Data Structures

HEAPS (CHAPTER 5)

Heap

A **heap** is a **complete binary tree** that satisfies the **heap-order property**.

Shape property (Structure)

- The heap is a complete binary tree:
  - Every level is completely filled, except possibly the last level.
  - The last level‘s nodes are filled from left to right.
- This property guarantees that the heap can be stored efficiently in an array (no gaps).

Heap-Order Property (Ordering)

Max heap

- Each parent node's key is always **greater than** or equal to their children (parent ≥ children).
- The **maximum** element is always at the root.

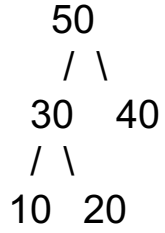
Min heap

- Each parent node's key is always **less than** or equal to their children (parent ≤ children).
- The **minimum** element is always at the root.

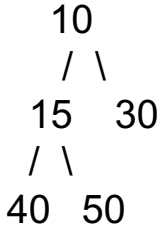
Heaps are commonly used to implement **priority queues** and for **heap sort algorithms**.

Max Heap vs. Min Heap

Max Heap



Min Heap



Array representation: `[50, 30, 40, 10, 20]`

Array representation: `[10, 15, 30, 40, 50]`

Operations and Time Complexity

Operation	Description	Time Complexity
`insert(x)`	Add new element and restore heap-order	O(log n)
`extract_root()`	Remove max (or min) element	O(log n)
`peek_root()`	Access max (or min) element	O(1)
`build_heap(array)`	Convert array to heap	O(n)
`heapify(i)`	Restore heap property at node `i`	O(log n)

Summary

Property	Meaning
Data structure type	Binary Tree (Complete)
Order property	Parent ≥ Children (Max) or ≤ Children (Min)
Stored in	Array (most efficient)
Typical uses	Priority Queue, Scheduling, Heapsort
Performance	Logarithmic insertion/removal

Why Order Means Priority in a Heap

Priority = Importance or Urgency

- In a **priority queue**, each element has a **priority value -** a higher (or lower) value means "more important" or "should be handled first."

Example	**Priority Interpretation**
Emergency patient with triage level 5	High priority
OS process with small remaining time	High priority (shortest job first)
Network packet with high QoS level	High priority

Why Order Means Priority in a Heap

Heap Order Property = Encoded Priority Rule

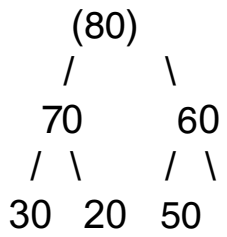
In a heap, we use the *key value* (number, weight, time, etc.) to represent the priority of each element.

- In a max-heap, the largest key = highest priority.
- In a min-heap, the smallest key = highest priority.

Summary

Concept	Meaning in Heap
Order property	Defines how nodes compare (≥ or ≤)
Key value	Represents *priority*
Root node	Element with *highest priority*
Heap operations	Maintain this priority rule efficiently (O(log n))

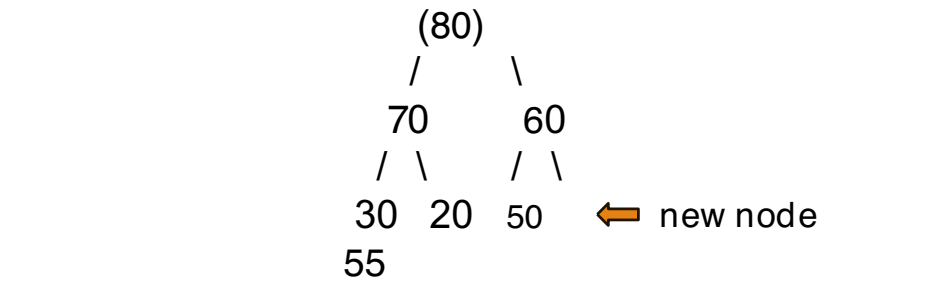
Example: Max Heap



Array representation: `[80, 70, 60, 30, 20, 50]`

Example: Max Heap (insert 55)

Insert 55

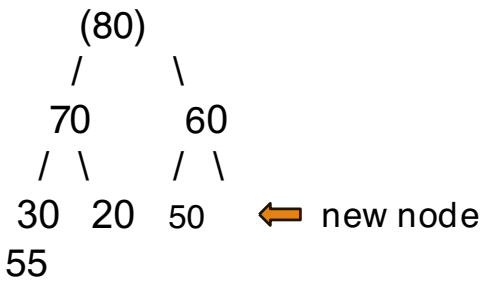


Array representation: `[80, 70, 60, 30, 20, 50]`

Array representation: `[80, 70, 60, 30, 20, 50, 55]`

Example: Max Heap (insert 55)

Check order after inserting 55,

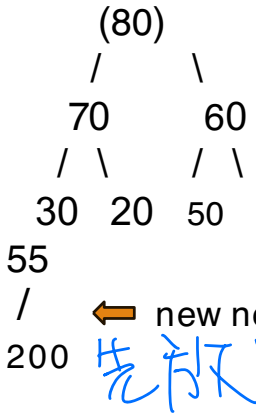


Array representation: `[80, 70, 60, 30, 20, 50]`

Array representation: `[80, 70, 60, 30, 20, 50, 55]`

Example: Max Heap (insert 200)

Insert 200



先放最後

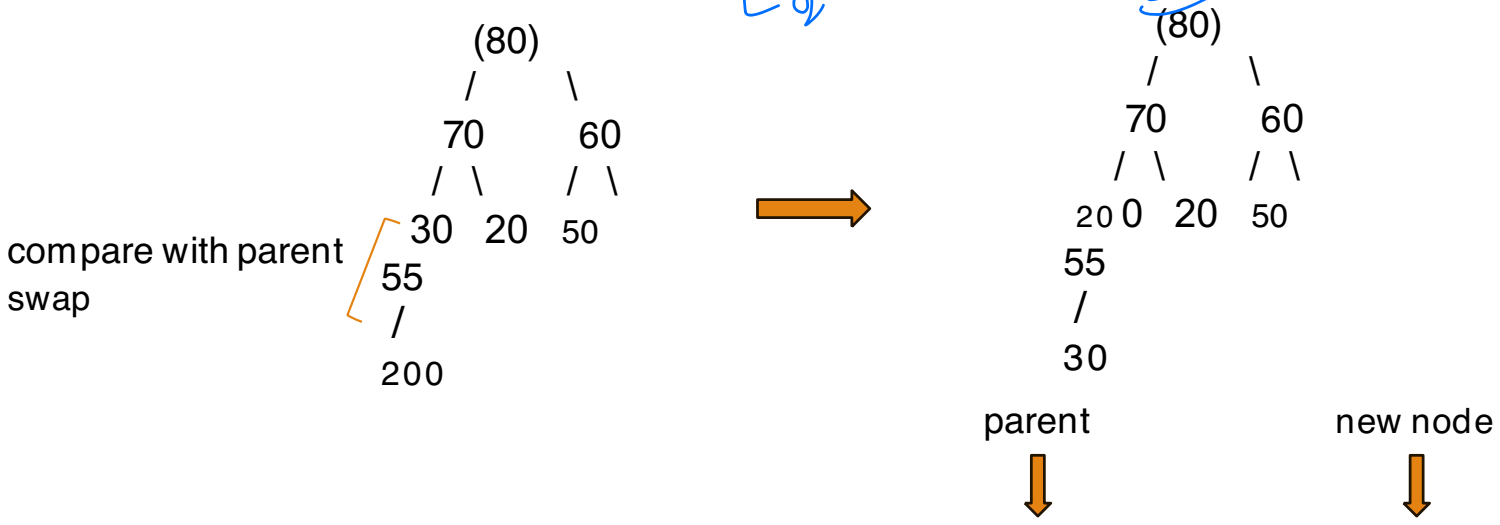
Array representation: `[80, 70, 60, 30, 20, 50, 55, 200]`

開始調整

Example: Max Heap (insert 200)

Check order after inserting 200

一路往上檢查



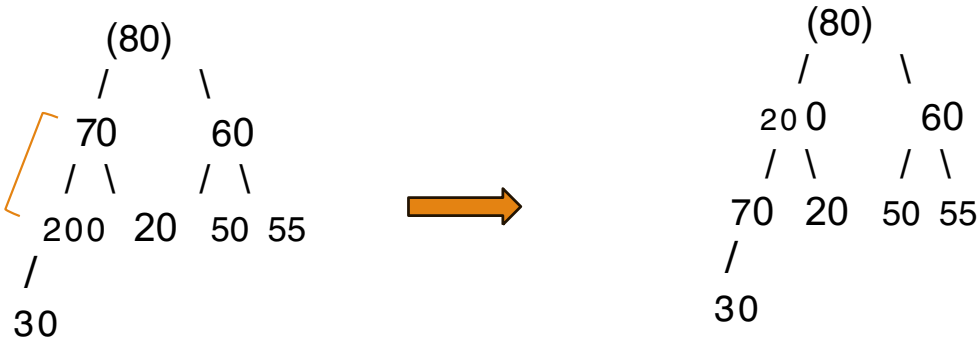
Array representation: `[80, 70, 60, 30, 20, 50, 55, 200]`

Array representation: `[80, 70, 60, 200, 20, 50, 55, 30]`

Example: Max Heap (insert 200)

Check order after inserting 200

compare with parent
swap



parent new node

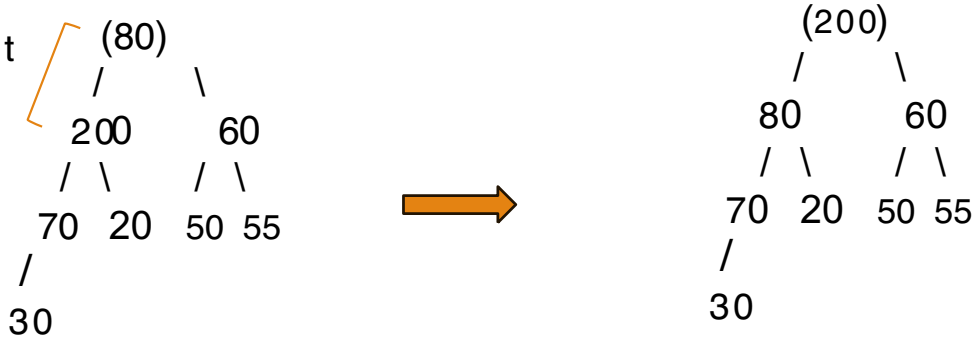
Array representation: [80, 70, 60, 200, 20, 50, 55, 30]

Array representation: [80, 200, 60, 70, 20, 50, 55, 30]

Example: Max Heap (insert 200)

Check order after inserting 200

compare with parent
swap



parent new node

Array representation: `[80, 200, 60, 70, 20, 50, 55, 30]`

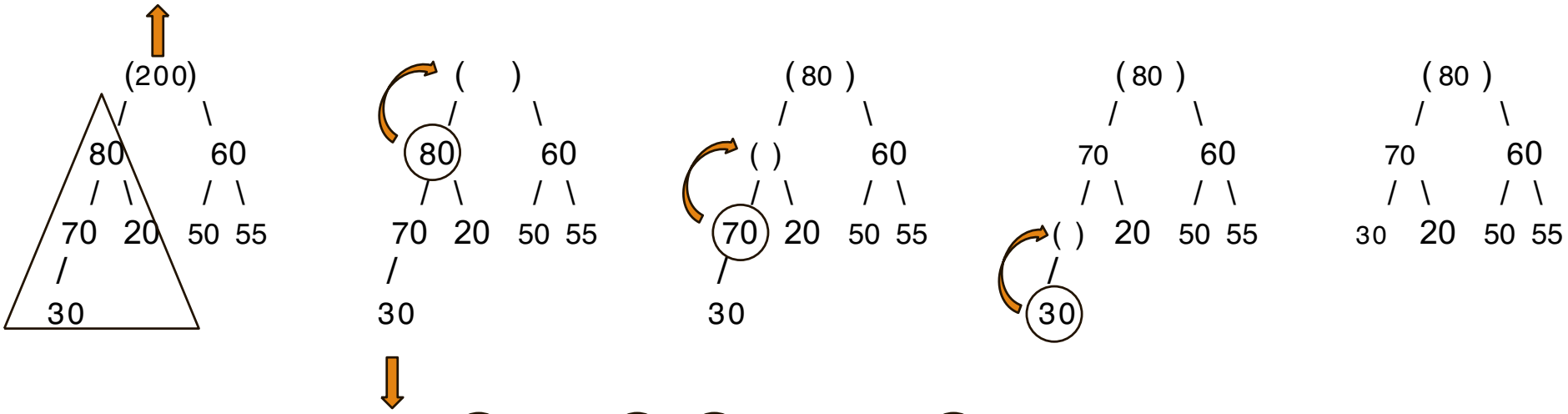
Array representation: `[200, 80, 60, 70, 20, 50, 55, 30]`

用 array

找替補節點

Example: Max Heap (Extract Max)

Extract Max (200)



Array representation: [200, 80, 60, 70, 20, 50, 55, 30]

Array representation: [80, 70, 60, 30, 20, 50, 55]