Student ID: 1113353 Student Name: 蕭妙宣

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: 1113353 Student Name: 蕭妙宣

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: 1113353　　　　　　　　　　　　Student Name: 蕭妙宣

1. General Tree
   Definition: 有階層特性的資料結構，每個節點可以有任意數的子節點，沒有特定排序規則。

2. Binary Tree
   Definition: 每個節點最多只能有 2 個子節點，沒有特定排序規則。

3. Complete Binary Tree
   Definition: 除了最後一層，每個節點都是滿的(有 2 個子節點)。最後一層從左到右連續有節點，不可以中斷。

4. Binary Search Tree (BST)
   Definition: 每個節點最多只能有 2 個子節點，任一個節點的左子樹都小於節點，右子樹都大於節點。

5. AVL Tree
   Definition: 一種自平衡二元搜尋樹，每個節點左右子樹的高度差<=1。如果高度差>1，會旋轉直到樹平衡。

6. Red-Black Tree
   Definition: 一種自平衡二元搜尋樹，必須滿足以下 5 個規則：1. 每個節點只能是紅色或黑色。2. 根節點一定要是黑色。3. 任何紅色節點的子節點一定是黑色，不能出現連續的紅色節點。4. 從根節點到每個葉節點的路徑中，必須包含相同數量的黑色節點。5. 每個葉都是黑色。如果不滿足以上規則，會透過變色和旋轉直到樹平衡

7. Max Heap
   Definition: 一種完全二元樹，每個節點值>=子節點值，根節點為最大值。

8. Min Heap
   Definition: 一種完全二元樹，每個節點值<=子節點值，根節點為最小值。

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.

Student ID: 1113353 Student Name: 蕭妙宣

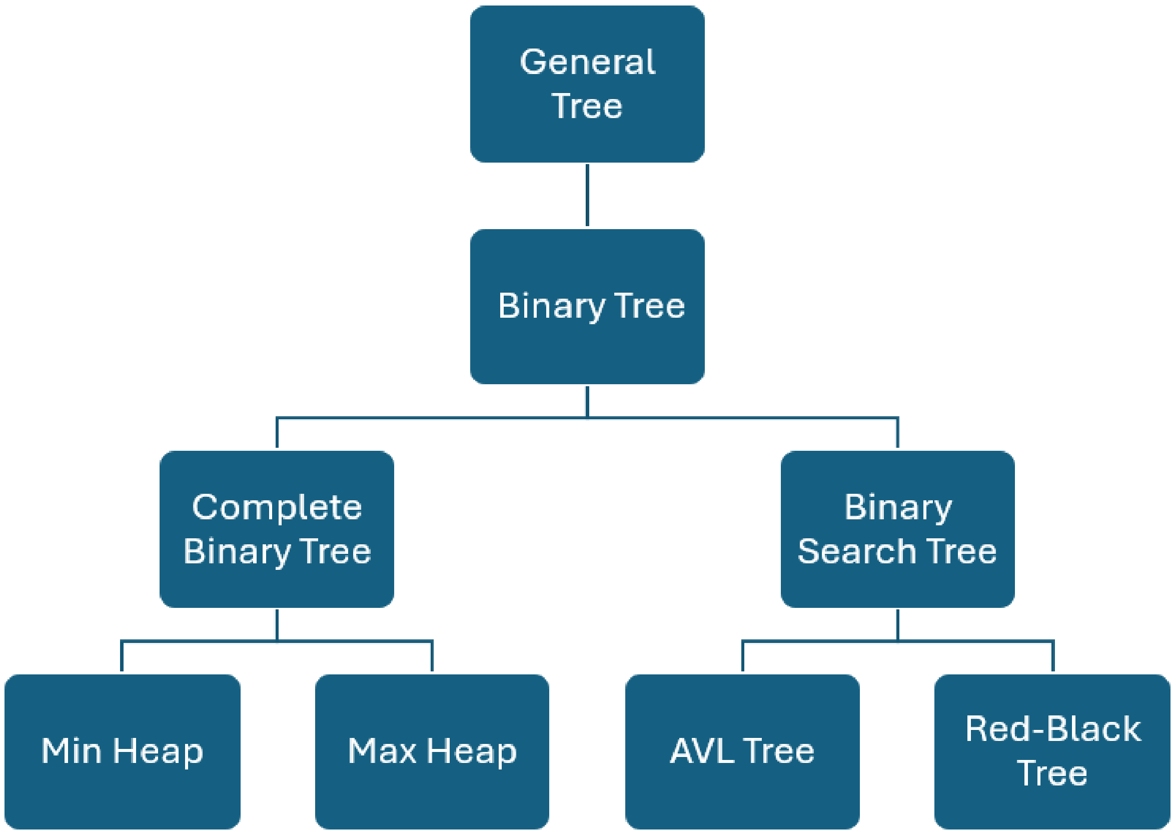
Suggested chain example (you may extend or adjust):

General Tree → Binary Tree → Complete Binary Tree

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

Binary Tree → Max Heap / Min Heap

Your Diagram:



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	每個節點最多有兩個子節點
Binary Tree	Complete Binary Tree	除了最後一層，每個節點都是滿的。最後一層從左到右連續有節點，不可以中斷
Binary Tree	Binary Search Tree	所有左子樹值<節點值，右子樹值>節點值
BST	AVL Tree	每個節點左右子樹高度差<=1，透過旋轉維持平衡
BST	Red-Black Tree	節點是紅或黑色，依照規則(無相鄰紅點、根和葉要是黑色、根到葉子黑點數相同等)，透過變色、旋轉維持平衡

Student ID: 1113353　　　　　　　　　　　Student Name: 蕭妙宣

Binary Tree	Max Heap	完全二元樹，節點值>=子節點值，根為最大值。
Binary Tree	Min Heap	完全二元樹，節點值<=子節點值，根為最小值。

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

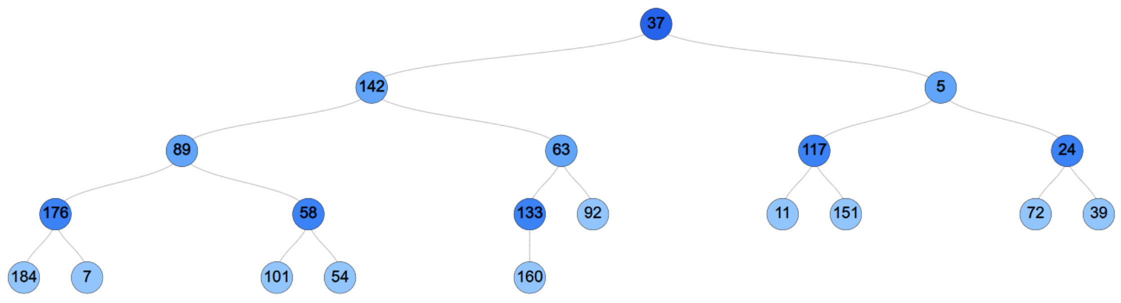
Tool name / URL:

Binary Tree Visualizer and Converter / https://treeconverter.com/

Construction / insertion description:

每個節點最多 2 個子節點，沒有其他特定規則，這裡的圖是依照順序插入根→左→右。

Screenshot of Binary Tree (paste below):



3.2 Complete Binary Tree

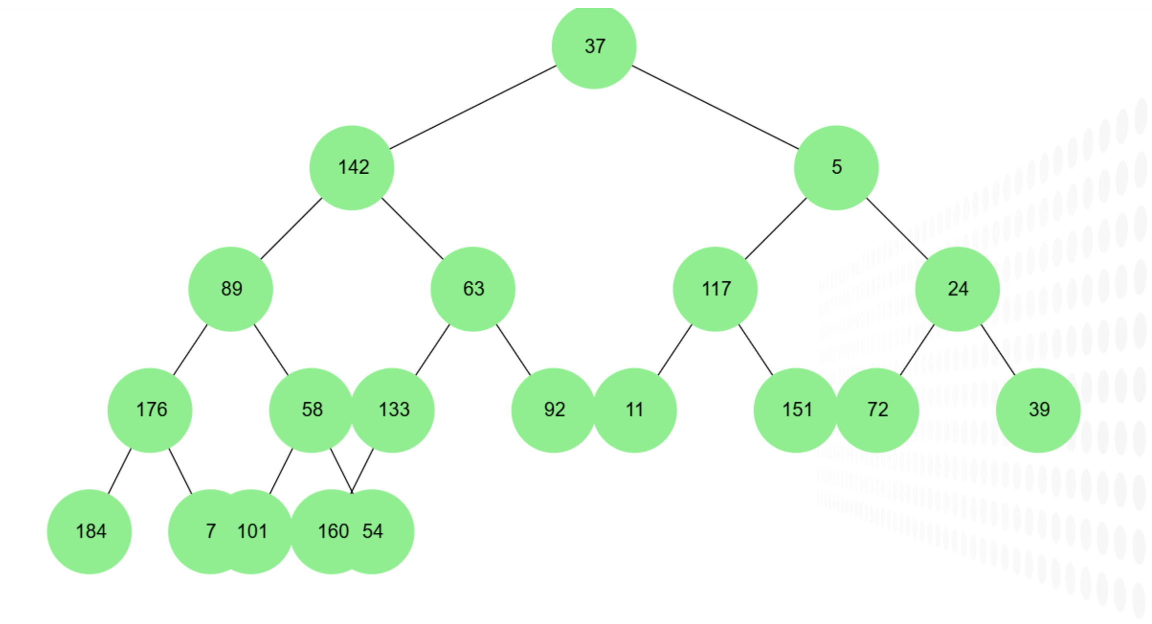
Tool name / URL:

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

Construction / insertion description:

依序插入節點，從左到右填滿每一層，填滿後再往下一層填。

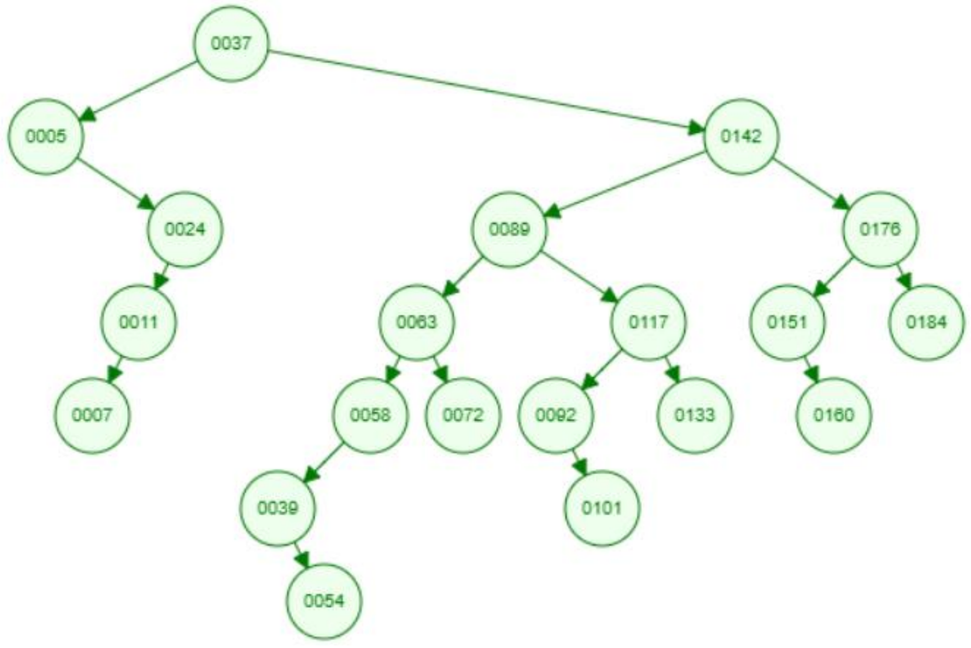
Screenshot of Complete Binary Tree (paste below):



3.3 Binary Search Tree (BST)
Tool name / URL:
Data Structure Visualizations / https://www.cs.usfca.edu/~galles/visualization/BST.html

Insertion rule (e.g., “insert in given order using BST rules”):
依序插入節點，值小於當前節點往左，大於往右。

Screenshot of BST (paste below):



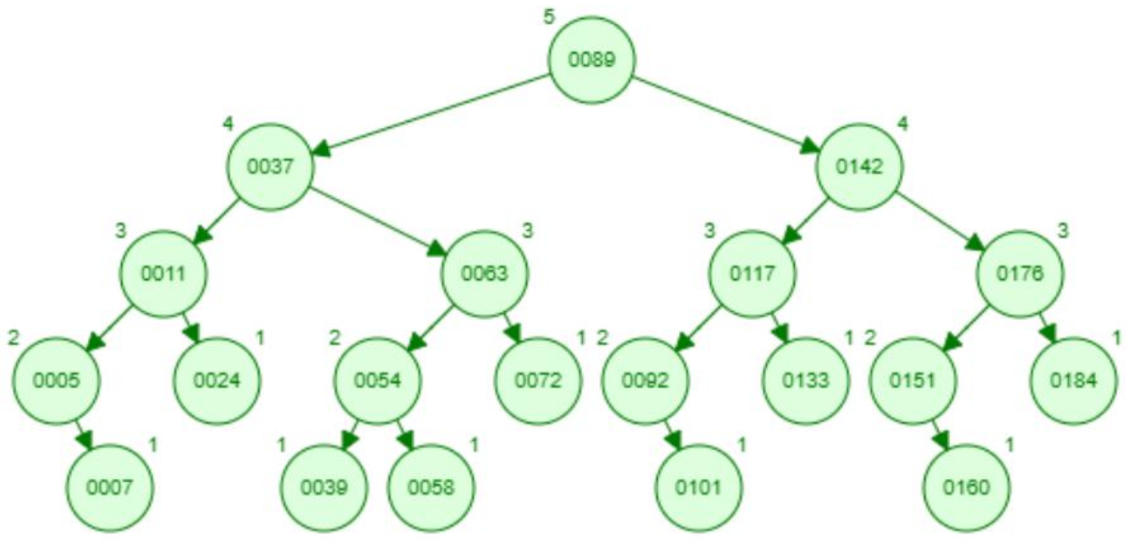
3.4 AVL Tree

Student ID: 1113353　　　　　　　　　　　　　　　Student Name: 蕭妙宣

Tool name / URL:
Data Structure Visualizations / https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insertion & balancing description:
依序插入節點，每個節點左右子樹的高度差<=1。如果高度差>1，會進行旋轉（LL, RR, LR, RL）直到樹平衡。

Screenshot of AVL Tree (paste below):

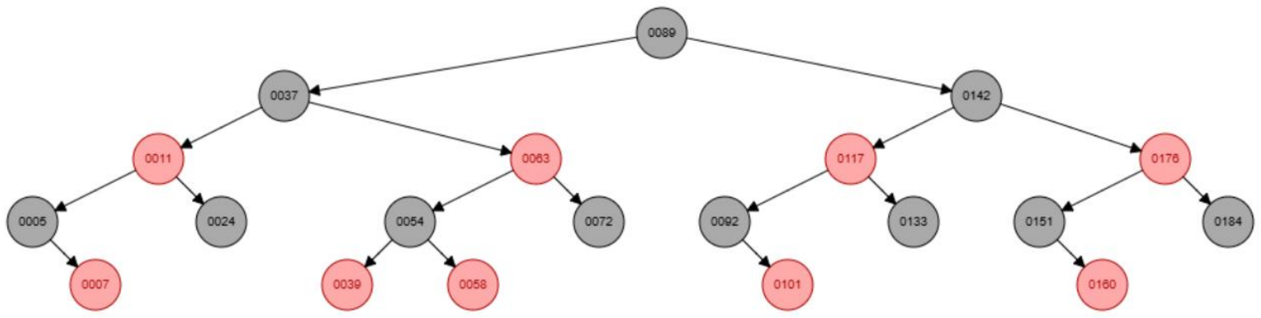


3.5 Red-Black Tree
Tool name / URL:
Data Structure Visualizations / https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

Insertion & balancing description:
依序插入節點，如果不滿足以下 5 個規則：1. 每個節點只能是紅色或黑色。2. 根節點一定要是黑色。3. 任何紅色節點的子節點一定是黑色，不能出現連續的紅色節點。4. 從根節點到每個葉節點的路徑中，必須包含相同數量的黑色節點。5. 每個葉都是黑色。會透過變色和旋轉直到樹平衡

Screenshot of Red-Black Tree (paste below):

Student ID: 1113353 Student Name: 蕭妙宣



3.6 Max Heap

Tool name / URL:

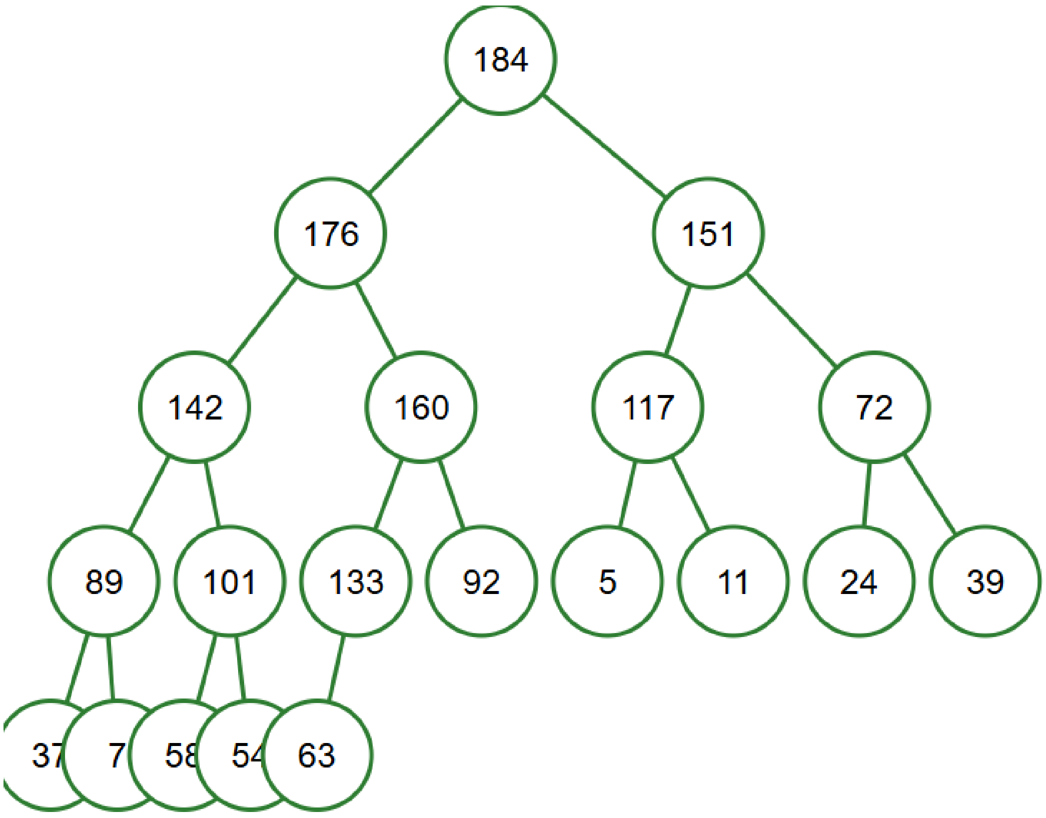
Max Heap Simulator /

https://sercankulcu.github.io/files/data_structures/slides/Bolum_08_Heap.html

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

依序插入節點，insert-and-sift-up 加入變成最後一個節點，逐一往上進行 sift-up，跟父節點比較，較大就交換，維持父 > 子的關係。

Screenshot of Max Heap (paste below):



Student ID: 1113353 Student Name: 蕭妙宣

3.7 Min Heap

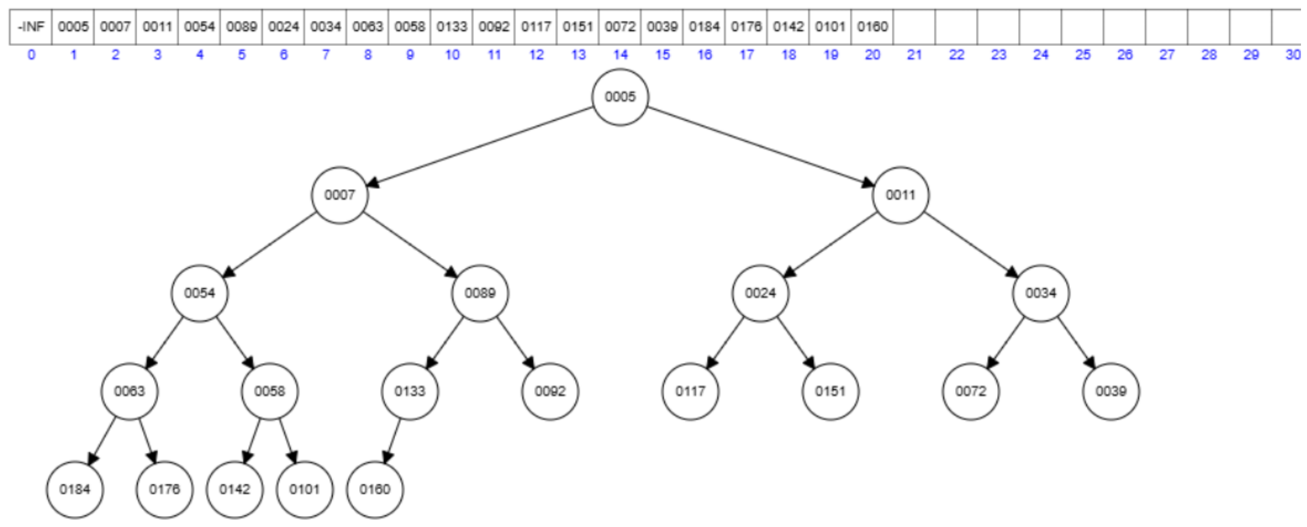
Tool name / URL:

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

Construction / heap-building description:

依序插入節點，insert-and-sift-up 加入變成最後一個節點，逐一往上進行 sift-up，跟父節點比較，較小就交換，維持父 < 子的關係。

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.

Tree Type	Application Example (name / context)	Why this tree fits (properties that matter)
Binary Tree	決策樹	每個節點代表一個決策，子節點代表不同決策結果，方便遍歷各種策略。
Complete Binary Tree	Tournament Tree 用在模擬淘汰賽	通常會設計成 Complete Binary Tree，具有最小高度、結構規則，可以用陣列實作提高比較與更新效率。
Binary Search Tree	電商商品目錄搜尋	商品依照價格或名稱排序，BST 可以快速找到商品，也支援範圍查詢。
AVL Tree	資料庫管理系統中的索引結構	樹的高度保持在較小的範圍內，複雜度最差為 O(log n)，在大型資料集上有高效的查找效能。

Red-Black Tree	Linux 預設 CPU 排程器 (CFS 和 EEVDF)	需要頻繁地插入跟移除節點 (任務)，因此開發者選擇用紅黑樹搭配一些效能調整。
Max Heap	作業系統優先順序排程	根節點一直是最大值，排序由上層到下層優先度高到低。處理優先權最高的任務時，最優先可以在 O(1)知道。
Min Heap	網路流量控制（Timer）	根節點一直是最小值，適合計時任務，每次新增任務後維持堆結構，確保任務按時間或優先順序執行。

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
AVL Tree 會比較平衡，高度最多只差 1，查找效率最情況 O(log n)比較有效率，Red-Black Tree 沒有像 AVL Tree 嚴格規定

Frequent insertions and deletions? Why?
Red-Black trees
Red-Black trees 的規則比較寬鬆，插入和刪除時需要的旋轉和重新平衡次數比較少，通常更快。

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?
sorted array + binary search
資料固定不變、無需插入或刪除時，排序陣列是最優選擇。陣列不需要指標的開銷，且連續記憶體中 cache-friendly。

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	12/29 23:30	ChatGPT	詳細解釋 這 8 種的定義 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
2	12/30 02:00	ChatGPT	推薦我畫這些樹的網頁 3.1 Binary Tree 3.2 Complete Binary Tree 3.3

			Binary Search Tree (BST) 3.4 AVL Tree 3.5 Red-Black Tree 3.6 Max Heap 3.7 Min Heap
3	12/30 04:10	ChatGPT	用 實際場景＋技術理由 把 Binary Tree Complete Binary Tree Binary Search Tree AVL Tree Red-Black Tree Max Heap Min Heap 用在真實系統中
4	12/30 04:32	ChatGPT	Linux cpu 有用到 Red-Black Tree 嗎?仔細說明
5	12/30 11:39	ChatGPT	Tournament Tree 是用在比賽?怎麼用?
6	12/30 12:24	ChatGPT	AVL and Red-Black trees 各有甚麼優缺點

You may extend this table as needed.