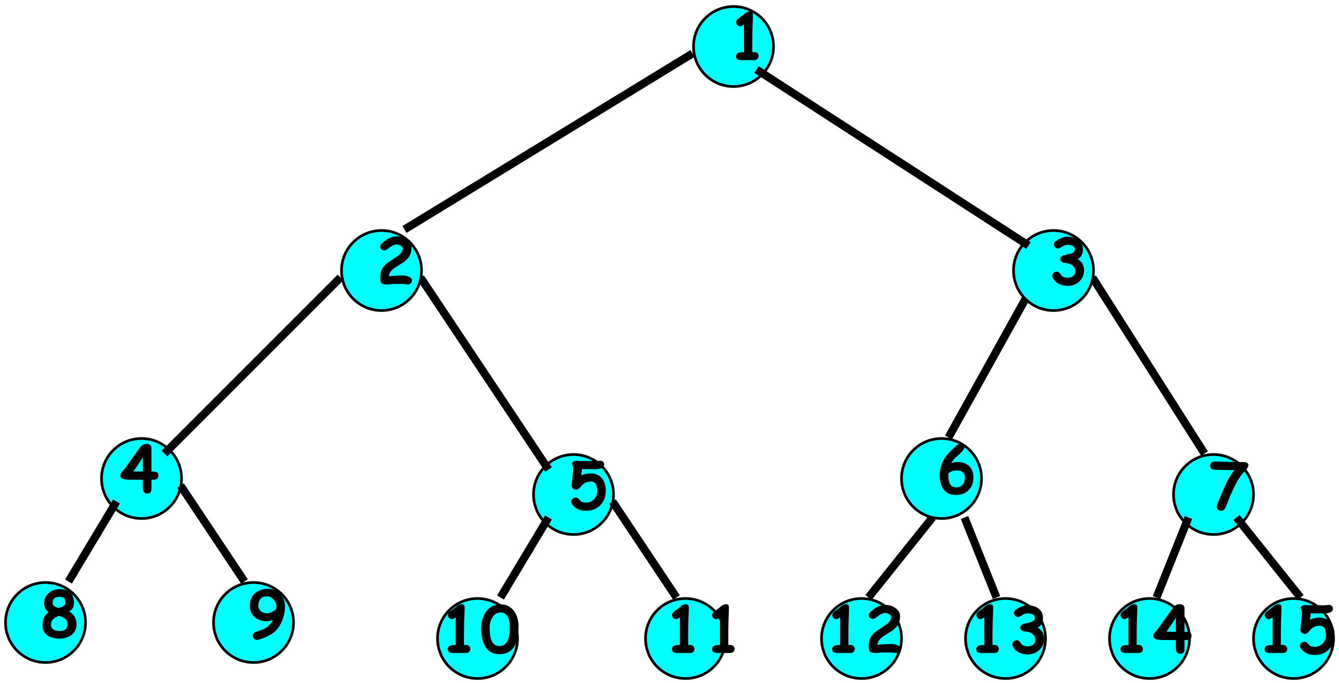
- Let n be the number of nodes in a binary tree whose height is h.
- $h <= n <= 2^h - 1$
- $\log_2(n+1) <= h$

h的上界 (upper bound)? 下界 (lower bound)?

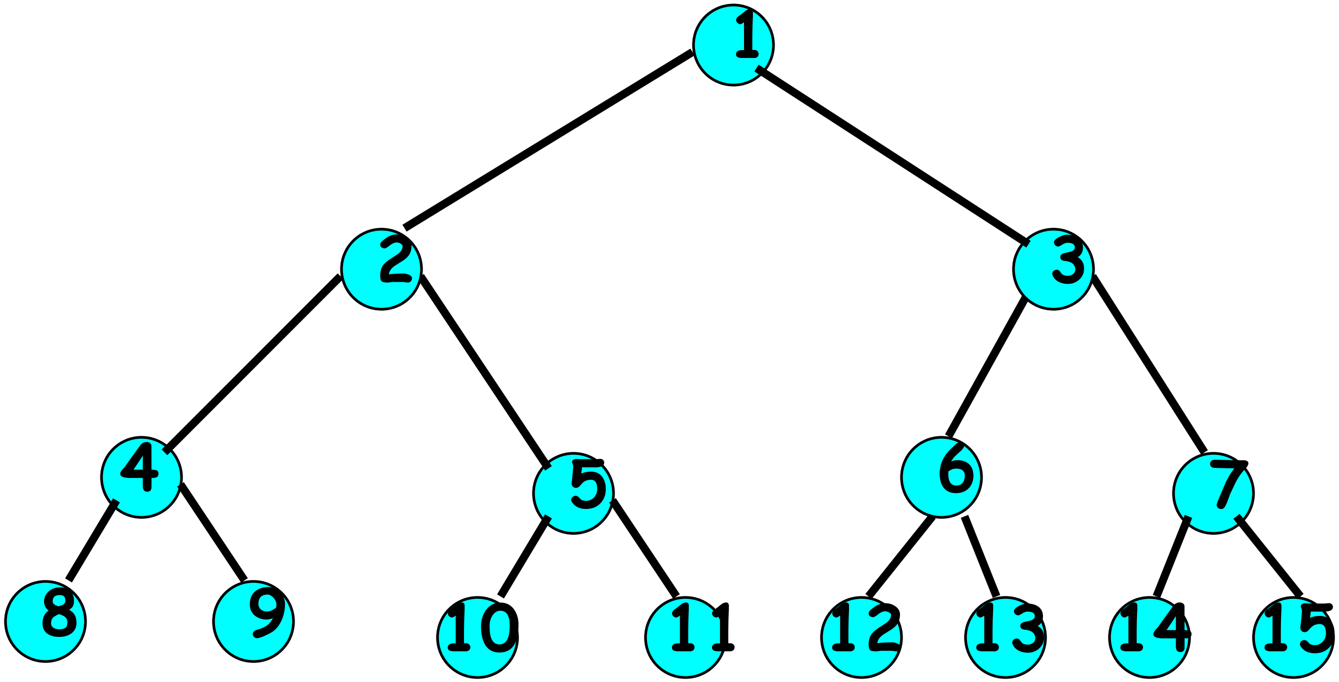
n $\log_2(n+1)$

Numbering Nodes in a Full Binary Tree

- Number the nodes 1 through $2^h - 1$.
- Number by levels from top to bottom.
- Within a level number from left to right.



Node Number Properties

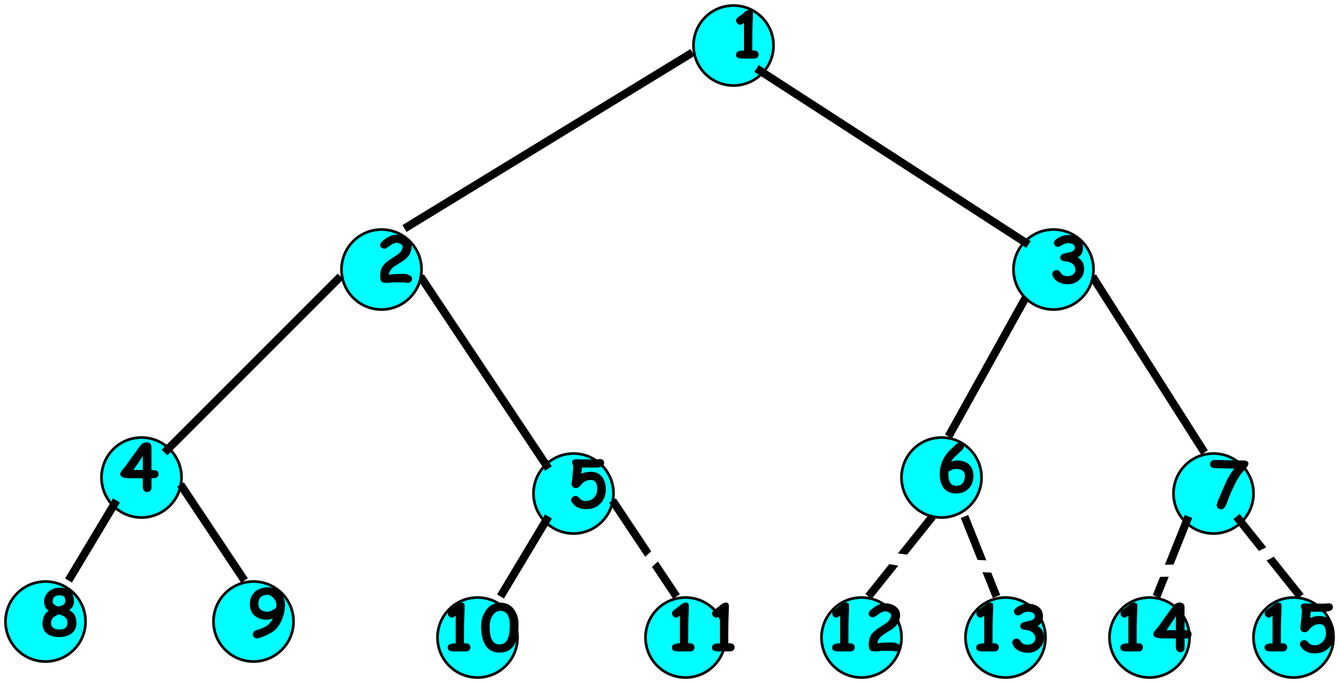


- Parent of node i is node floor(i / 2)
  - Node 1 is the root and has no parent
- Left (or Right) child of node i is node 2i (or 2i+1)

"Complete" Binary Tree with n Nodes

- Number the nodes as described earlier.
- The binary tree defined by the nodes numbered 1 through n is the unique n node complete binary tree.

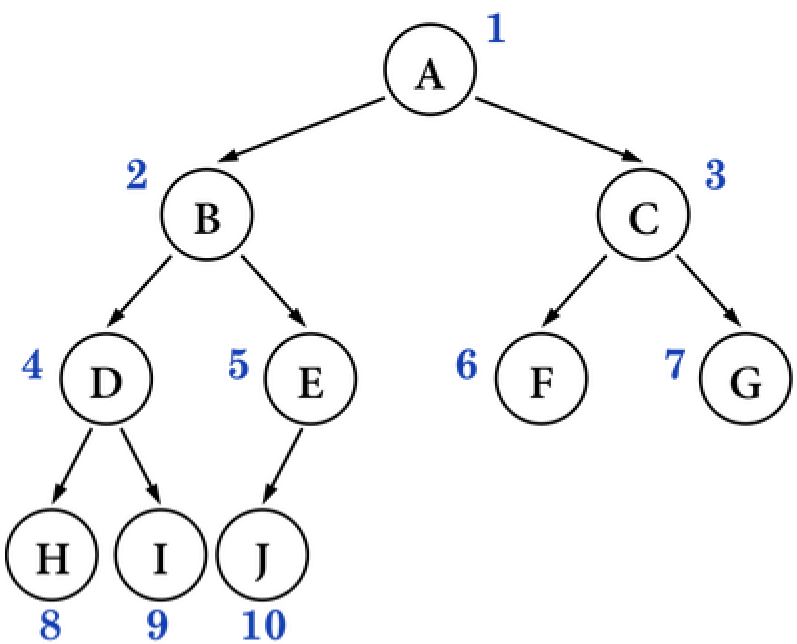
Example

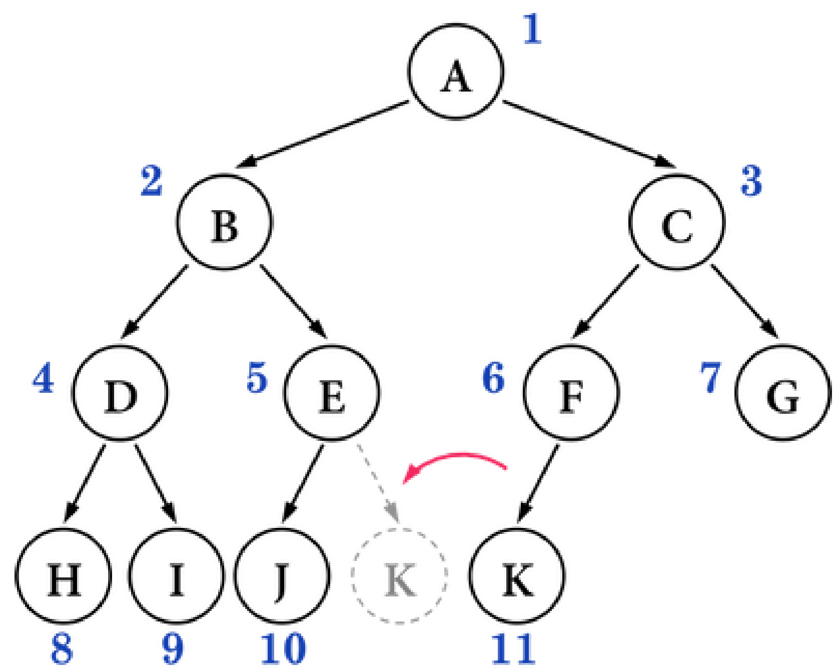


Complete binary tree with 10 nodes.

Note: Full binary tree是complete binary tree的一個特例

Yes (Left) and No (Right)





Binary “Search” Trees

Binary Search Trees

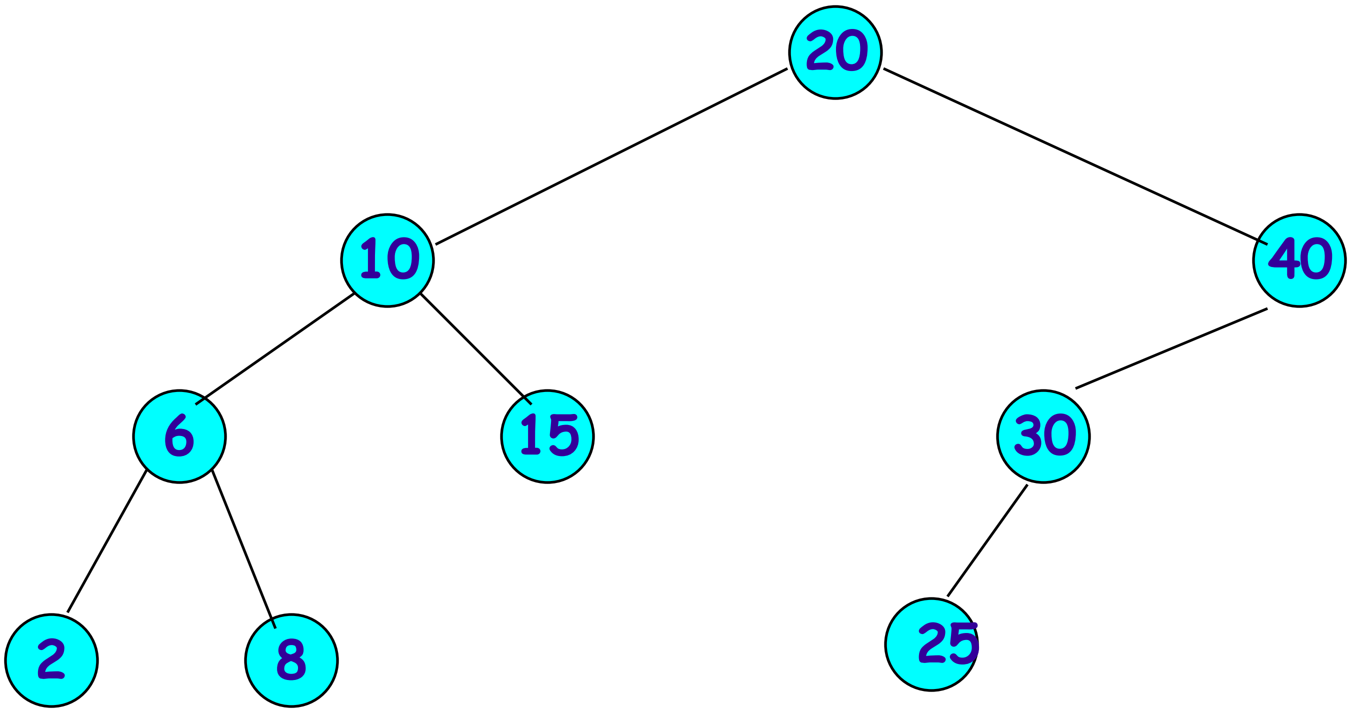
- Operations:
  - Search(key)
  - Insert(key, value)
  - Delete(key)
  - …

Definition of Binary Search Tree

- A binary tree.
- Each node has a (key, value) pair.
- For every node x, all keys in the left subtree of x are smaller than that in x.
- For every node x, all keys in the right subtree of x are greater than that in x.

我們暫時僅考慮所有資料的keys皆不同的情形

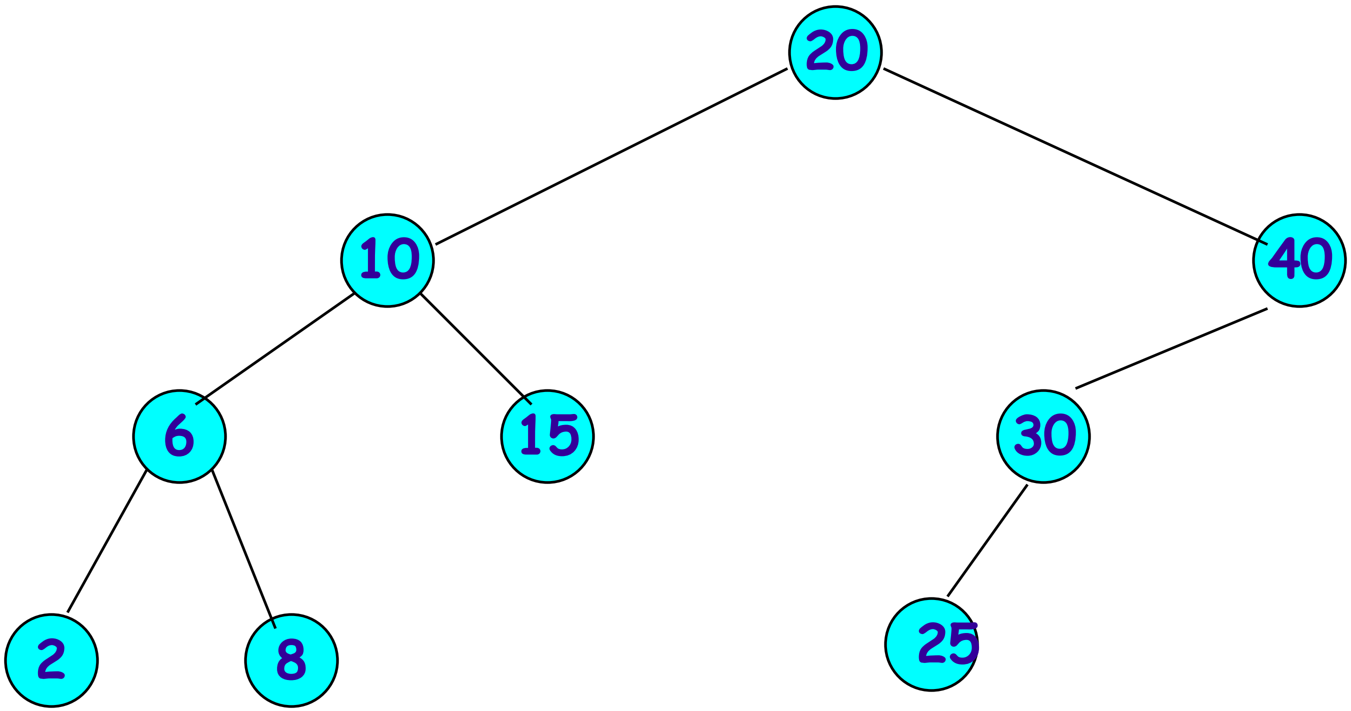
Example Binary Search Tree



Only "**keys**" are shown.

(節點內還會有其他的資料欄位)

The Operation Search()

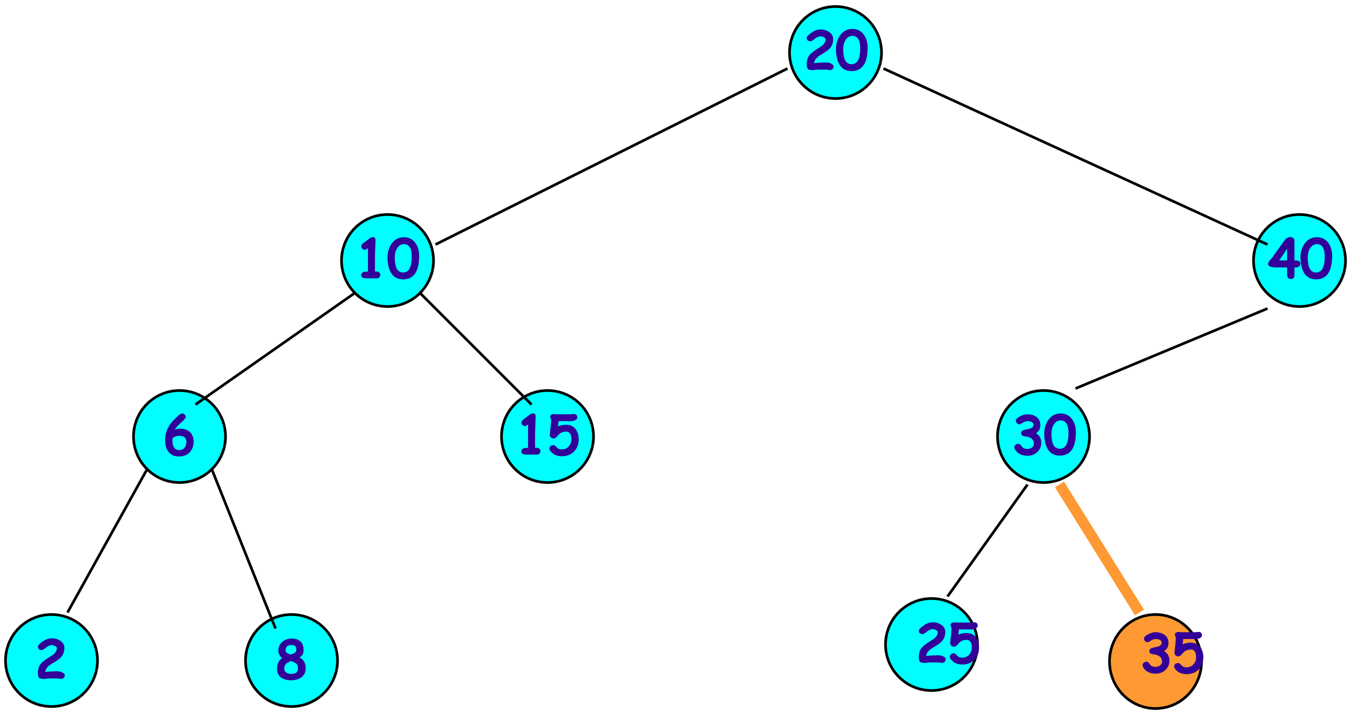


Complexity is O(height) = O(n), where n is number of nodes/elements. (樹可能歪歪斜斜的, skew)

Insert與Delete的初體驗

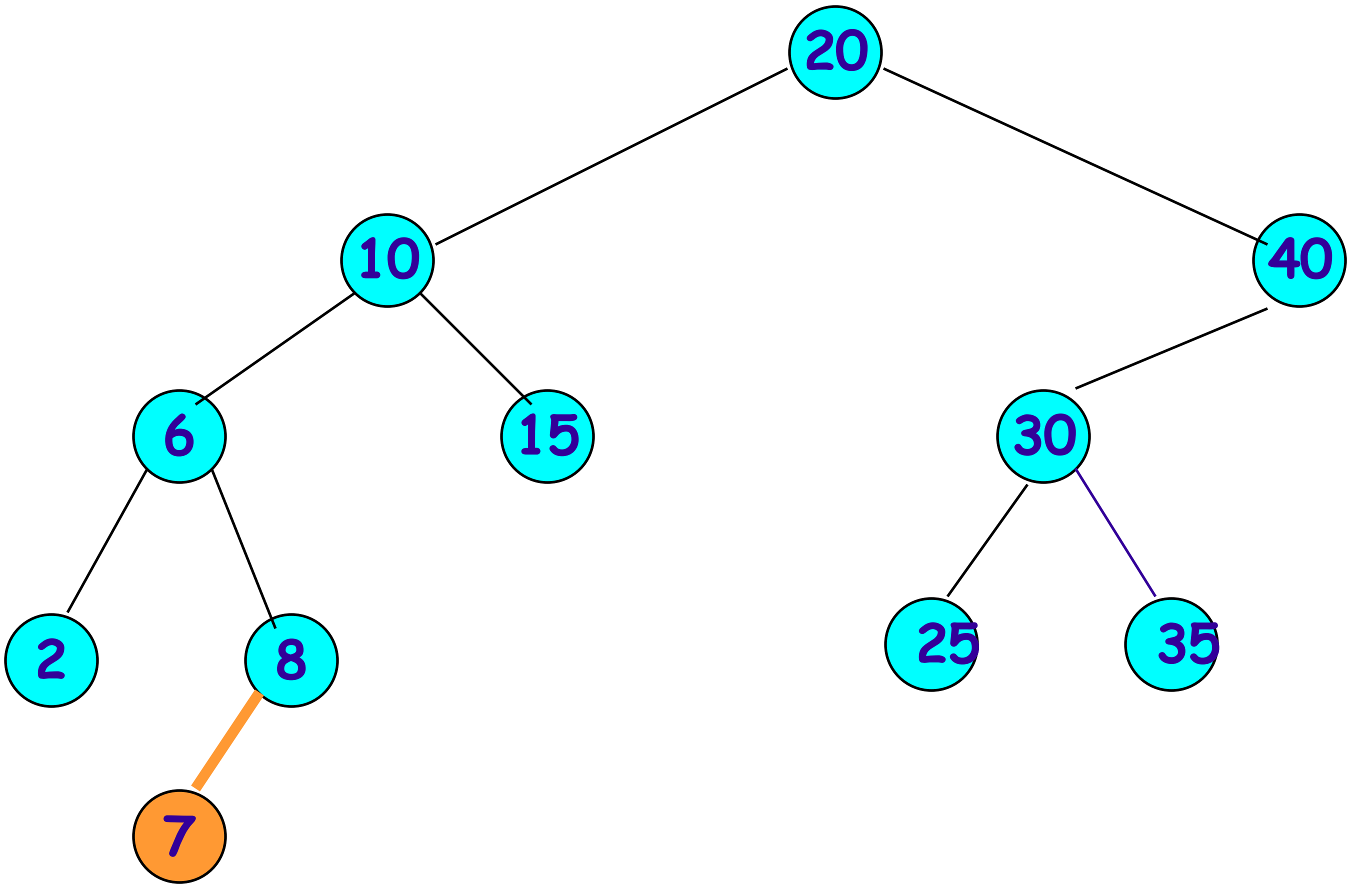
- 僅是讓同學了解insert與delete之於binary search tree上的操作之初體驗
  - Insert前是binary search tree，insert後還是binary search tree (binary tree的定義必須被維持)
  - Delete也是如此
- 往後，我們會面對“結構”上較具要求的binary search tree

The Operation Insert()



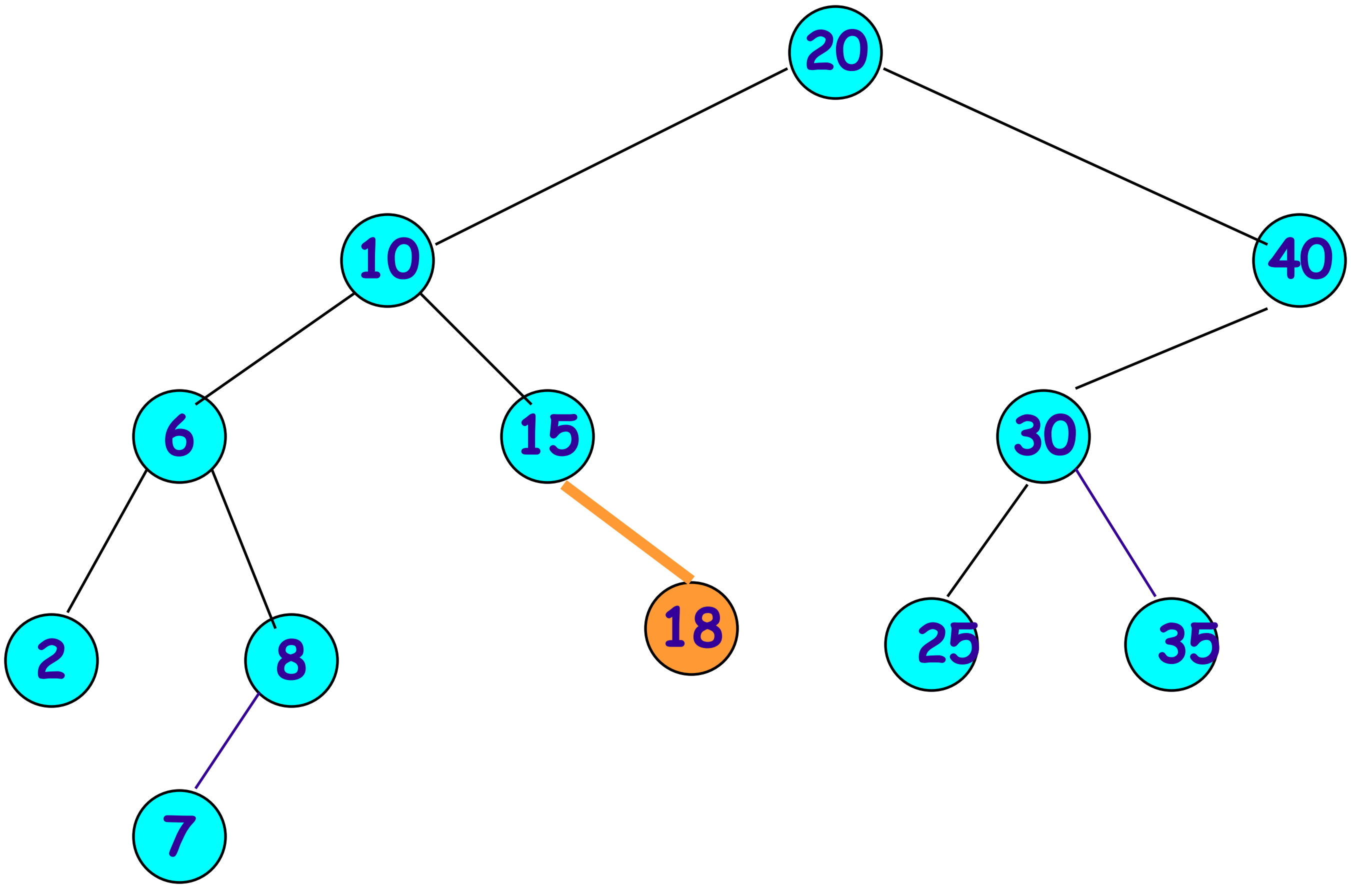
Insert a pair whose key is 35.

The Operation Insert()



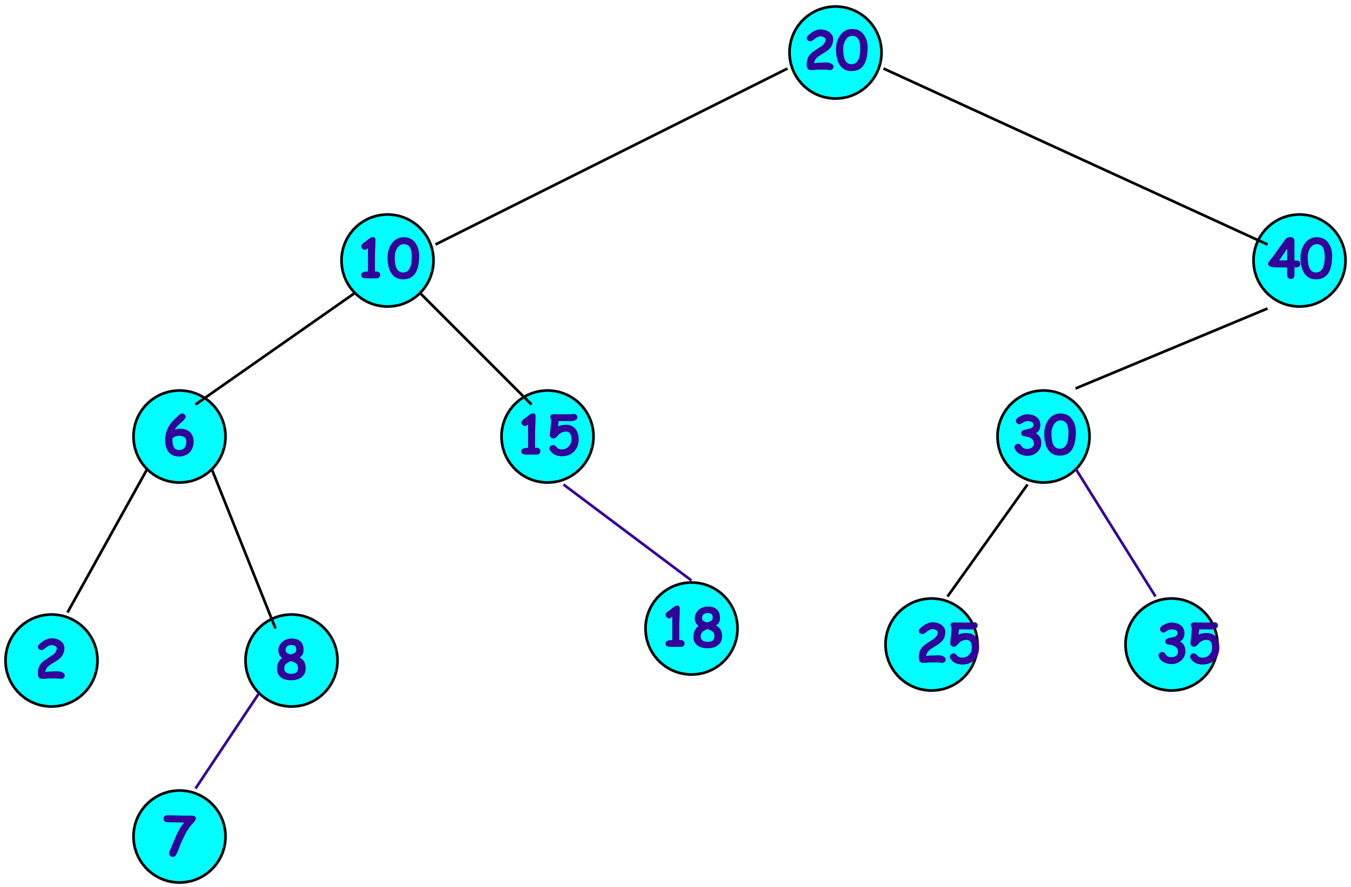
Insert a pair whose key is 7.

The Operation Insert()



Insert a pair whose key is 18.

The Operation Insert()

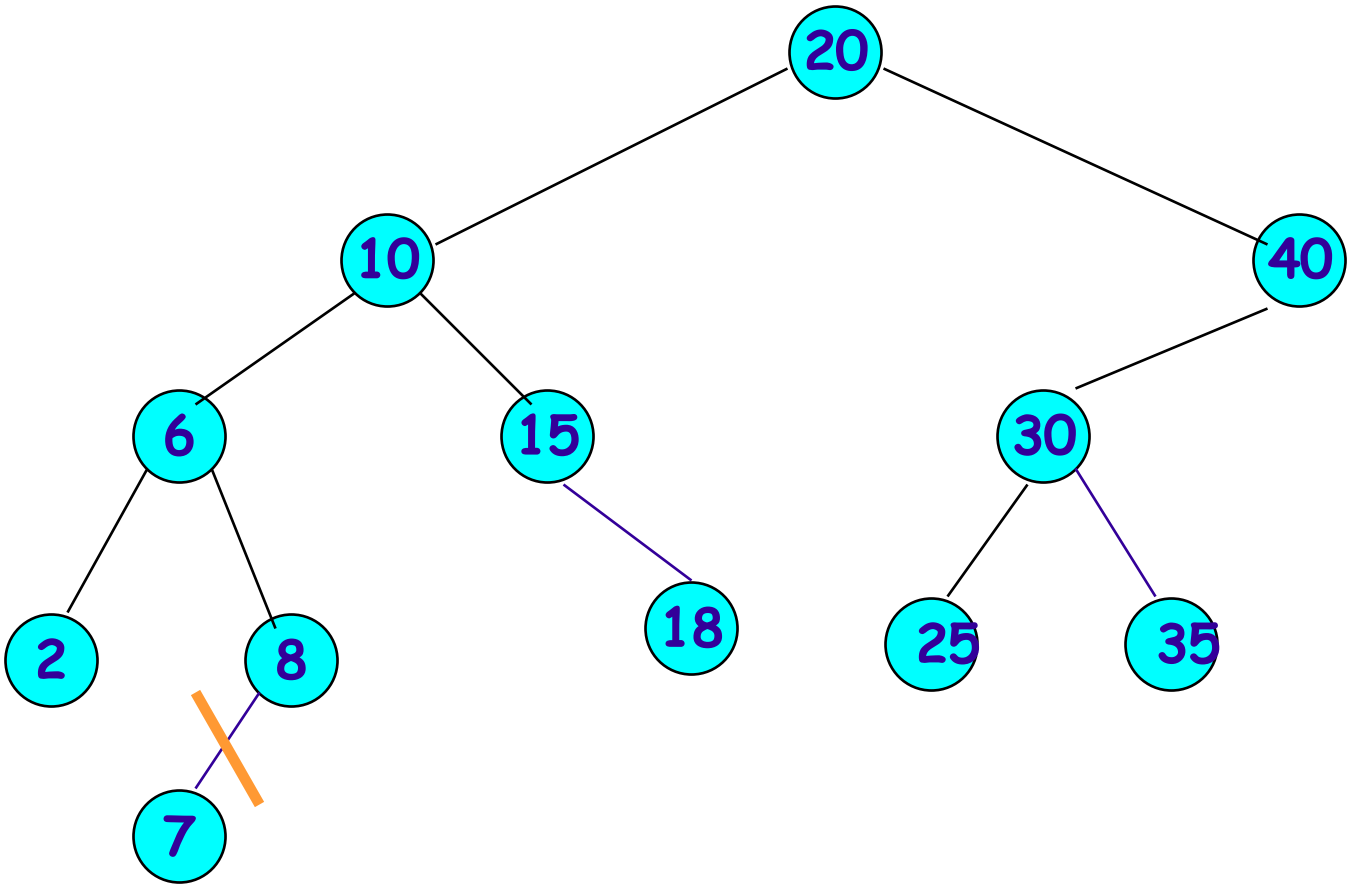


Complexity of Insert() is O(height).

The Operation Delete()

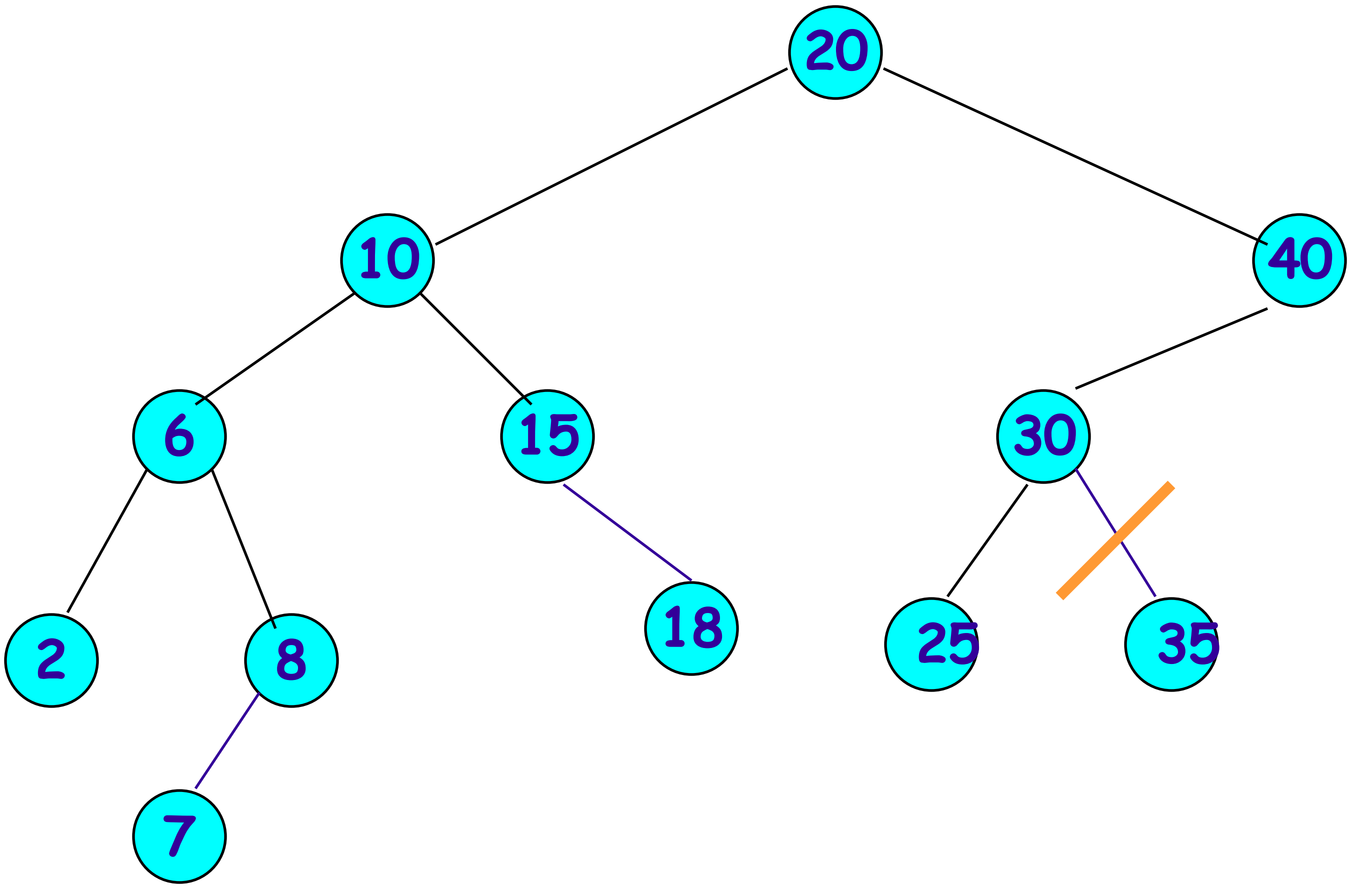
- Four cases:
  - No element with delete key.
  - Element is in a leaf.
  - Element is in a degree 1 node.
  - Element is in a degree 2 node.

Delete from a Leaf



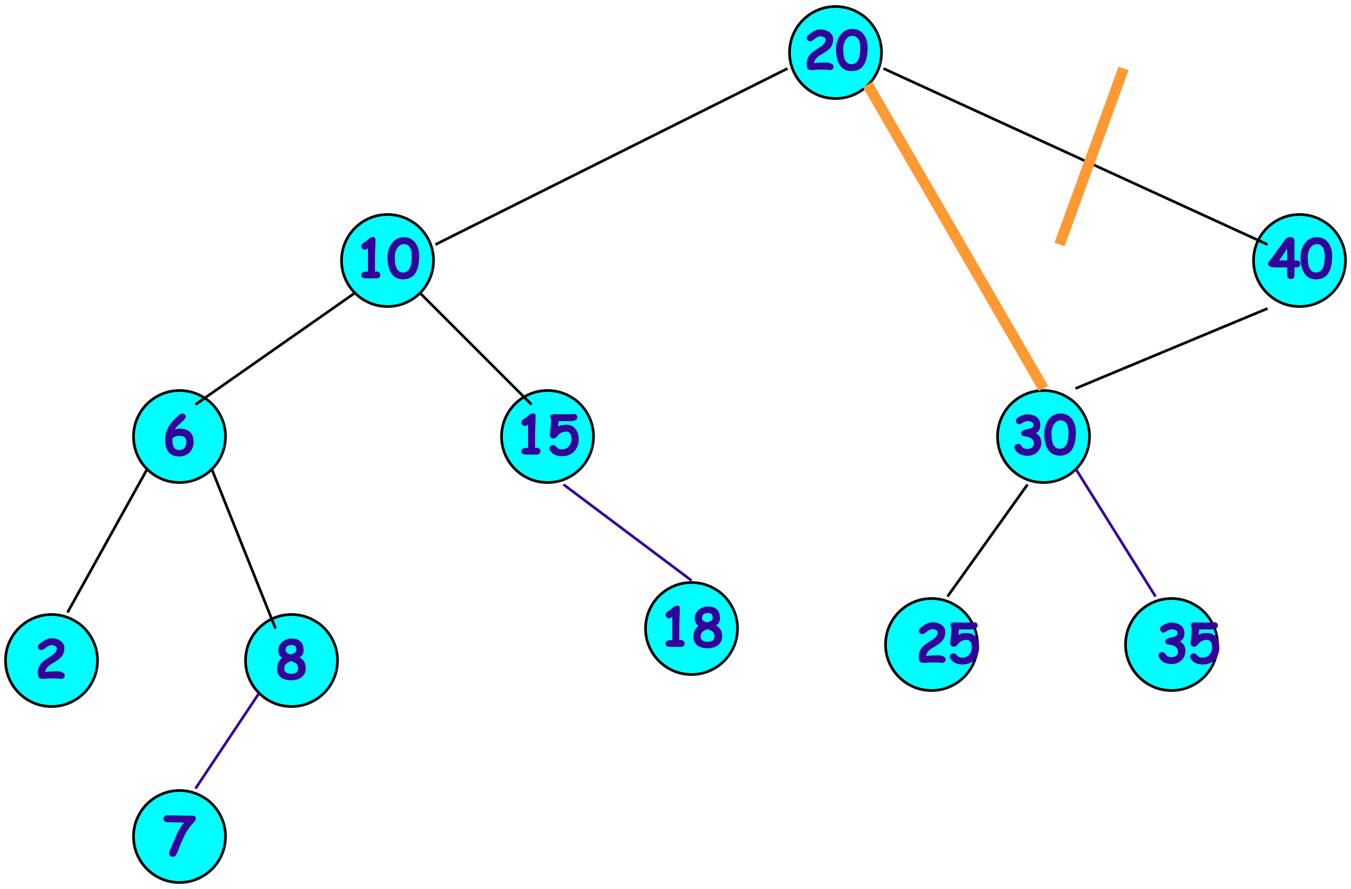
Delete a leaf element. key = 7

Delete from a Leaf (contd.)



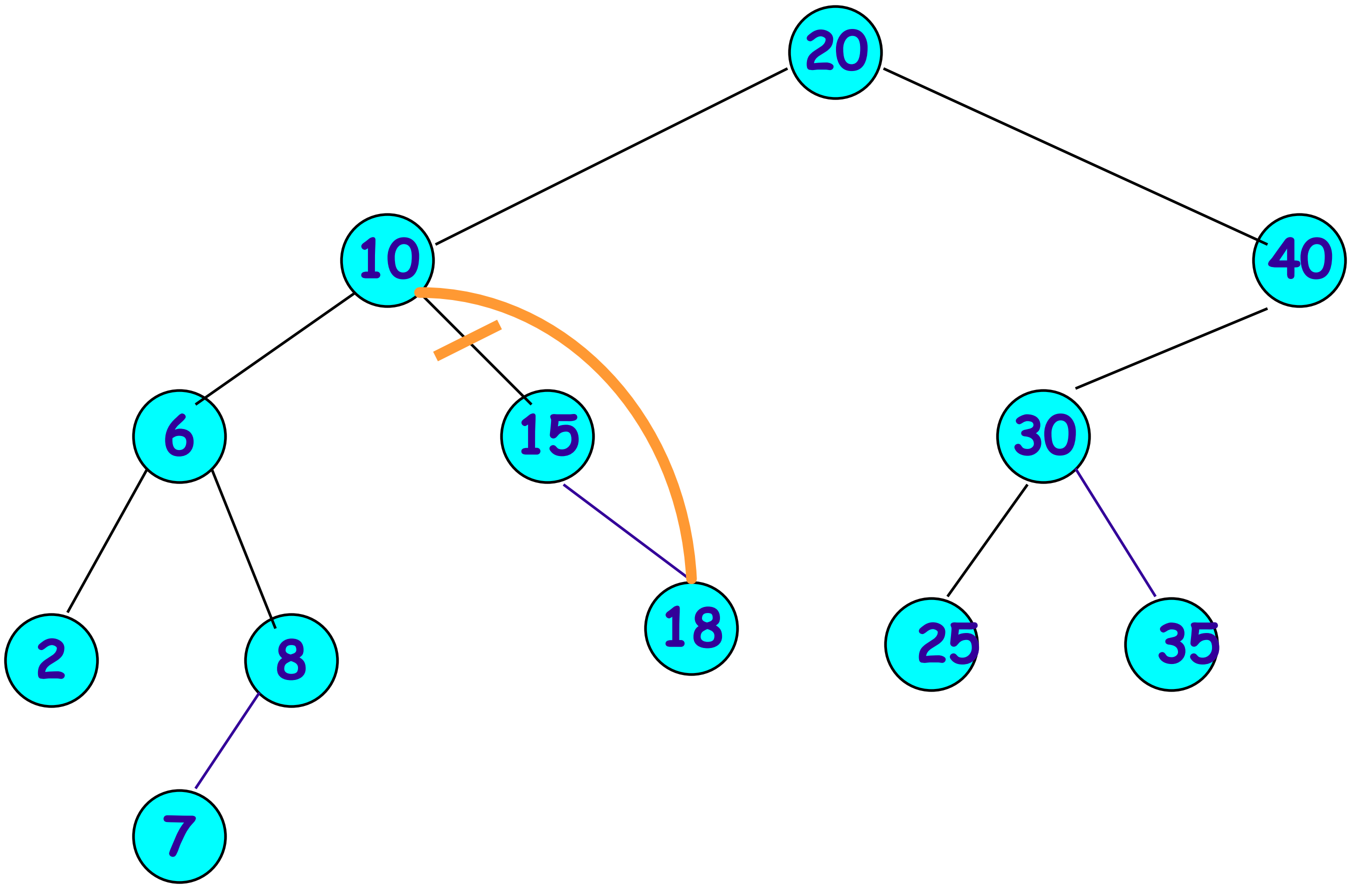
Delete a leaf element. key = 35

Delete from a Degree 1 Node



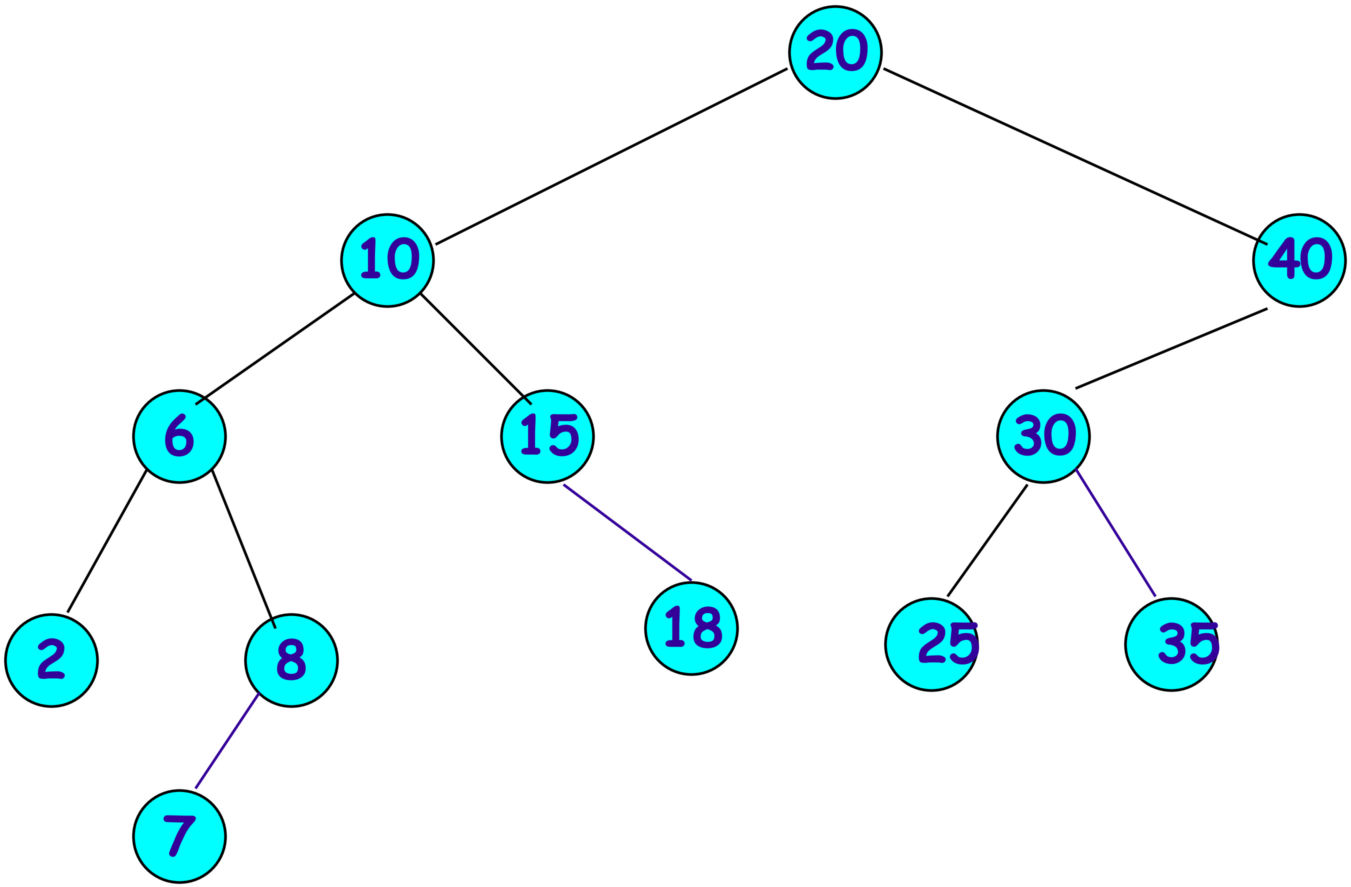
Delete from a degree 1 node. key = 40

Delete from a Degree 1 Node (contd.)



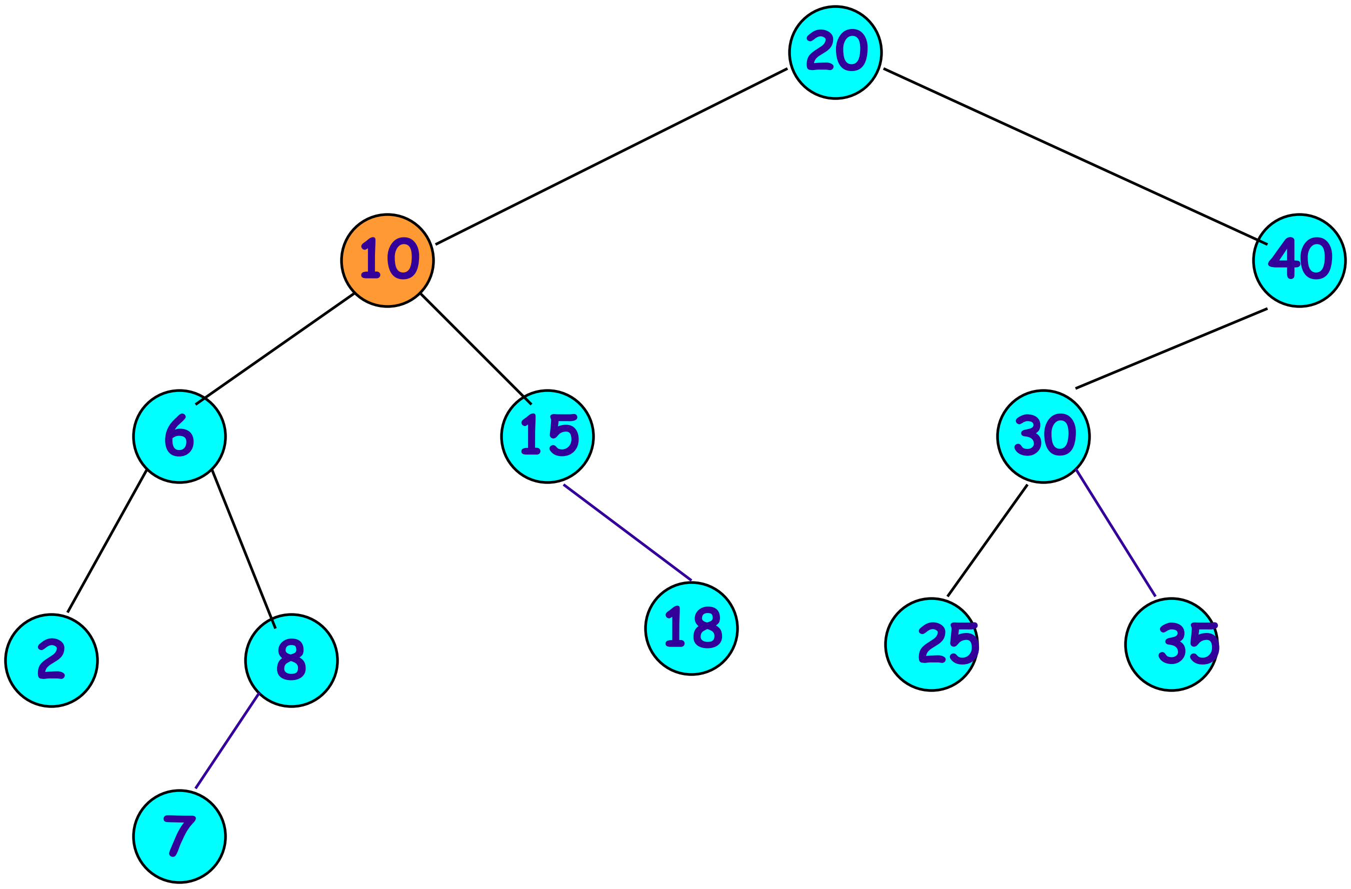
Delete from a degree 1 node. key = 15

Delete from a Degree 2 Node



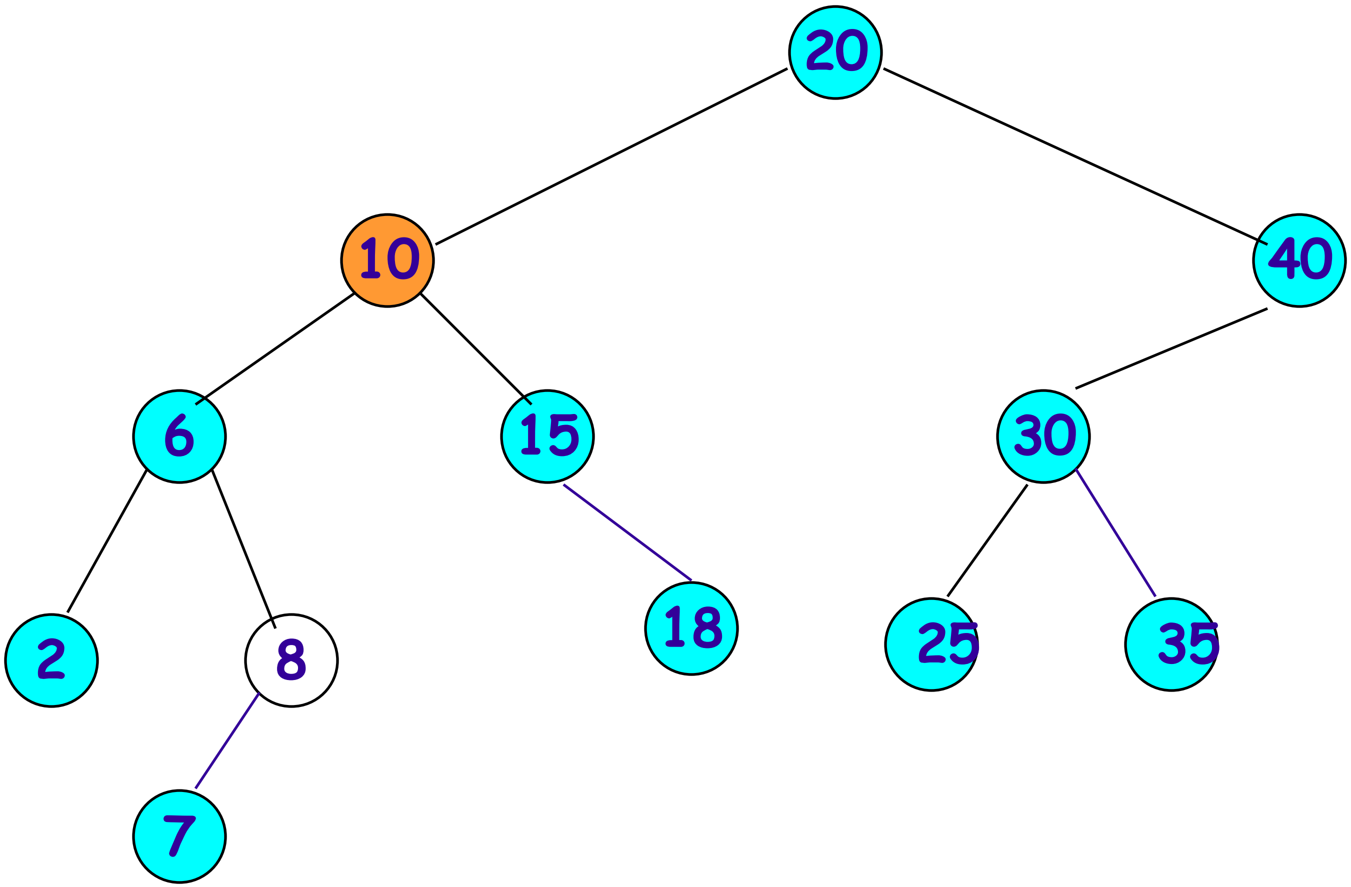
Delete from a degree 2 node. key = 10

Delete from a Degree 2 Node



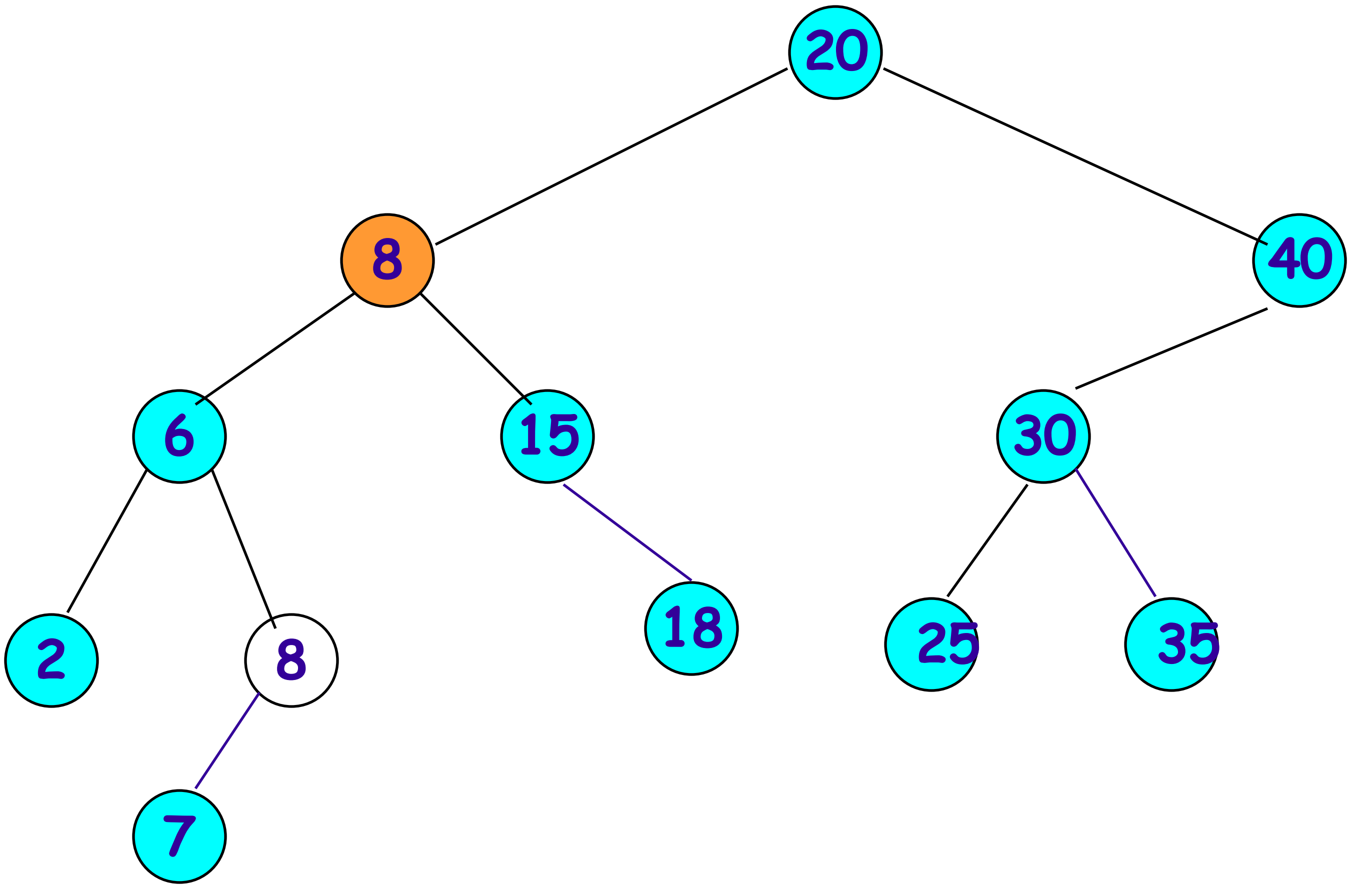
Replace with largest key in left subtree (or smallest in right subtree).

Delete from a Degree 2 Node
(我們說明一種可能的做法，做法非唯一)



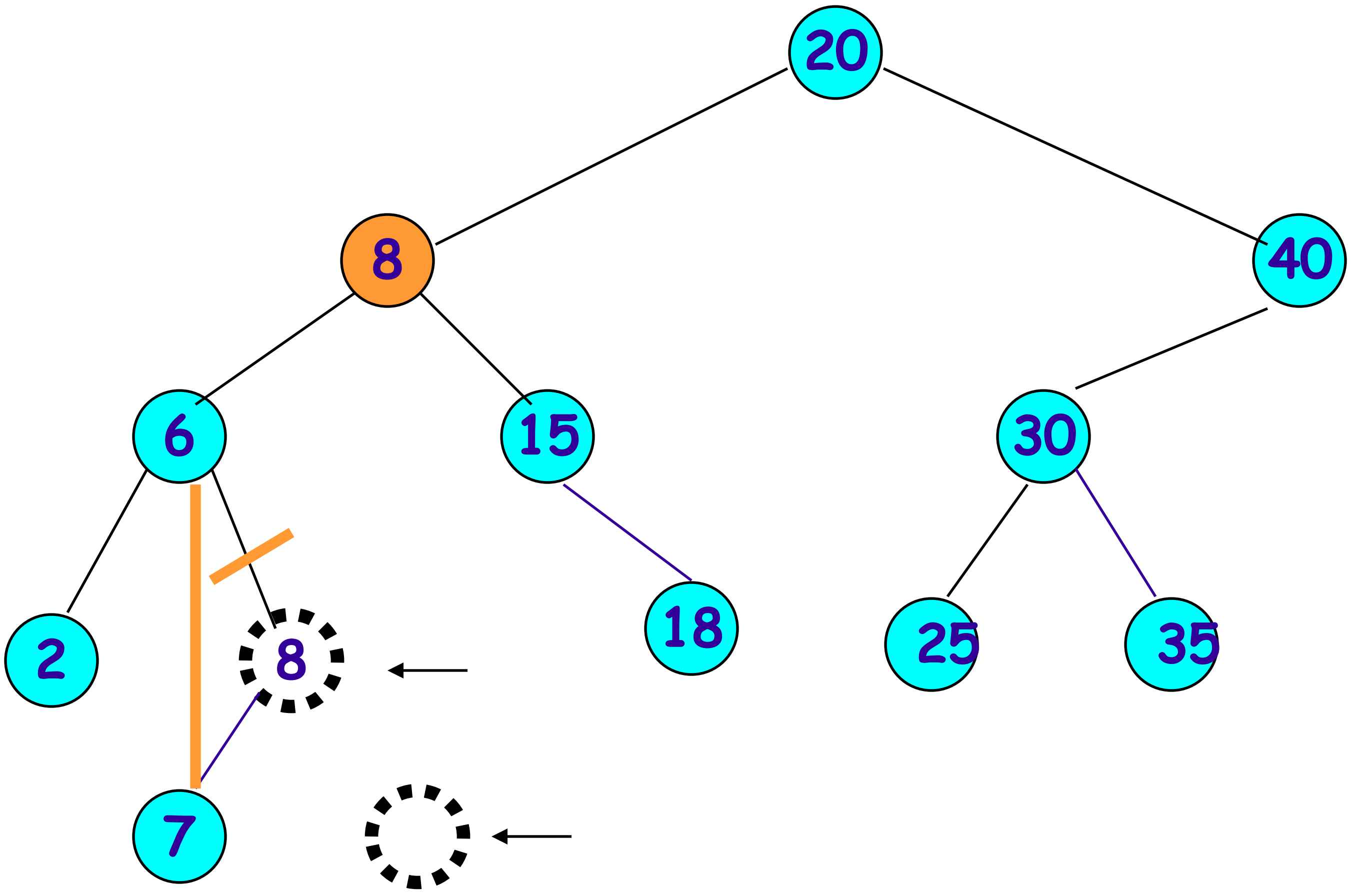
Replace with largest key in left subtree (or smallest in right subtree).

Delete from a Degree 2 Node



Replace with largest key in left subtree (or smallest in right subtree).

Delete from a Degree 2 Node

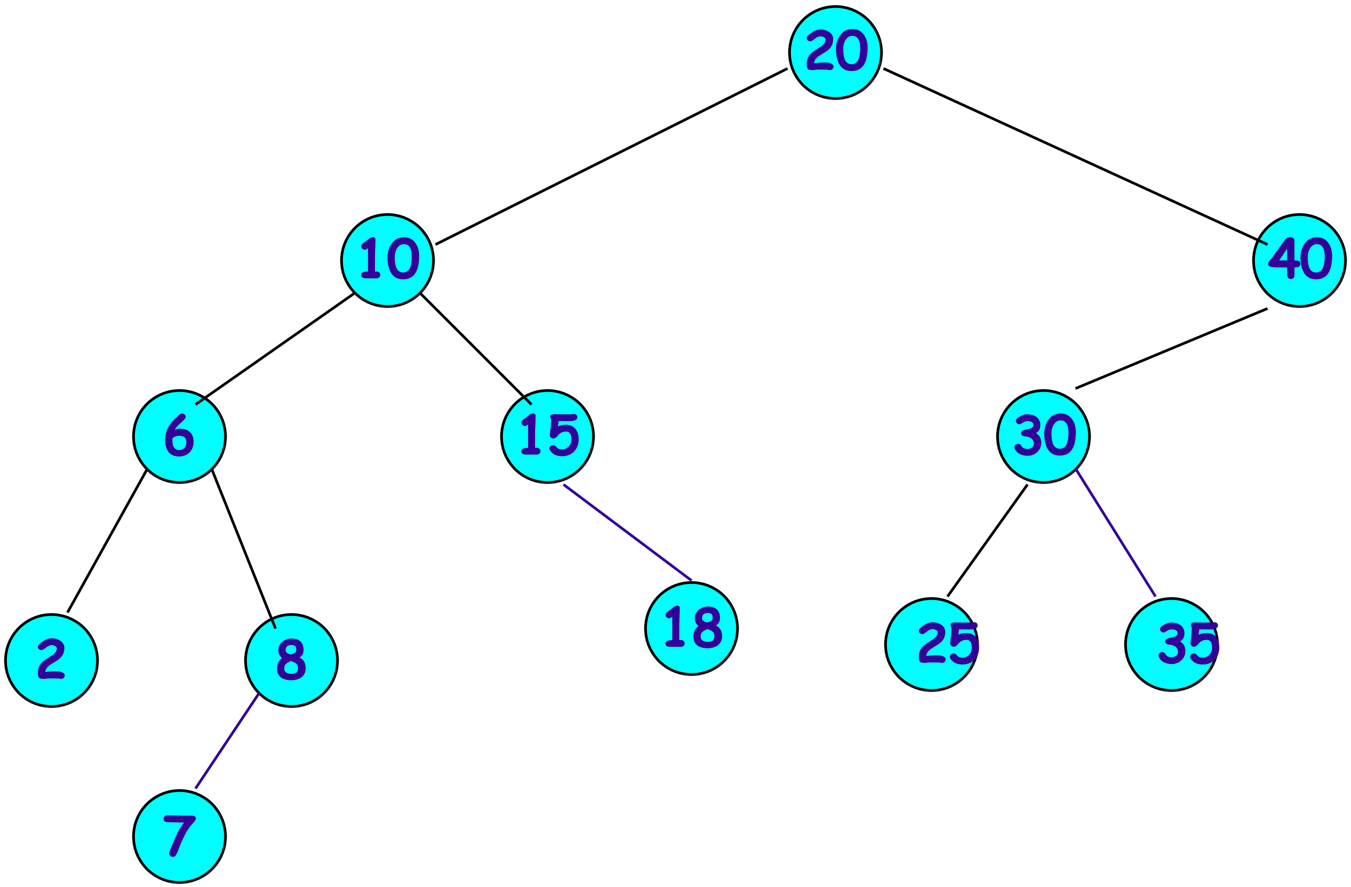


Largest key must be in a leaf or degree 1 node. Why?

善用已經存在的操作...

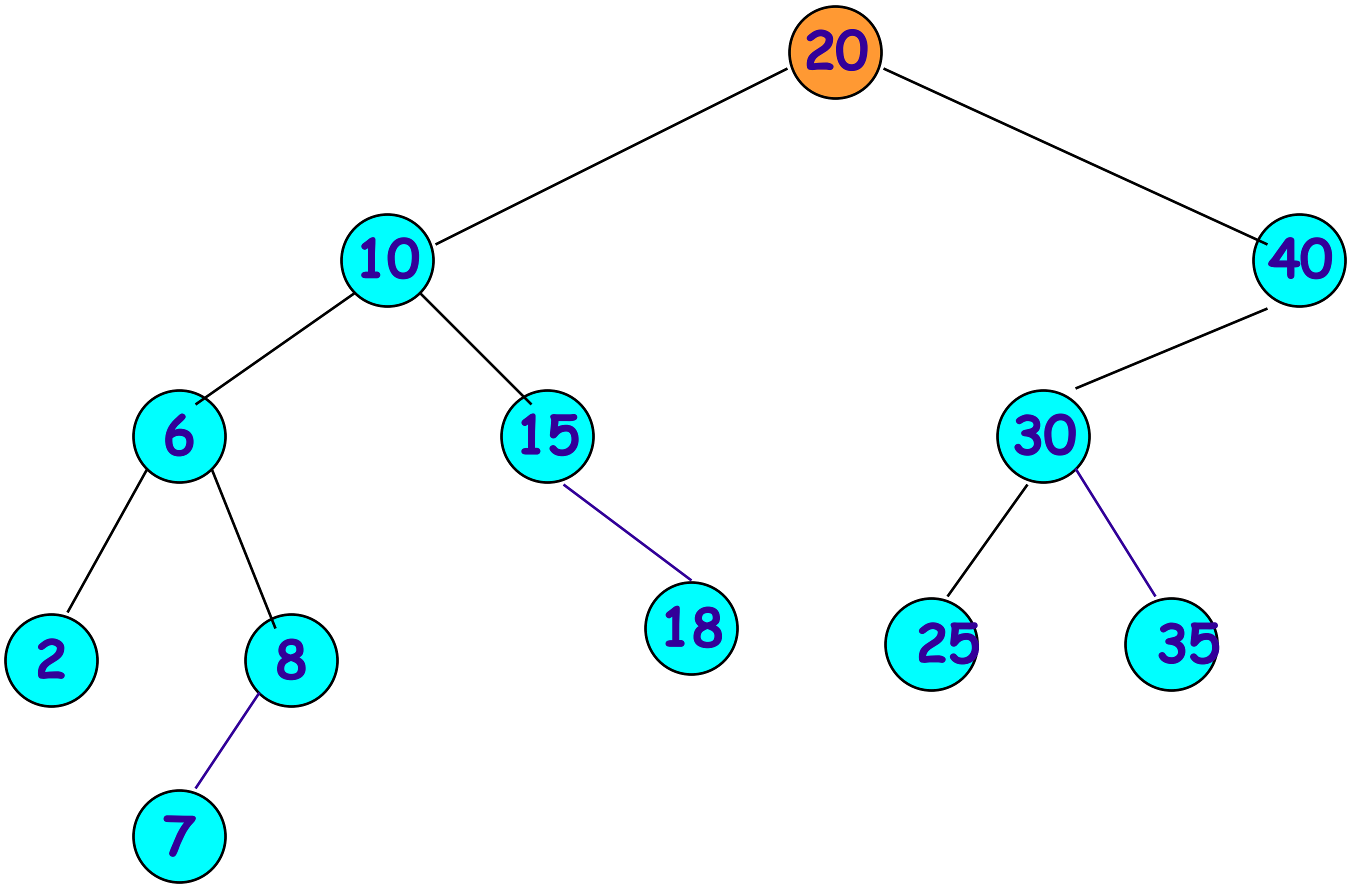
- 轉換成delete
  - degree-1 node, or
  - leaf

Another Delete from a Degree 2 Node



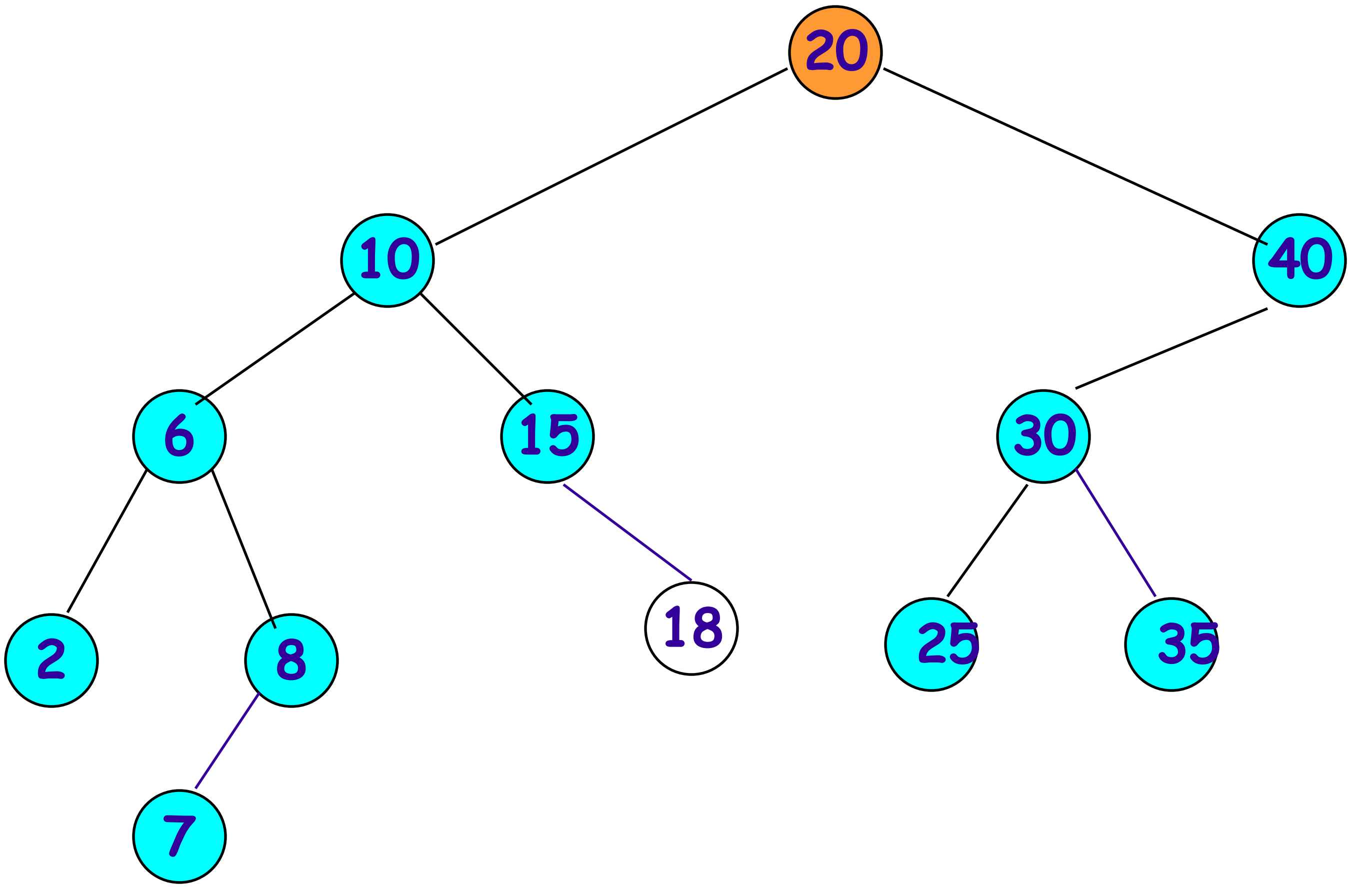
Delete from a degree 2 node. key = 20

Delete from a Degree 2 Node



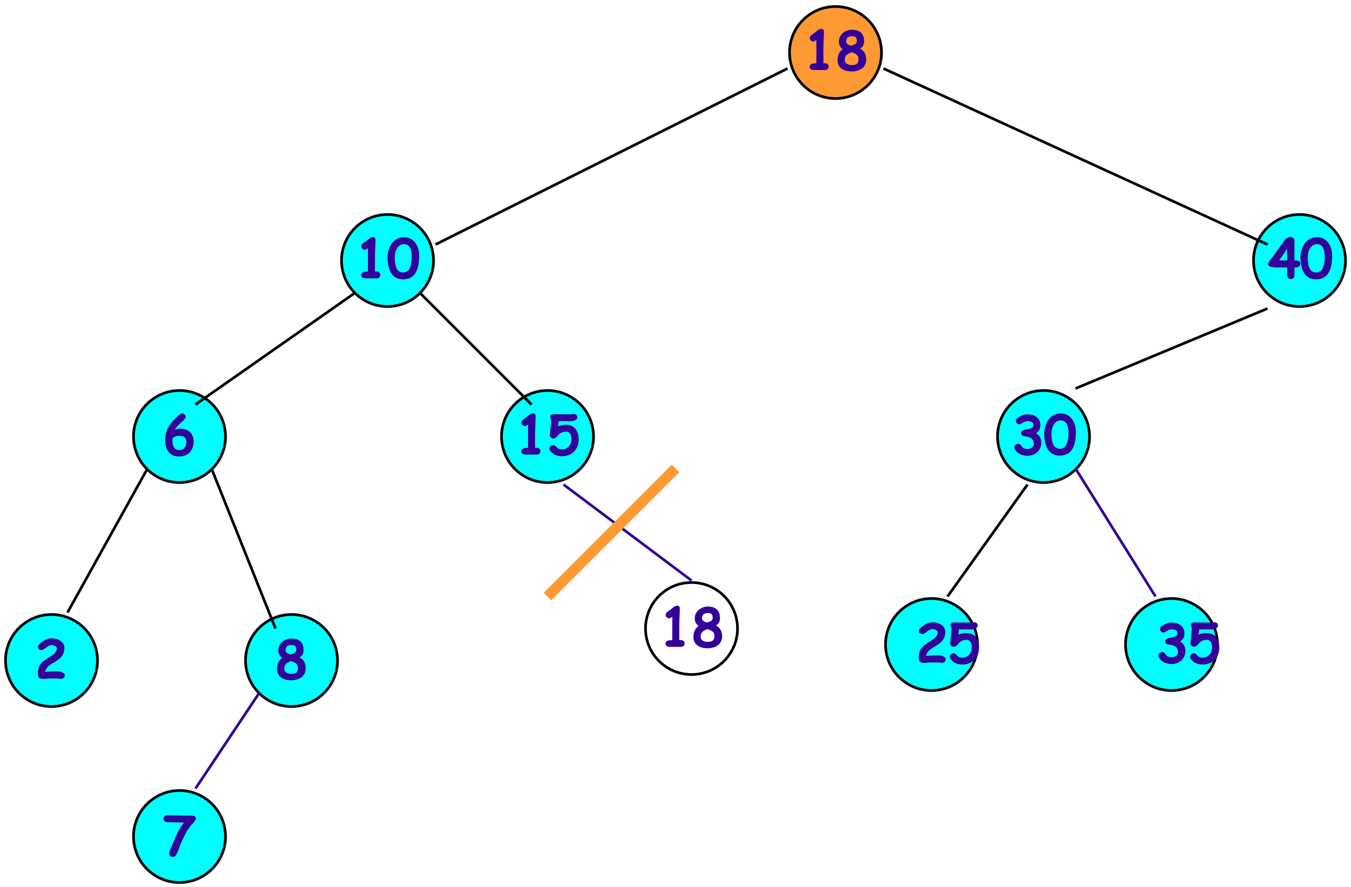
Replace with largest in left subtree.

Delete from a Degree 2 Node



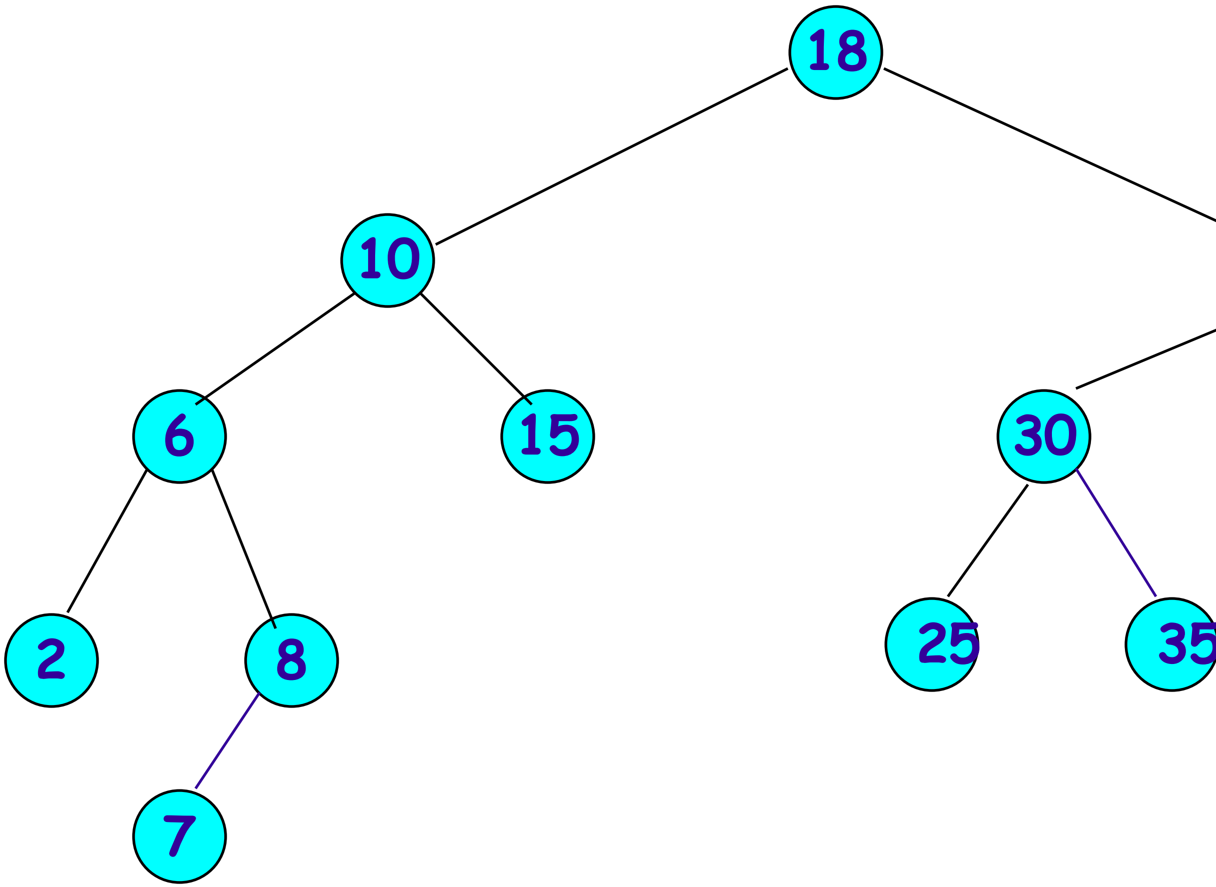
Replace with largest in left subtree.

Delete from a Degree 2 Node



Replace with largest in left subtree.

Delete from a Degree 2 Node



問: 有可能replacement (藍色節點) 是一個degree-2的node?
不可能!

Why?
(見前述)

Complexity is O(height).

Indexing

- Databases最重要、最核心的功能之一
  - 基於some search trees

Graphs

Graphs

- G = (V,E)
- V is the vertex set.
- Vertices are also called nodes and points.
- E is the edge set.
- Each edge connects two different vertices.
- Edges are also called arcs and lines.
- Directed edge has an orientation (u,v).

$$u \longrightarrow v$$

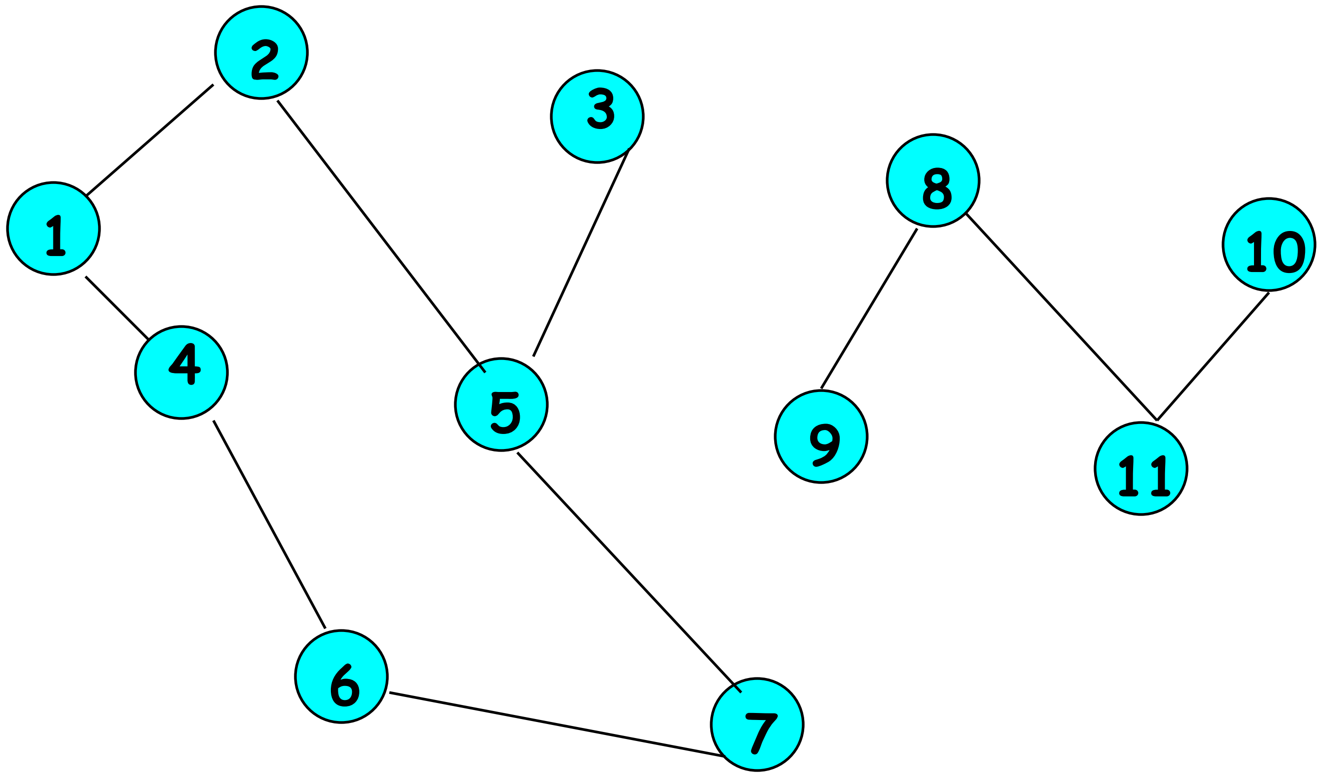
Undirected and Directed Graphs

- Undirected edge has no orientation (u,v).

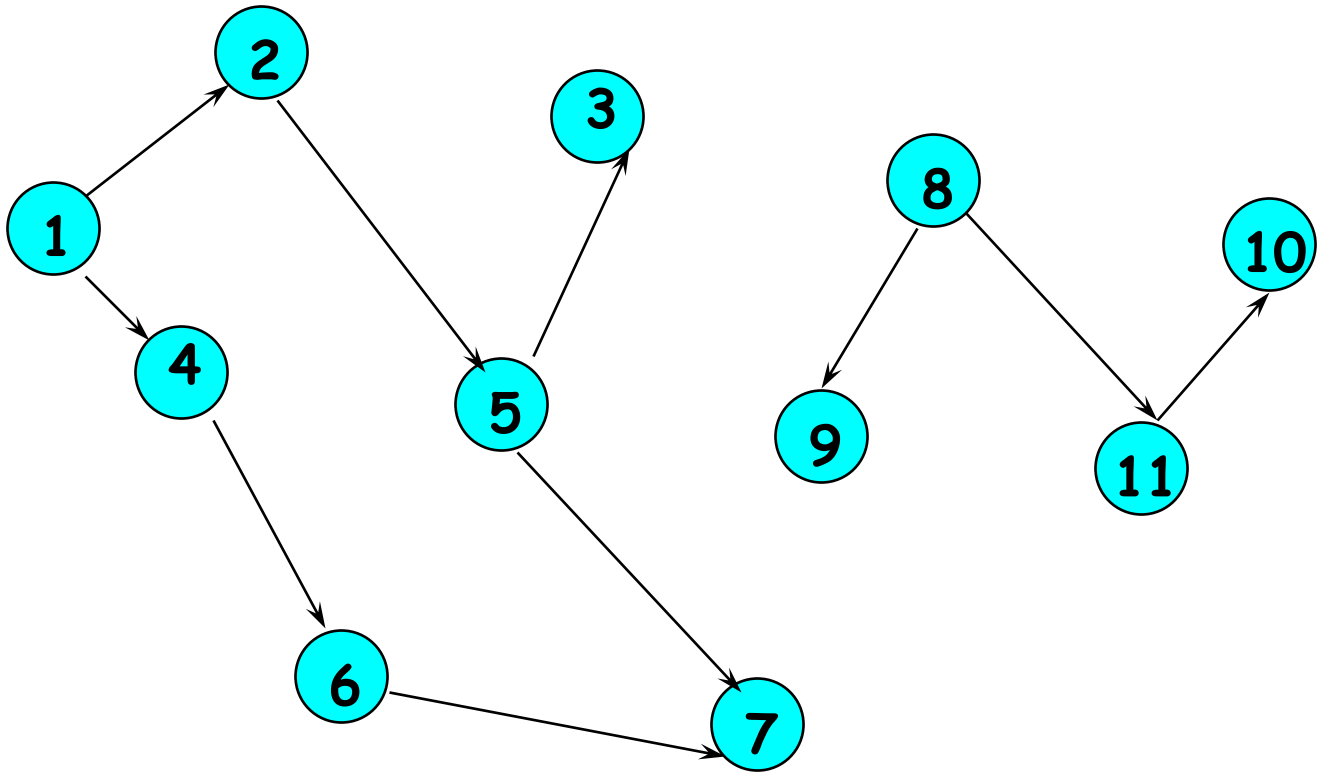
u ——— v

- Undirected graph => no oriented edge.
- Directed graph => every edge has an orientation.

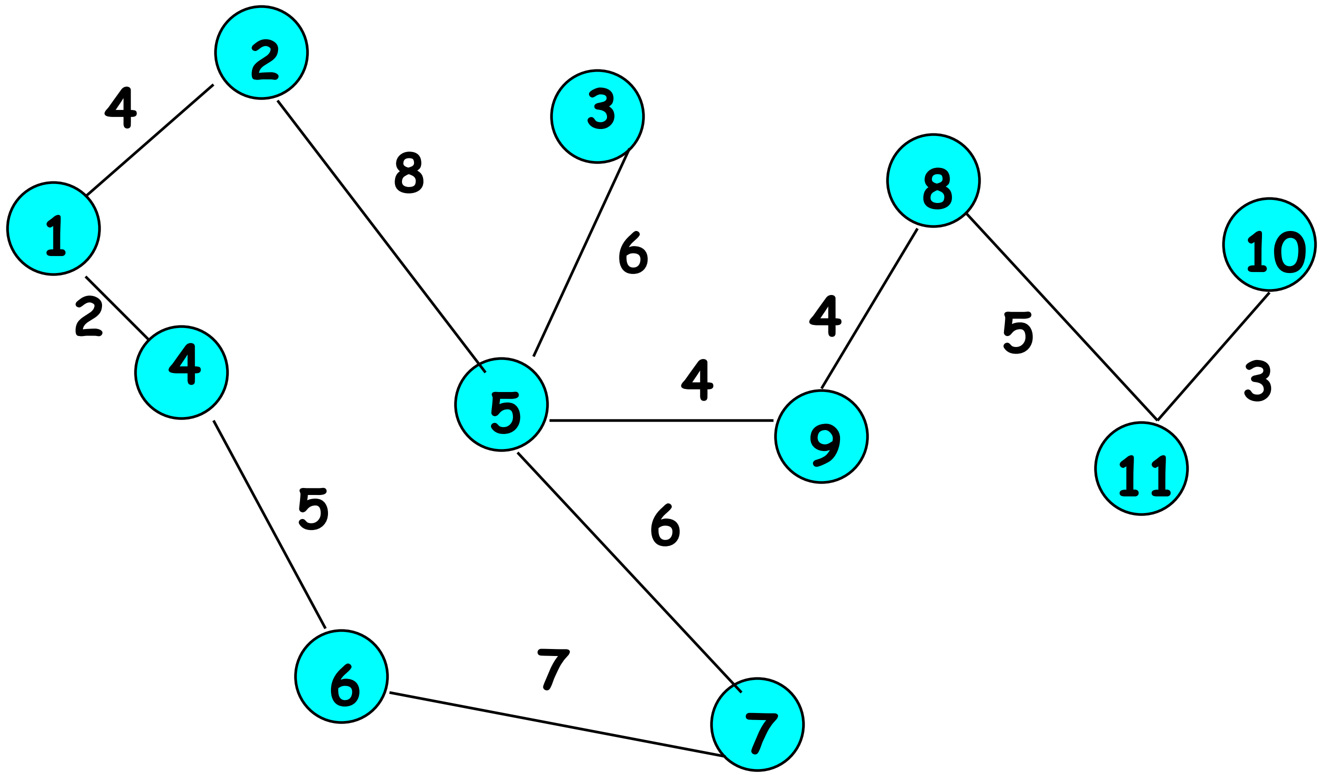
Undirected Graph



Directed Graph (Digraph)



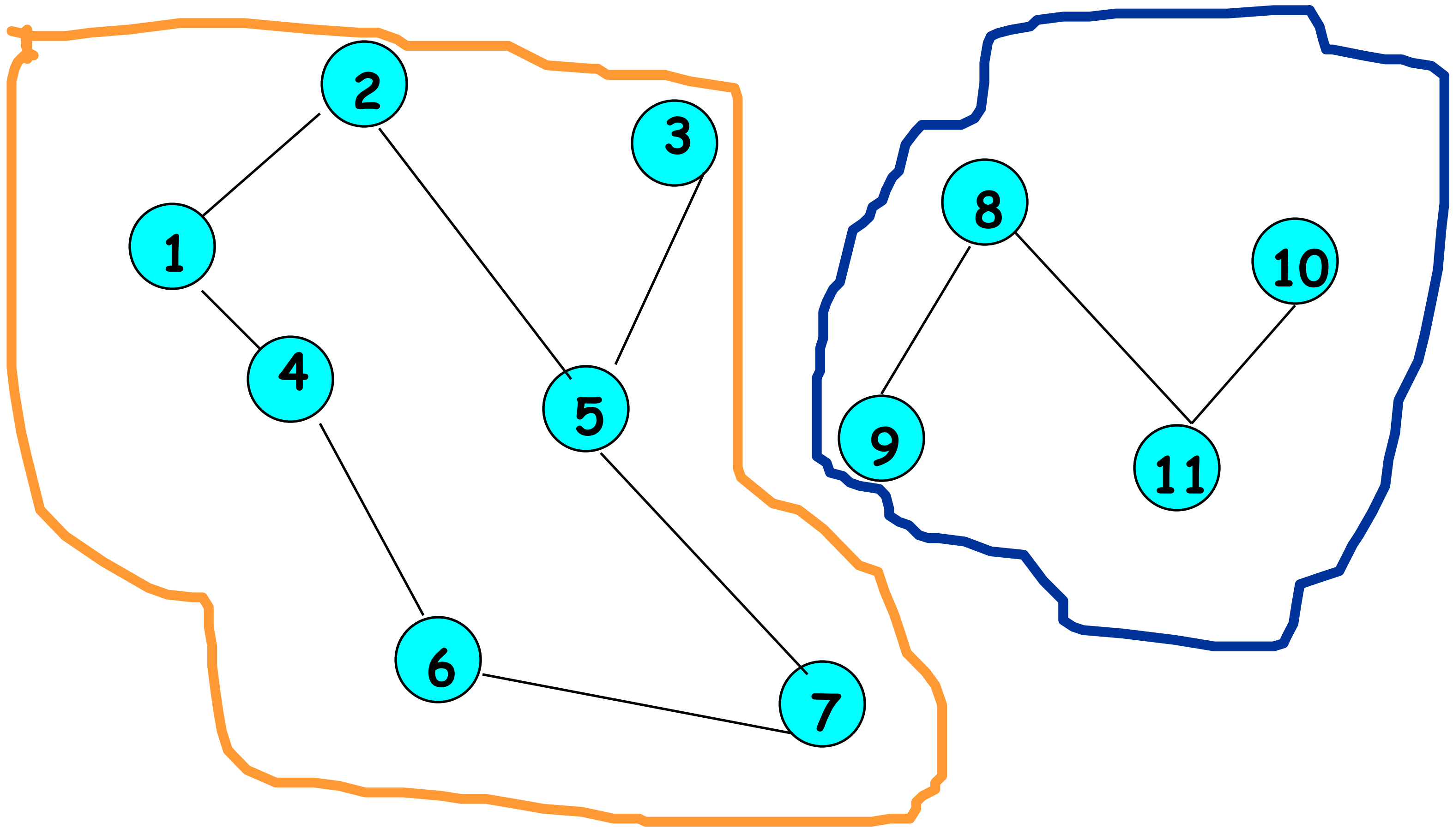
Applications: Driving Distance/Time Map



Vertex = city

Edge weight = driving distance/time.

Applications: Connected Components

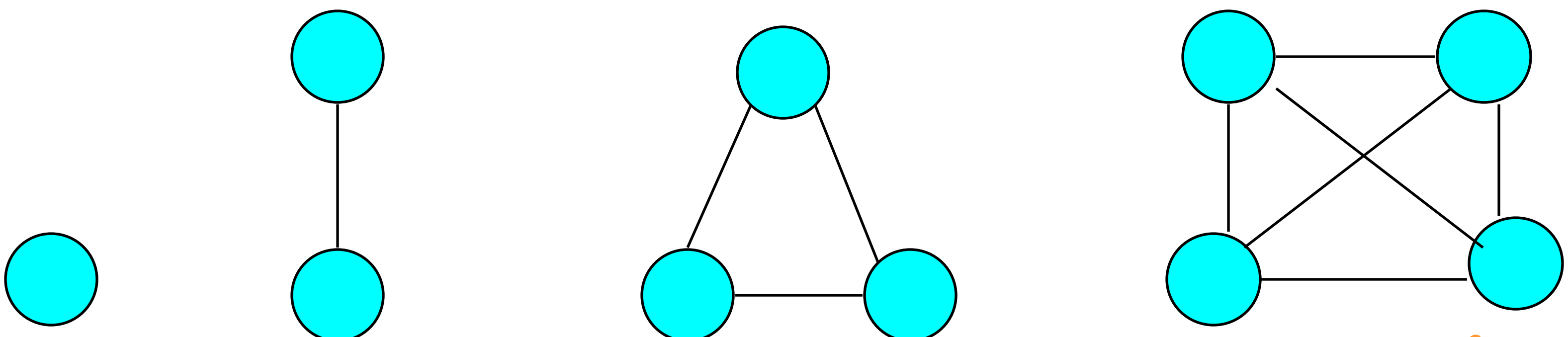


Notes

- Tree是一種特別的graph
- Graph本質上並沒有超出tree會考量使用的資料結構
  - Array, e.g., adjacent matrix
  - Linked list

Complete Undirected Graph

Has all possible edges.



n = 1 n = 2 n = 3 n = 4

Some Trivial Properties

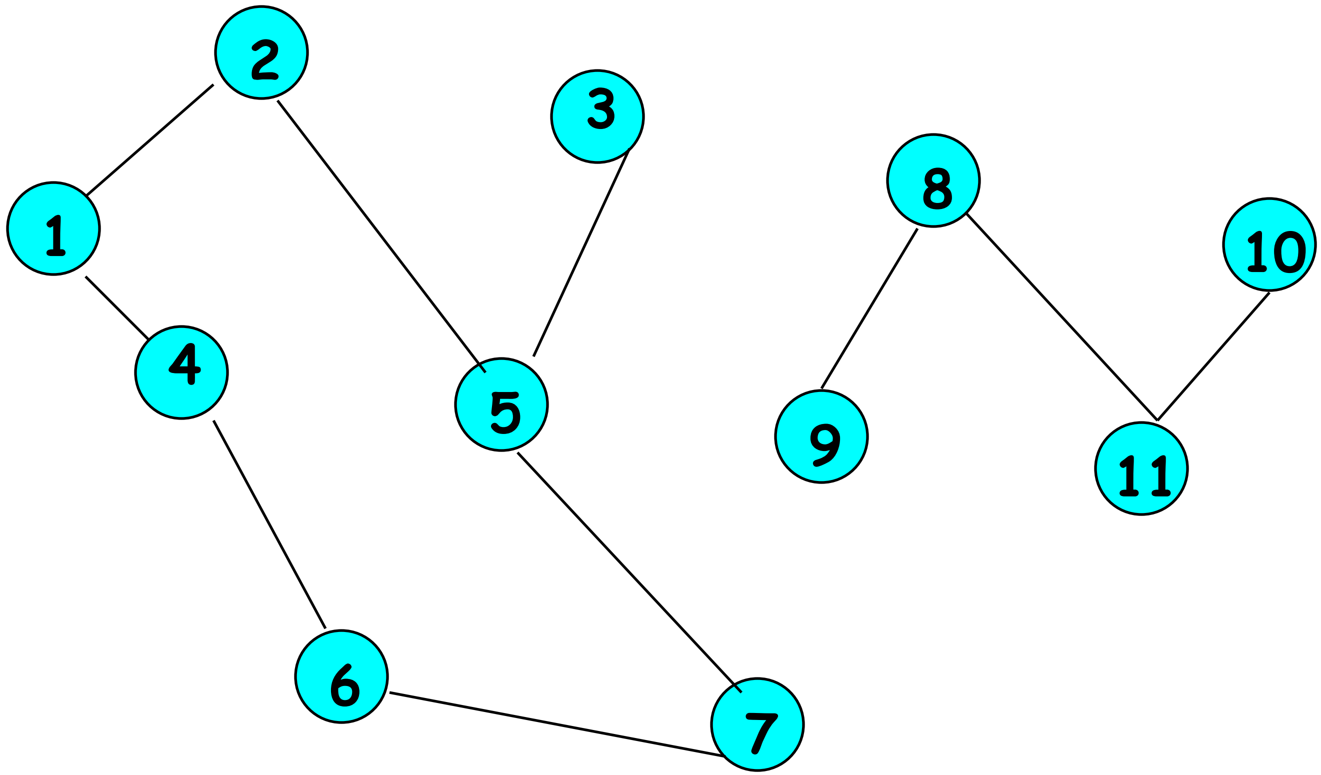
Number of Edges—Undirected Graph

- Each edge is of the form (u,v), u != v.
- Number of such pairs in an n vertex graph is n(n-1).
- Since edge (u,v) is the same as edge (v,u), the number of edges in a complete undirected graph is n(n-1)/2.
- Number of edges in an undirected graph is <= n(n-1)/2.

Number of Edges—Directed Graph

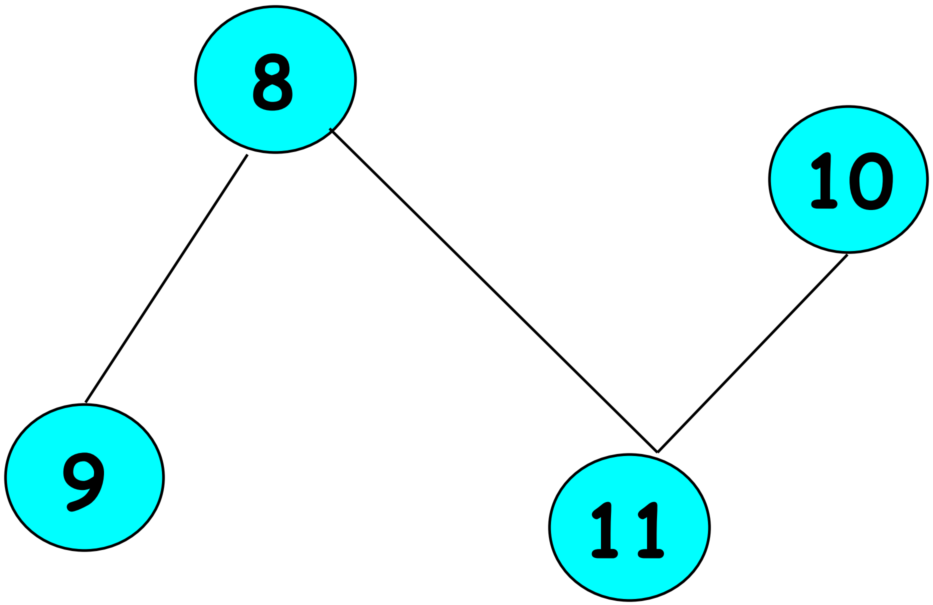
- Each edge is of the form (u,v), u != v.
- Number of such pairs in an n vertex graph is n(n-1).
- Since edge (u,v) is not the same as edge (v,u), the number of edges in a complete directed graph is n(n-1).
- Number of edges in a directed graph is <= n(n-1).

Vertex Degree



- Number of edges incident to vertex.
- degree(2) = 2, degree(5) = 3, degree(3) = 1

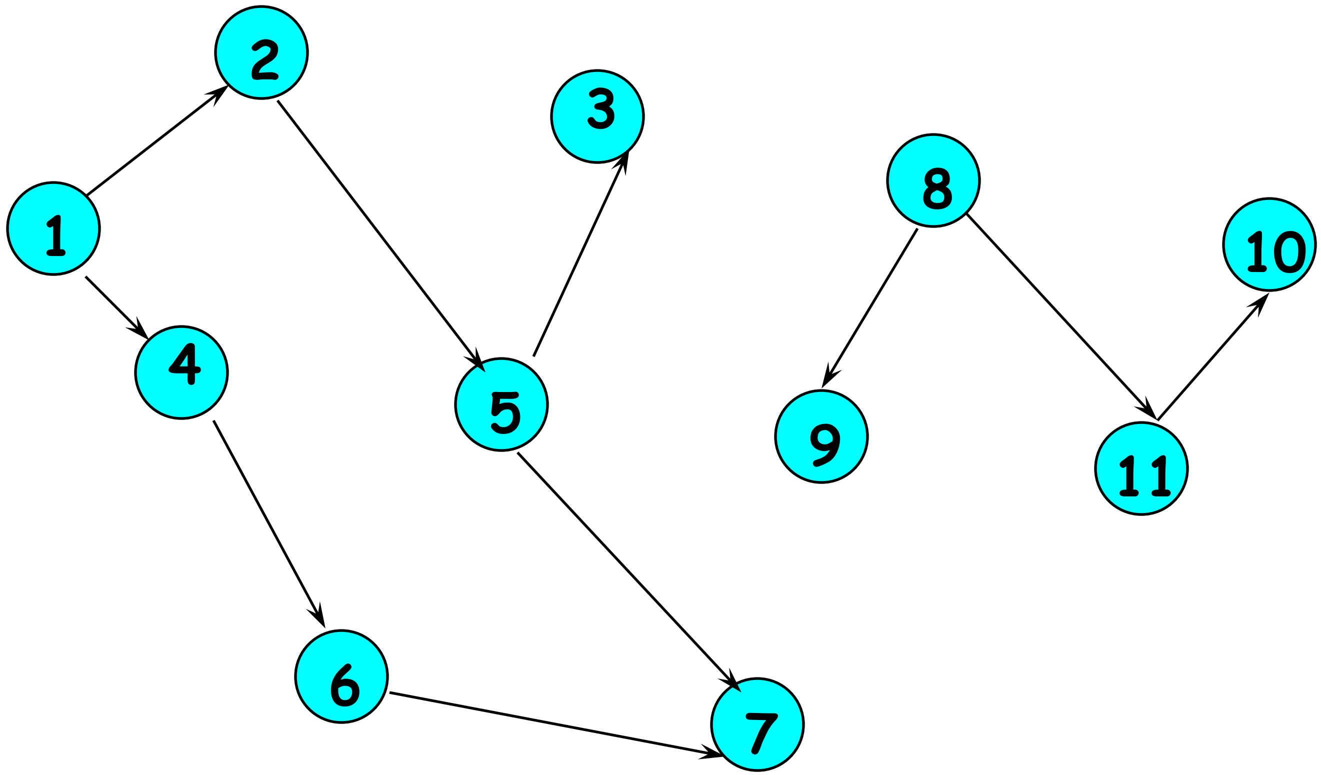
Sum of Vertex Degrees



Sum of degrees = 2e

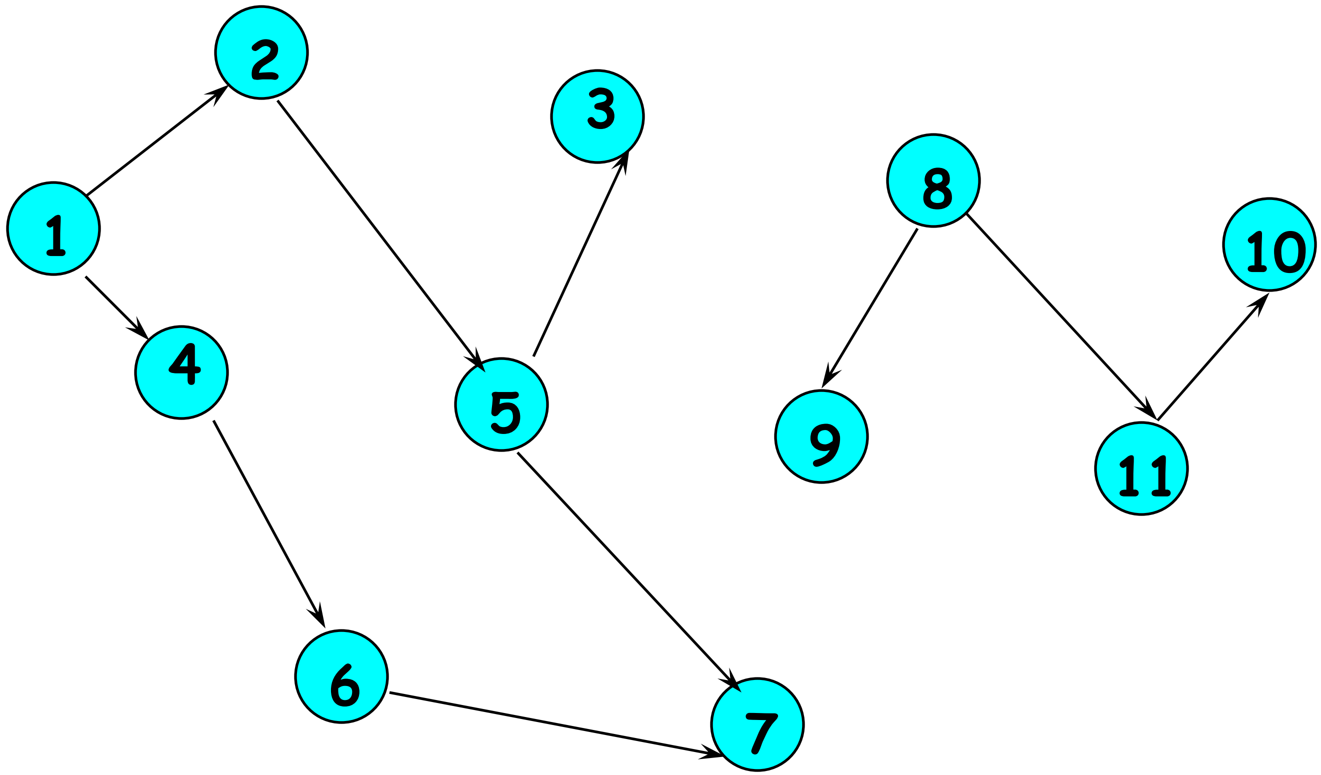
(where e is number of edges)

In-Degree of a Vertex



- in-degree is number of incoming edges
- indegree(2) = 1, indegree(8) = 0

Out-Degree of a Vertex



- out-degree is number of outbound edges
- outdegree(2) = 1, outdegree(8) = 2

Sum of In- and Out-Degrees

- each edge contributes 1 to the in-degree of some vertex and 1 to the out-degree of some other vertex

- sum of in-degrees = sum of out-degrees = e, where e is the number of edges in the digraph

Tree

- Connected graph that has no cycles.
  - Connected graph: a graph contains a single connected component
- n vertex connected graph with n-1 edges.

- Tree本質是directed graph，惟我們大部分的時候不會如此強調。Tree透過parent，children，left child (in case binary tree)，right child (in case binary tree) 等等定義了方向性。