Student ID: 1131709 Student Name: 李啓睿

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

Student ID: 1131709 Student Name: 李啓睿

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: 1131709 Student Name: 李啓睿

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: 1131709 Student Name: 李啓睿

1. General Tree
   Definition: 最基本的層次結構。一個節點可以有任意數量的子節點（0 到 N 個），除了根節點（Root）外，每個節點都有且只有一個父節點。
2. Binary Tree
   Definition: 加上了「分支限制」的一般樹。每個節點**最多只能有兩個**子節點，通常稱為「左子節點」和「右子節點」。
3. Complete Binary Tree
   Definition: 除了最後一層外，每一層都被填滿，且最後一層的節點都**由左向右**依序排列。這種結構非常適合用陣列（Array）來存儲。
4. Binary Search Tree (BST)
   Definition: 加上了「排序規則」的二元樹。對任何節點而言：左子樹的所有值都比它小，右子樹的所有值都比它大。
5. AVL Tree
   Definition: 一種「高度平衡」的 BST。它要求任何節點的左右子樹高度差（平衡因子）絕對值不超過 1。若失去平衡，會透過「旋轉」來修復。
6. Red-Black Tree
   Definition: 另一種「自平衡」的 BST。它透過給節點著色（紅或黑）並遵循特定規則（如：根是黑色、紅節點的子節點必須是黑色等）來確保樹的高度大致平衡，雖然不像 AVL 那麼嚴格，但在插入和刪除時效能更穩定。
7. Max Heap
   Definition: 一種完全二元樹，且滿足：父節點的值永遠**大於或等於**其子節點的值。根節點一定是整棵樹的最大值。
8. Min Heap
   Definition: 一種完全二元樹，且滿足：父節點的值永遠**小於或等於**其子節點的值。根節點一定是整棵樹的最小值。

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

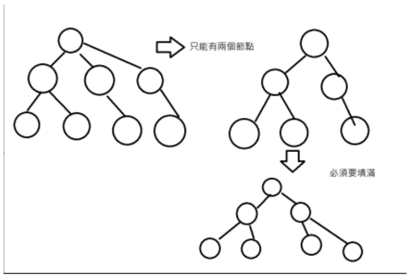
Binary Tree → Binary Search Tree → AVL / Red-Black

Binary Tree → Max Heap / Min Heap

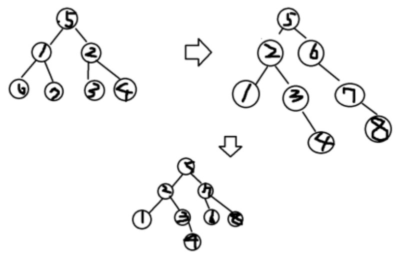
Your Diagram:

Student ID: 1131709 Student Name: 李啓睿

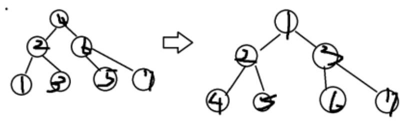
General Tree → Binary Tree → Complete Binary Tree



Binary Tree → Binary Search Tree → AVL



Binary Tree → Min Heap



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	限制每個節點最多只能有 **2** 個子節點。
Binary Tree	Complete Binary Tree	強制層次填滿順序（由上而下、由左至右），不准留空位。
Binary Tree	Binary Search Tree	增加「左小右大」的數值排序規則。
BST	AVL Tree	增加「嚴格平衡」規則（高度差 ≤ 1），確保搜尋效率為 $O(\log n)$。
BST	Red-Black Tree	增加「顏色規則」進行鬆散平衡，減少旋轉頻率，適合頻繁變動的資料。
Binary Tree	Max Heap	同時滿足「完全二元樹」結構與「父子大小關係」約束。
Binary Tree	Min Heap	同時滿足「完全二元樹」結構與「父子大小關係」約束。

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

Student ID: 1131709 Student Name: 李啓睿

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:

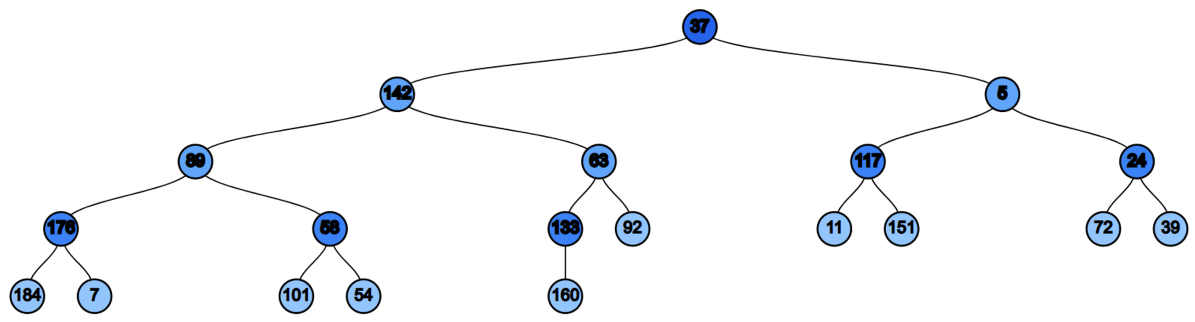
https://treeconverter.com/

Construction / insertion description:

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

數字中間逗號或空白隨意

Screenshot of Binary Tree (paste below):



3.2 Complete Binary Tree

Tool name / URL:

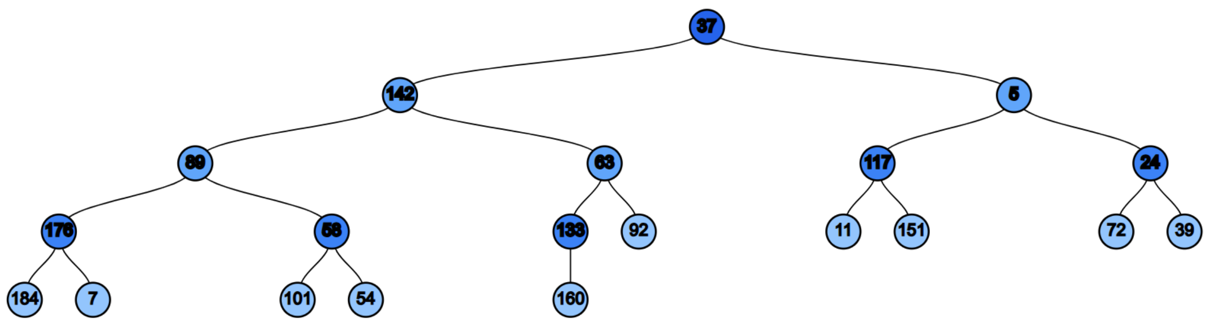
https://trees-visualizer.netlify.app/trees

Construction / insertion description:

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

數字中間逗號或空白隨意

Screenshot of Complete Binary Tree (paste below):



Student ID: 1131709 Student Name: 李啓睿

3.3 Binary Search Tree (BST)

Tool name / URL:

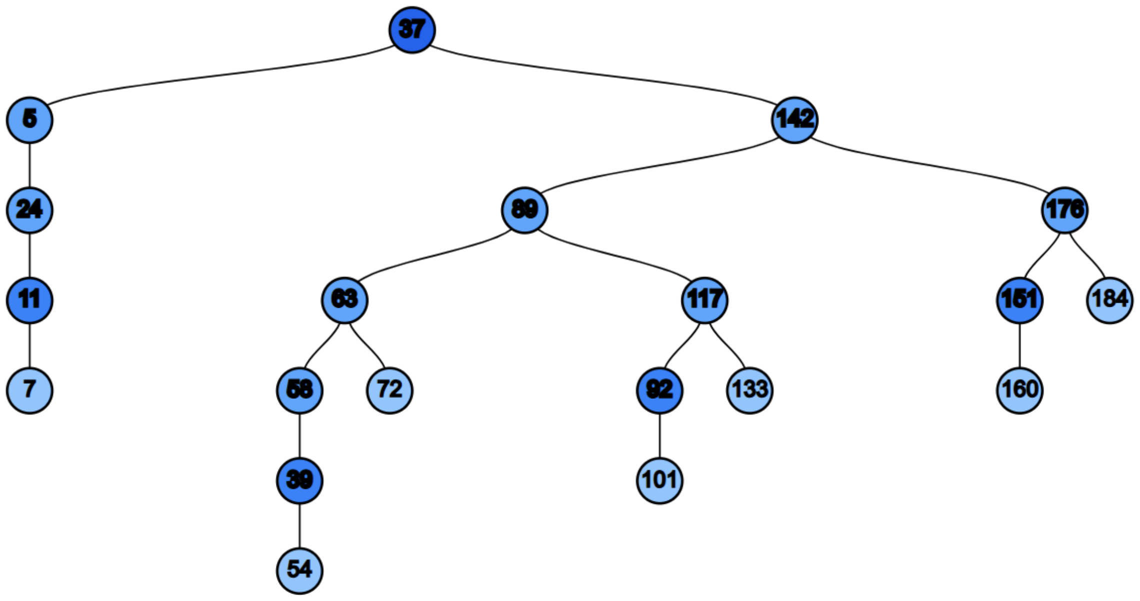
https://trees-visualizer.netlify.app/trees

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

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

數字中間逗號或空白隨意

Screenshot of BST (paste below):



3.4 AVL Tree

Tool name / URL:

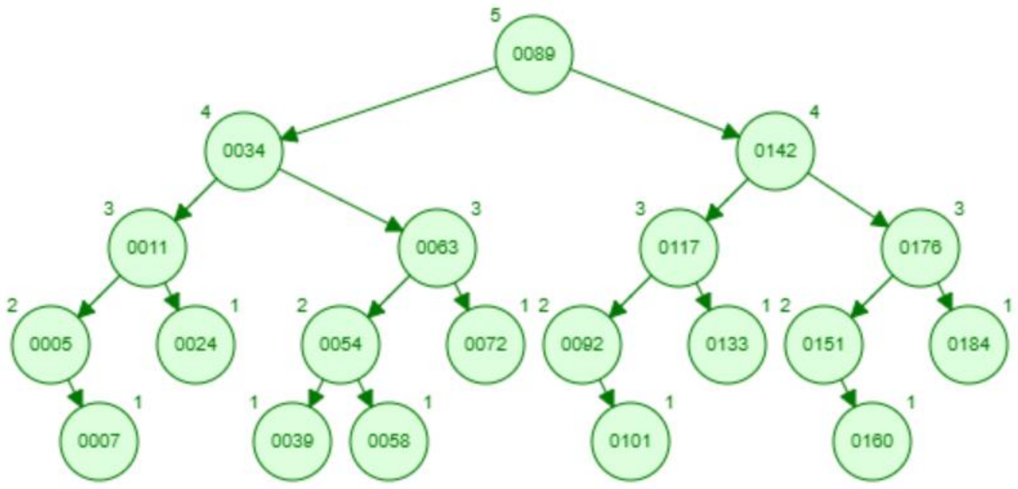
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insertion & balancing description:

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

數字要分開照順序輸入

Screenshot of AVL Tree (paste below):



3.5 Red-Black Tree

Tool name / URL:

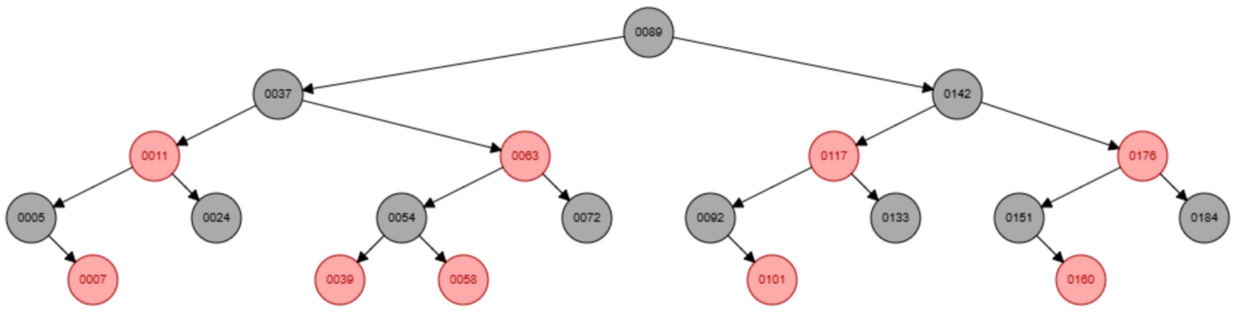
https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Insertion & balancing description:

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

數字要分開照順序輸入

Screenshot of Red-Black Tree (paste below):



3.6 Max Heap

Tool name / URL:

https://trees-visualizer.netlify.app/trees

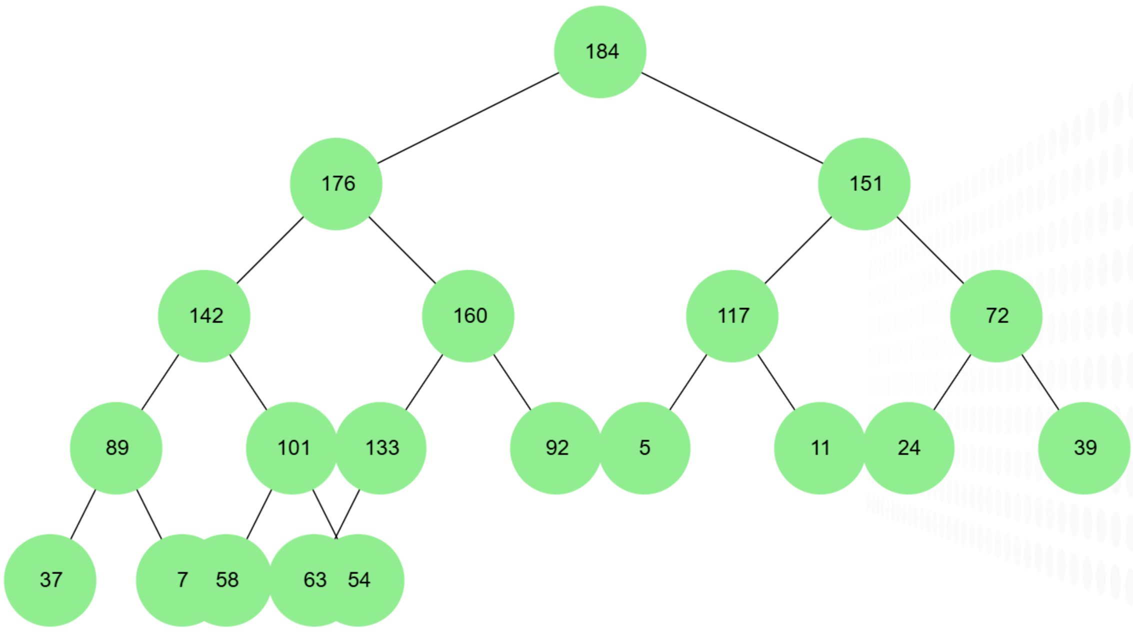
Construction / heap-building description (e.g. heapify, insert-and-sift-up):

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

數字中間空白

Screenshot of Max Heap (paste below):

Student ID: 1131709 Student Name: 李啓睿



3.7 Min Heap

Tool name / URL:

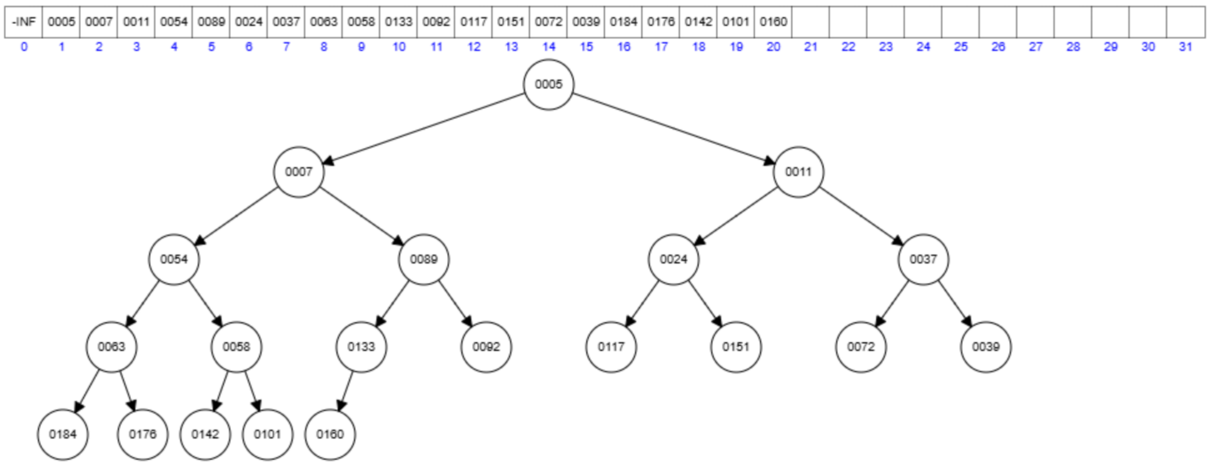
https://www.cs.usfca.edu/~galles/visualization/Heap.html

Construction / heap-building description:

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

數字要分開照順序輸入

Screenshot of Min Heap (paste below):



-INF	0005	0007	0011	0054	0089	0024	0037	0063	0058	0133	0092	0117	0151	0072	0039	0184	0176	0142	0101	0160											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Section 4. Application Examples

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

Tree Type	Application Example (name / context)	Why this tree fits (properties that matter)

Student ID: 1131709 Student Name: 李啓睿

Binary Tree	算術表達式解析 (Expression Tree)	操作數為葉子，運算符為父節點，完美對應二元運算邏輯。
Complete Binary Tree	陣列存儲優化	沒有空位，可用 $2i+1, 2i+2$ 索引直接存取，節省空間。
Binary Search Tree	簡單字典查詢	平均情況下搜尋速度快，且能輕易進行範圍查詢。
AVL Tree	靜態資料庫索引	搜尋極快且穩定。適合資料變動少、查詢多的場景。
Red-Black Tree	C++ STL std::map / Java TreeMap	插入、刪除與搜尋的綜合效能最優，維持平衡的開銷較小。
Max Heap	優先權佇列 (Priority Queue)	永遠能以 $O(1)$ 獲取最大（最高優先權）的元素。
Min Heap	Dijkstra 最短路徑演算法	用於快速選取當前距離最小的節點。

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?

BST

因為我可以很快找到我想要的數字

Frequent insertions and deletions? Why?

Red-Black trees

插入速度跟刪除可以很快

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?

BST

因為數字是有規律排放的

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	2025.12.23	Gemini	請定義 General Tree, Binary tree, BST, AVL, Red-Black Tree, heap
2	2025.12.23	Gemini	這些 tree 的應用
3	2025.12.23	Gemini	Red-Black Tree 運作方式

You may extend this table as needed.