Student ID: 1133336 Student Name: 劉繐慈

Course: Data Structures (CSE CS203A)

Assignment V: Tree

Due date: 2025.12.30 23:59:59

Important Notice – Use of AI Tools

In this assignment, you must use at least one AI assistant (e.g. ChatGPT, Gemini, Claude, Grok, M365 Copilot) as a learning tool to help you:

- review definitions,
- compare tree variants, and
- organize your report.

You are not allowed to let the AI directly produce your final diagrams or final report content without your own understanding and rewriting.

You must log all AI prompts and services used (see "AI Usage Log" section below).

1. Goal of This Assignment

In the lectures, we introduced the concept of the tree as a data structure, starting from the general tree and then moving to more specialized forms.

In this assignment, you will:

a. Understand and clearly define:
   1. General tree
   2. Binary tree
   3. Complete binary tree
   4. Binary search tree (BST)
   5. AVL tree
   6. Red-Black tree
   7. Max heap
   8. Min heap

b. Build a hierarchy and transformation path from the general tree to these variants, and explain how each variant adds more structure or constraints.

c. Use a fixed list of integers to construct multiple tree variants and visualize them.

d. Choose one real-world application for each tree type and explain why that data structure fits the application.

e. Practice using AI tools as study companions and keep a simple Q&A log.

2. Given Data

Use the following 20 integers as the input data for all your tree constructions:

37, 142, 5, 89, 63, 117, 24, 176, 58, 133, 92, 11, 151, 72, 39, 184, 7, 101, 54, 160

You will reuse this same sequence for every tree type (binary tree, complete binary tree, BST, AVL, Red-Black, max heap, min heap).

3. Deliverables

Please complete your work in the Student Worksheet Companion and upload it to the YZU Portal System.

Your report should include the following parts:

a. Definitions (Concept Review)

   Provide clear, concise definitions for each of the following:

   1. General tree
   2. Binary tree
   3. Complete binary tree
   4. Binary search tree (BST)
   5. AVL tree
   6. Red-Black tree
   7. Max heap
   8. Min heap

   You are encouraged to use AI tools to help you understand these concepts, but you must rewrite the definitions in your own words.

b. Hierarchy and Transformation of Tree Variants

   Based on the definitions above, build a "tree family hierarchy" that shows how these structures are related. For example:

   - General tree → Binary tree
   - Binary tree → Complete binary tree / Binary search tree
   - BST → AVL tree / Red-Black tree
   - Binary tree → Max heap / Min heap

   Tasks:

   - Draw a diagram or flow chart that shows the transformation or specialization path: general tree → binary tree → complete binary tree → BST → AVL/Red-Black, etc.
   - For each arrow (transformation), briefly explain:
     - What new constraint or property is added?
     - e.g., "Binary tree = tree with at most 2 children per node", "BST = binary tree with left < root < right", "AVL = BST with strict height-balance rule", etc.

c. Tree Construction with the Given Integers

   Using the given 20 integers, construct the following tree variants:

   1. Binary tree
   2. Complete binary tree
   3. Binary search tree (BST)

4. AVL tree
5. Red-Black tree
6. Max heap
7. Min heap

Important Hint / Restriction:

- For these trees, you must use tree visualization tools (e.g., online visualizers or software) to build and display the tree.
- You may not ask AI tools to directly generate the final tree pictures for you.
- Instead:
    - Use AI only to help you understand algorithms,
    - Then apply those algorithms in a visualizer (or your own implementation).

What to submit for this part:

For each tree type:

- A snapshot (image) of the constructed tree.
- The URL / name of the visualization tool you used.
- A short note on how you inserted the integers (e.g., “insert in the given order as BST”, “build max heap using heapify”, etc.).

d. Application Example for Each Tree

For each of the following:

1. Binary tree
2. Complete binary tree
3. Binary search tree (BST)
4. AVL tree
5. Red-Black tree
6. Max heap
7. Min heap

Choose one application (real-world or system-level) and explain:

1. Application description
    - e.g., priority scheduling, dictionary lookup, memory allocation, database indexing, etc.
2. Why this tree structure fits
    - What property of this data structure makes it suitable?
    - Example:
        - Max heap → good for priority queue because the largest element is always at the root, so extracting max is efficient.
        - Red-Black tree → good for standard library maps/sets because it guarantees O(log n) operations even under many insertions/deletions.

Your explanation should show that you understand the link between the data structure and its use case.

e. Report Layout and Organization

You are free to design the layout of your report, but it should:

- Be well-structured (use sections, headings, tables, and diagrams).
- Have a clear flow from:
    - definitions →
    - hierarchy/transformation →
    - constructed trees →
    - applications →
    - AI usage log.
- Be easy for another student to read and learn from.

Feel free to use AI to suggest a good outline, but you must decide and finalize the layout yourself.

f. AI Usage Log (Q&A Table)

Every time you use an AI copilot service for this assignment, record:

- Index (1, 2, 3, …)
- Prompt (what you asked)
- Service (e.g., ChatGPT, Gemini, Copilot, …)

Example log table:

Index	Prompt	Service
1	Assist me to have the definition of general tree, binary tree, complete binary tree, binary search tree, AVL tree, red-black tree, max heap and min heap for self-learning.	ChatGPT
2	Explain the difference between AVL tree and Red-Black tree in terms of balancing strategy and use cases.	Gemini
…	…	…

Place this table at the end of your report.

4. Evaluation (100 pts)

A possible breakdown (you can adjust if needed):

a. Concept definitions (20 pts)

- Correctness and clarity of all 8 tree type definitions.

b. Hierarchy & transformation explanation (20 pts)

- Clear diagram / explanation of how each tree variant evolves from the general tree.
- Correct identification of constraints/invariants.

c. Tree constructions & visualizations (25 pts)

- Correct constructions for each tree type using the given integers.
- Proper screenshots and tool URLs.

- Consistent insertion / heap-building strategy descriptions.

d. Applications & explanations (20 pts)
- One application per tree type.
- Clear explanation linking data structure properties to the application.

e. Report organization & AI usage log (15 pts)
- Logical report structure and readability.
- AI log completeness (all prompts listed with service names).
- Thoughtful use of AI as a learning assistant, not as a copy-paste generator.

Student ID: 1133336 Student Name: 劉繐慈

Course: Data Structures (CSE CS203A)

Assignment V: Tree

Student Worksheet Companion

Due date: 2025.12.30 23:59:59

Academic Integrity and AI Usage Statement

In this assignment, you must use AI tools (such as ChatGPT, Gemini, Claude, Grok, M365 Copilot, etc.) as learning assistants, but you must also take full responsibility for understanding and organizing your own work.

1. Permitted Use of AI Tools

   You may use AI to:
   - Review or clarify definitions and concepts.
   - Compare different tree data structures.
   - Get suggestions for report layout or examples.
   - Ask for explanations of algorithms (e.g., BST insertion, AVL rotation, heapify process).

   You should read, think about, and rewrite the content in your own words.
2. Not Permitted
   - Do not copy/paste AI-generated content directly as your final answer.
   - Do not ask AI to draw the final diagrams or directly produce the final tree screenshots.
   - Do not ask AI to complete the whole assignment report for you.
3. Your Responsibility
   - You are responsible for understanding the definitions and algorithms.
   - You are responsible for verifying whether AI answers are correct or not.
   - You must produce your own original explanations and diagrams.
4. AI Usage Log
   - You must record all AI queries related to this assignment.
   - At the end of your report, include an AI Usage Log table with: Index, Prompt, AI service name.

   By submitting this assignment, you acknowledge that you have used AI tools only as study aids, and that the final content of this assignment represents your own understanding and work.

Section 1. Definitions of Tree Variants

Task: Write your own definitions for each tree type. You may use AI for learning, but rewrite in your own words.

Student ID: 1133336 Student Name: 劉繐慈

1. General Tree
   Definition: General Tree 是由多個節點組成，每個節點可以有**任意數量的子節點**。整棵樹只有一個根節點，其餘節點各自只有一個父節點。
2. Binary Tree
   Definition: Binary Tree **每個節點最多只能有兩個子節點**，通常為左子節點與右子節點。樹的結構由根節點向下延伸。
3. Complete Binary Tree
   Definition: Complete Binary Tree 是一種二元樹，除了最後一層以外，其餘各層都被**完全填滿**，而最後一層的節點會**由左到右依序排列**，中間不會出現空缺。
4. Binary Search Tree (BST)
   Definition: Binary Search Tree 是具備排序規則的二元樹。對於任一節點，其左子樹中的所有節點值都**小於該節點值**，右子樹中的所有節點值都**大於該節點值。**
5. AVL Tree
   Definition: AVL Tree 是一種**自我平衡的二元搜尋樹**，任一節點的左右子樹高度差最多為 1。當樹失去平衡時，系統會自動透過旋轉操作來維持平衡。
6. Red-Black Tree
   Definition: Red-Black Tree 是一種**自我平衡的二元搜尋樹**，每個節點都被標記為紅色或黑色。透過特定的顏色規則與結構限制。
7. Max Heap
   Definition: Max Heap 是一種**完全二元樹**，其中每個父節點的值都**大於或等於其子節點的值**，因此整棵樹中最大的值一定位於根節點。
8. Min Heap
   Definition: Min Heap 是一種**完全二元樹**，其中每個父節點的值都**小於或等於其子節點的值**，因此整棵樹中最小的值會存放在根節點。

Section 2. Tree Family Hierarchy and Transformations

Task: Show how these structures are related (general → specialized). Use a simple diagram and explanations of what constraints are added at each step.

2.1 Tree Family Diagram

You may draw this by hand and paste a photo, or use drawing tools.

Suggested chain example (you may extend or adjust):

General Tree → Binary Tree → Complete Binary Tree

Binary Tree → Binary Search Tree → AVL / Red-Black

Binary Tree → Max Heap / Min Heap

Your Diagram:

Student ID: 1133336 Student Name: 劉繐慈

2.2 Explanation of Transformations

Fill in what new property or constraint is added at each step.

From	To	New property / constraint added
General Tree	Binary Tree	每個節點**最**多只能有兩個子節點，並區分為左子節點與右子節點。
Binary Tree	Complete Binary Tree	除最後一層外，其餘層必須**完全**填滿，且最後一層節點需由左到右依序排列，中間不可有空缺。
Binary Tree	Binary Search Tree	左子樹所有節點值小於根節點，右子樹所有節點值大於根節點。
BST	AVL Tree	任一節點的左右子樹高度差**不得超過 1**。
BST	Red-Black Tree	加入節點顏色規則，透過紅黑顏色限制維持樹的平衡。
Binary Tree	Max Heap	樹必須為完全二元樹，且滿足父節點值必須大於或等於其子節點值。
Binary Tree	Min Heap	樹必須為完全二元樹，且滿足父節點值必須小於或等於其子節點值。

Section 3. Tree Constructions Using Given Integers

Given integers (fixed for all parts):

37, 142, 5, 89, 63, 117, 24, 176, 58, 133, 92, 11, 151, 72, 39, 184, 7, 101, 54, 160

Task: For each tree type below, construct the tree using these integers, take a screenshot of the tree from your chosen tool, record the tool name/URL, and describe the insertion / heap-building procedure.

3.1 Binary Tree

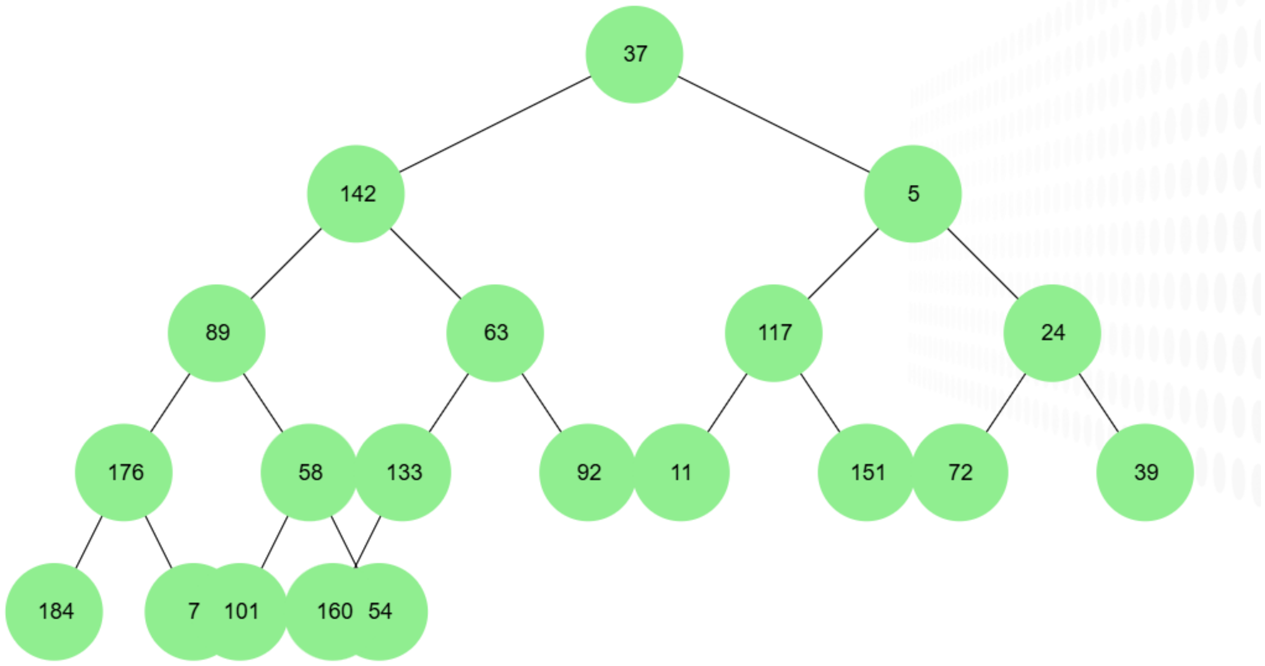
Tool name / URL:

[Tree Visualizer]

Construction / insertion description:

依題目順序插入節點，採用「層序填入」從根開始，左子空就放左子，右子空就放右子，若該節點左右都滿，就換下一個節點（用 queue）。

Screenshot of Binary Tree (paste below):



3.2 Complete Binary Tree

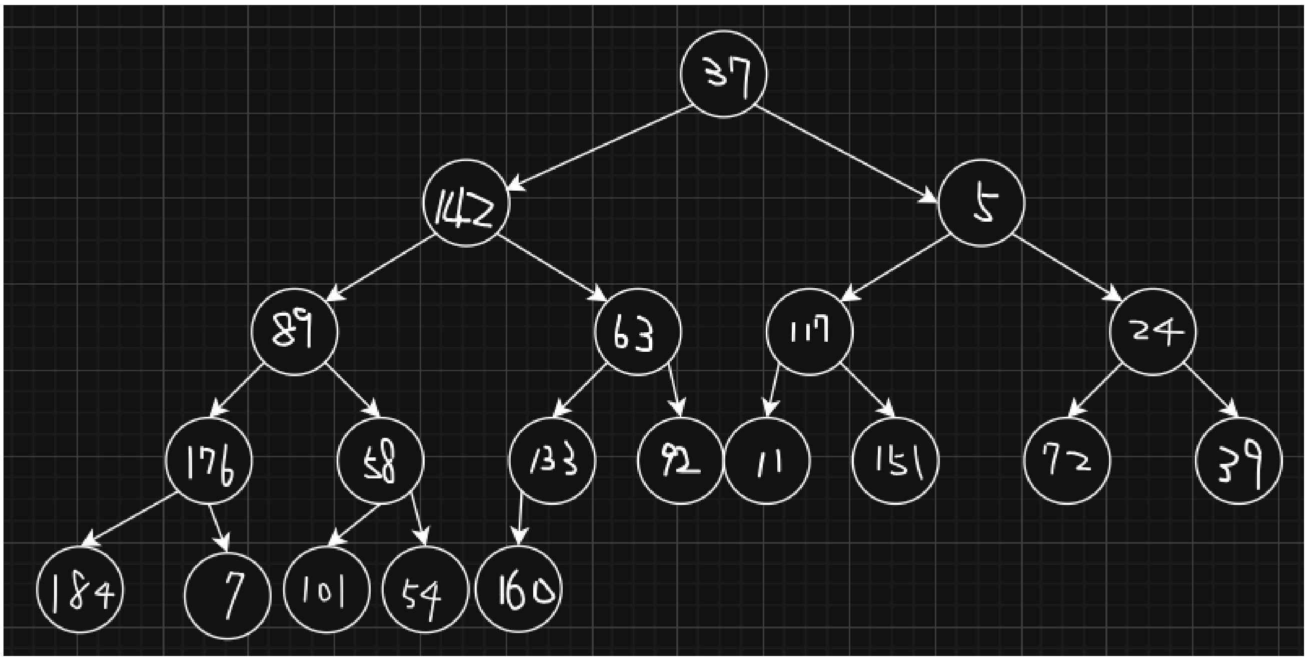
Tool name / URL:

https://app.diagrams.net

Construction / insertion description:

完全二元樹最常見的建法就是把資料依序放進陣列（array），其形狀自然是 complete（最後一層由左到右填）。

Screenshot of Complete Binary Tree (paste below):



Student ID: 1133336 Student Name: 劉繐慈

3.3 Binary Search Tree (BST)

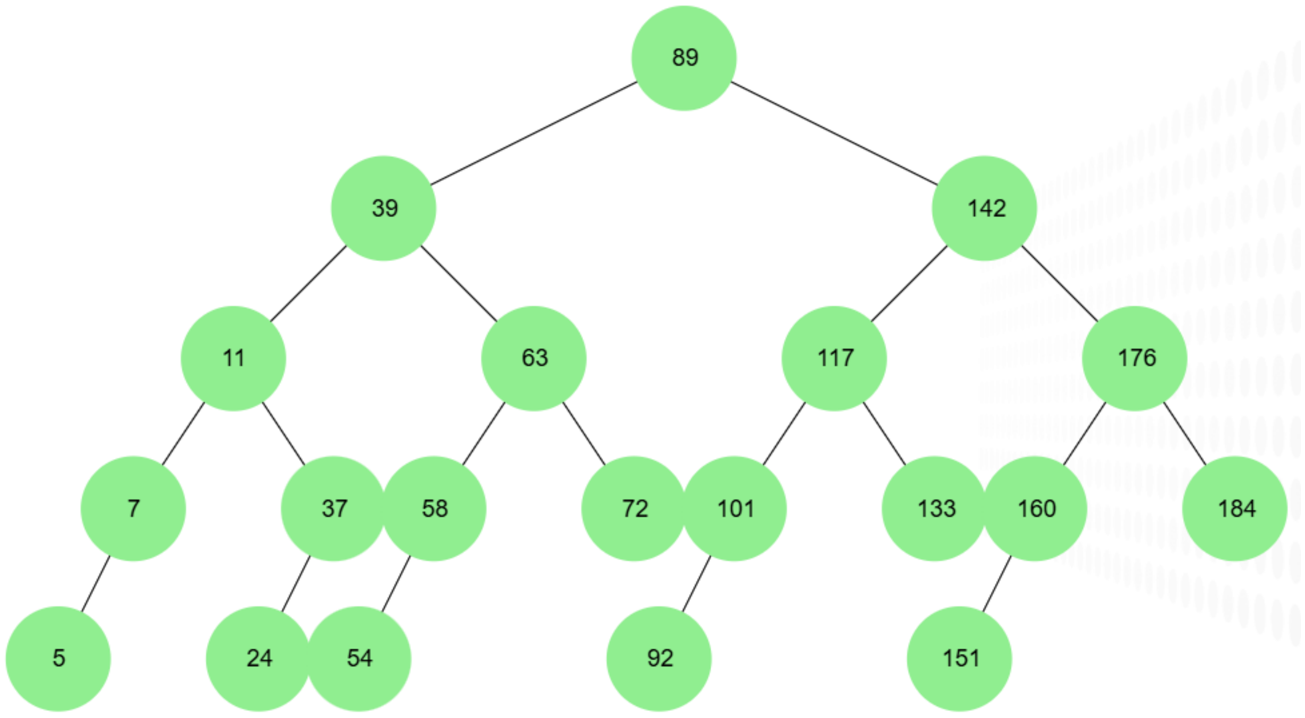
Tool name / URL:

Tree Visualizer

Insertion rule (e.g., “insert in given order using BST rules”):

依給定順序插入，比目前節點小往左，比目前節點大往右，直到空位放入。

Screenshot of BST (paste below):



3.4 AVL Tree

Tool name / URL:

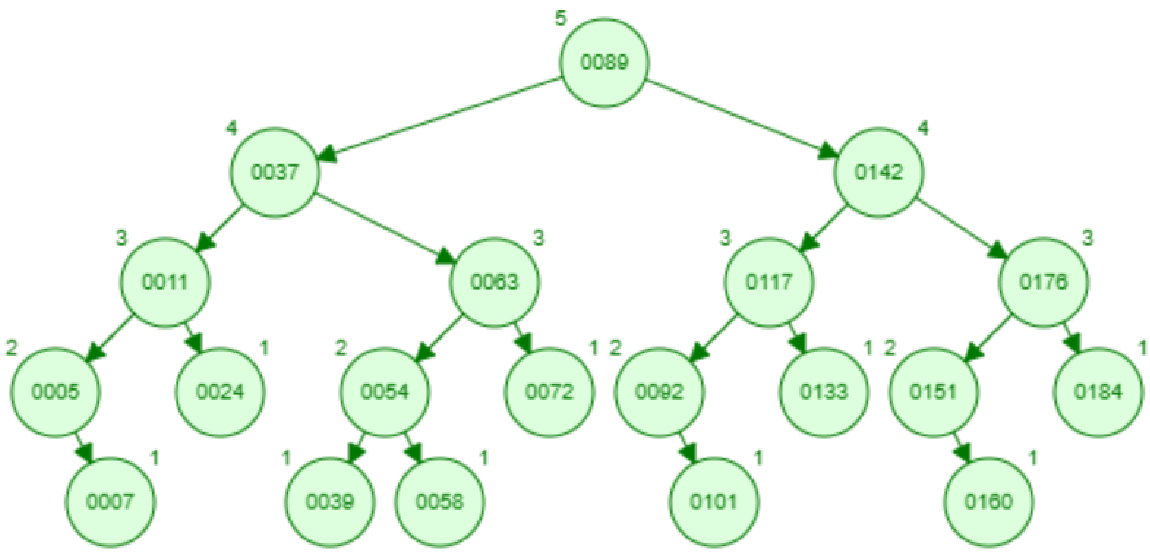
AVL Tree / https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insertion & balancing description:

先照 BST 插入，每插入一個值，就從插入點往上更新高度與 BF，若 BF 超過 ±1，依 LL/RR/LR/RL 做旋轉恢復平衡。

Screenshot of AVL Tree (paste below):

Student ID: 1133336 Student Name: 劉繐慈



3.5 Red-Black Tree

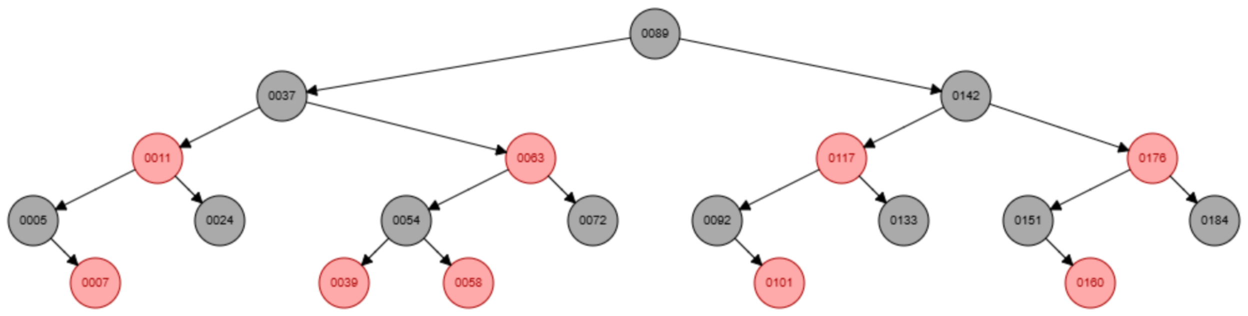
Tool name / URL:

Red/Black Tree / https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Insertion & balancing description:

先依 BST 規則插入新節點（新節點預設紅），若出現「連紅」或黑高不一致，透過「變色 + 旋轉」修正，直到符合紅黑規則。

Screenshot of Red-Black Tree (paste below):



3.6 Max Heap

Tool name / URL:

Tree Visualizer

Construction / heap-building description (e.g. heapify, insert-and-sift-up):

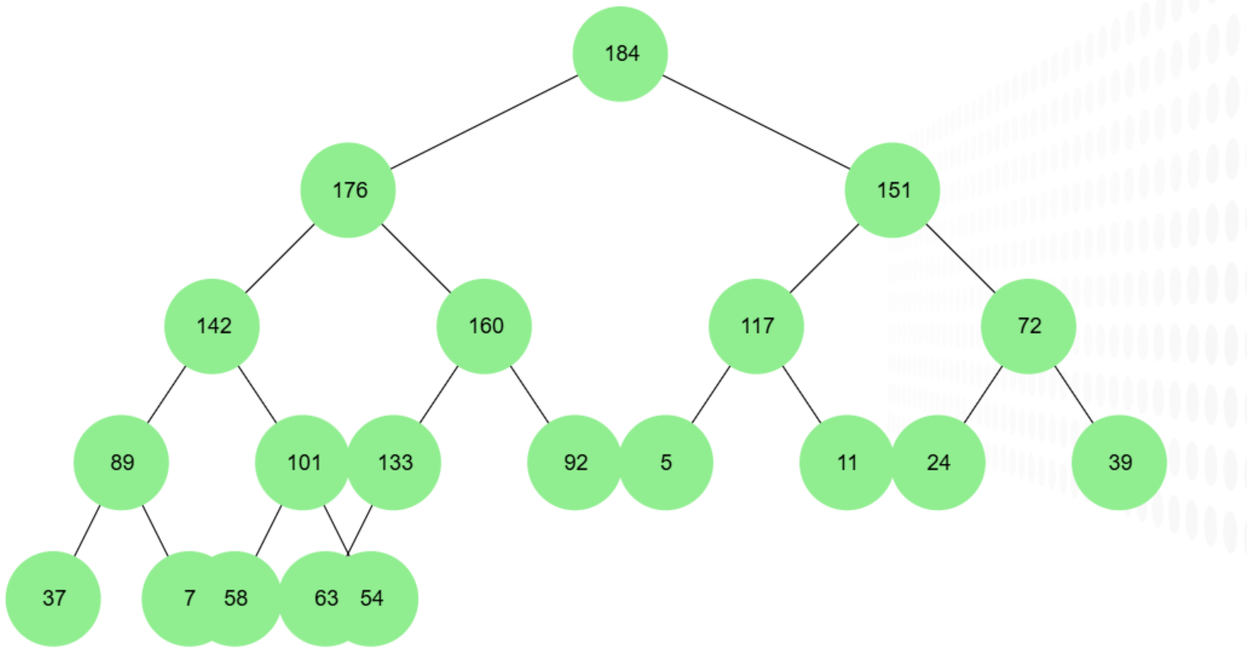
Heapify：把資料先當成 complete binary tree，從最後一個非葉節點往上做 sift-down。

或

Insert + sift-up：每插入一個新元素就往上交換直到父 ≥ 子。

Screenshot of Max Heap (paste below):

Student ID: 1133336 Student Name: 劉繐慈



3.7 Min Heap

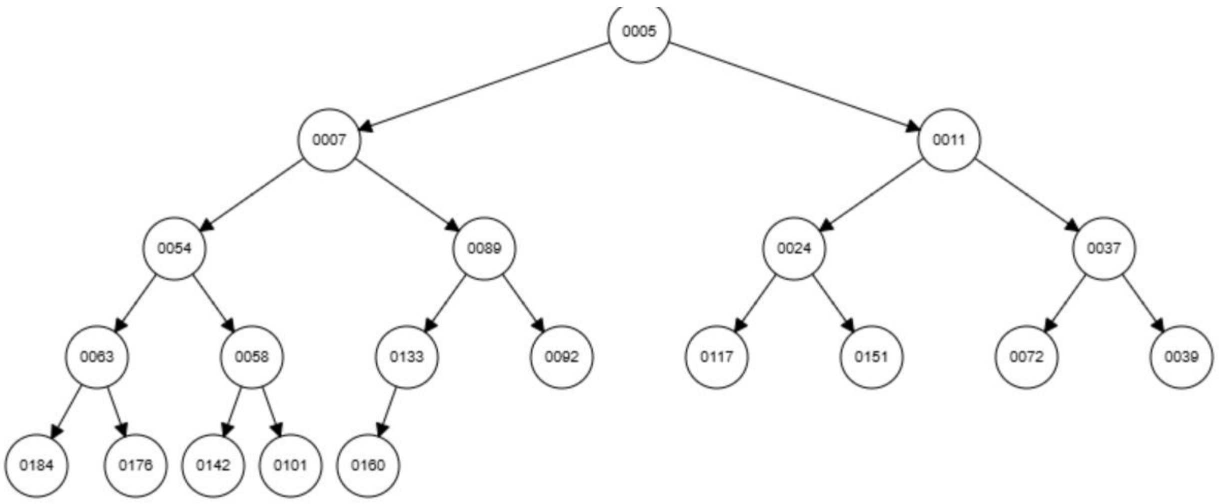
Tool name / URL:

Min Heap / https://www.cs.usfca.edu/~galles/visualization/Heap.html

Construction / heap-building description:

同上（heapify 或 insert+sift-up），但條件改成父 ≤ 子。

Screenshot of Min Heap (paste below):



Section 4. Application Examples

Task: For each tree type, choose one application and explain why this tree is suitable.

Student ID: 1133336   Student Name: 劉繐慈

Tree Type	Application Example (name / context)	Why this tree fits (properties that matter)
Binary Tree	表達式樹（Expression Tree，用於計算算術運算）	每個運算子最多需要兩個運算元，剛好對應二元樹「最多兩個子節點」的結構，方便用遞迴方式計算結果。
Complete Binary Tree	陣列式資料儲存結構（如 Heap 的底層結構）	節點位置連續、沒有空洞，可直接用陣列表示，節省空間且能快速存取父子節點。
Binary Search Tree	簡單的資料查找系統（如小型資料庫索引）	具有排序性質（左小右大），搜尋、插入、刪除平均時間複雜度為 O(log n)。
AVL Tree	需要快速查詢的系統（如即時資料查詢）	自我平衡確保樹高維持在 O(log n)，即使資料依序插入，也不會退化成鏈結串列。
Red-Black Tree	程式語言內建資料結構（如 C++ map、Java TreeMap）	透過顏色規則維持近似平衡，插入與刪除成本比 AVL 低，適合頻繁修改的情境。
Max Heap	工作排程系統（依優先權執行最高優先任務）	根節點永遠是最大值，可在 O(1) 時間取得最高優先權任務。
Min Heap	最短路徑演算法（如 Dijkstra 演算法）	根節點永遠是最小值，能快速取出目前距離最短的節點，提高演算法效率。

Section 5. Reflection on Tree Family and Performance (Optional but recommended)

Among BST, AVL, and Red-Black trees, which one would you pick for:

Mostly search (few updates)? Why?

AVL Tree，因為 AVL Tree 是一種嚴格自我平衡的二元搜尋樹，能確保樹高始終維持在 $O(\log n)$。在查詢為主、插入與刪除很少的情況下，AVL Tree 提供**最穩定且快速的搜尋效能**，不會因為資料分布不均而退化
。

Frequent insertions and deletions? Why?

Red-Black Tree，因為 Red-Black Tree 也是自我平衡的 BST，但它的平衡條件比 AVL Tree 寬鬆。因此在插入與刪除時，**需要的旋轉次數較少**，整體更新成本較低，非常適合「資料常變動」的情境。

If you must store these 20 integers for static search only (no updates), which structure or representation would you prefer (sorted array + binary search, BST, AVL, etc.)? Why?

排序陣列（**Sorted Array**）＋ **Binary Search**，因為在資料不再變動的情況下，排序陣列結構最簡單、最節省空間。透過 binary search，可以在 $O(\log n)$時間內完成搜尋，而且不需要維護樹結構或平衡條件。

Section 6. AI Usage Log (Required)

Task: Record every time you ask an AI assistant about this assignment.

Index	Date / Time	AI Service (ChatGPT, Gemini, etc.)	Your Full Prompt / Question
1	12/27 22:54	ChatGPT	General Tree、Binary Tree、Complete Binary Tree、BST、AVL Tree、Red-Black Tree、Max Heap、Min Heap 的定義。
2	12/29 23:42	ChatGPT	請比較 Binary Tree、BST、AVL Tree 與 Red-Black Tree 在結構與用途上的差異。
3	12/29 23:45	ChatGPT	請說明各種樹狀結構之間的家族關係，以及從一般樹轉換成專門化樹時新增了哪些限制。
4	12/30 00:12	ChatGPT	請解釋 Complete Binary Tree 的定義，並說明它與 Binary Tree 的差別。
5	12/30 00:26	ChatGPT	給定一組整數，請說明如何使用層序（level-order）方式建構 Complete Binary Tree。
6	12/30 00:38	ChatGPT	請說明 AVL Tree 如何透過旋轉來維持平衡，以及與一般 BST 的差異。
7	12/30 00:46	ChatGPT	請說明 Red-Black Tree 的基本顏色規則，以及它如何維持樹的平衡。
8	12/30 00:53	ChatGPT	請比較 AVL Tree 與 Red-Black Tree 在搜尋效能與插入／刪除成本上的差異。
9	12/30 01:23	ChatGPT	請解釋 Max Heap 與 Min Heap 的結構特性，以及它們在實際應用上的差別。

Student ID: 1133336 Student Name: 劉總慈

10	12/30 01:46	ChatGPT	請比較 BST、AVL Tree 與 Red-Black Tree 在不同使用情境下的效能，並說明適合搜尋或更新的原因。

You may extend this table as needed.