Student ID: Student Name:

Course: Data Structures (CSE CS203A)

Assignment V: Tree

Due date: 2025.12.30 23:59:59

Important No+ce – 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 definiOons, • 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 rewriOng.

You must log all AI prompts and services used (see “AI Usage Log” secOon below).

1. Goal of This Assignment

In the lectures, we introduced the concept of the tree as a data structure, starOng 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 transformaOon 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 mulOple tree variants and visualize them.

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

e. PracOce 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 construcOons:

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

Student ID: Student Name:

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. DefiniOons (Concept Review)

Provide clear, concise definiOons 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 definiOons in your own words.

b. Hierarchy and TransformaOon of Tree Variants

Based on the definiOons 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 transformaOon or specializaOon path:

general tree → binary tree → complete binary tree → BST → AVL/Red-Black, etc.

- For each arrow (transformaOon), 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 lek < root < right",
    "AVL = BST with strict height-balance rule", etc.

c. Tree ConstrucOon 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 / RestricOon:

- For these trees, you must use tree visualizaOon tools (e.g., online visualizers or sokware) 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 implementaOon).

What to submit for this part:

For each tree type:

- A snapshot (image) of the constructed tree.
- The URL / name of the visualizaOon 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. ApplicaOon 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 applicaOon (real-world or system-level) and explain:

1. ApplicaOon descripOon ○ e.g., priority scheduling, dicOonary lookup, memory allocaOon, 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 extracOng max is efficient.
     - Red-Black tree → good for standard library maps/sets because it guarantees O(log n) operaOons even under many inserOons/deleOons.

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

e. Report Layout and OrganizaOon

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

- Be well-structured (use secOons, headings, tables, and diagrams).
- Have a clear flow from:

- definiOons → ○ hierarchy/transformaOon → ○ constructed trees → ○ applicaOons → ○ 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 Ome 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. EvaluaOon (100 pts)

A possible breakdown (you can adjust if needed):

a. Concept definiOons (20 pts) ● Correctness and clarity of all 8 tree type definiOons.

b. Hierarchy & transformaOon explanaOon (20 pts) ● Clear diagram / explanaOon of how each tree variant evolves from the general tree.
   ● Correct idenOficaOon of constraints/invariants.

c. Tree construcOons & visualizaOons (25 pts) ● Correct construcOons for each tree type using the given integers. ● Proper screenshots and tool URLs.
   ● Consistent inserOon / heap-building strategy descripOons.

d. ApplicaOons & explanaOons (20 pts) ● One applicaOon per tree type.
   ● Clear explanaOon linking data structure properOes to the applicaOon.

e. Report organizaOon & AI usage log (15 pts) ● Logical report structure and readability. ● AI log completeness (all prompts listed with service names).

Student ID: Student Name:

- Thoughqul use of AI as a learning assistant, not as a copy-paste generator.

Student ID: Student Name:

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.

Student ID: Student Name:

1131428 徐翊甄

Section 1. Definition of Tree Variants

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

1. General Tree
   Definition:
   由點和邊構成的基本樹，每個點可以有任意數量的子節點並具有上下層關係，不能有迴圈

2. Binary Tree
   Definition:
   General Tree 增加額外規則:每個節點最多只能有兩個子節點，並且被分成左和右節點
3. Complete Binary Tree
   Definition:
   Binary tree 增加額外規則: 除了最後一層外其他層都是滿的，最後一層必須從左到右填滿
4. Binary Search Tree (BST)
   Definition:
   Binary tree 增加額外規則:任意節點的左子樹都要比他小，右邊都要比他大。
5. AVL Tree Definition:
   一種平衡樹，由 BST 增加規則: 任意節點的左子樹與右子樹的高度差不能超過 1。
6. Red-Black Tree Definition:
   一種平衡樹，由 BST 增加規則:節點分成紅與黑兩種，root 是黑節點，紅點的子節點必為黑，leaf 必是黑節點且值為 null，從任意點到其 leaf 的路徑中，黑節點數目要相同。
7. Max Heap
   Definition:
   Complete Binary Tree 增加規則: 父節點的值必大於等於子節點。
8. Min Heap
   Definition:
   Complete Binary Tree 增加規則: 父節點的值必小於等於子節點。

Student ID: Student Name:

Section 2. Tree Family Hierarchy and Transformants

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

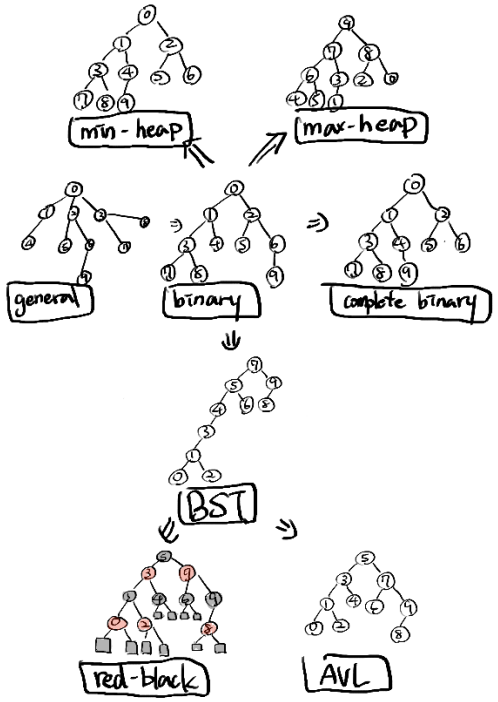
2.1 Tree Family Diagram

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:



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	每個節點最多只有兩個子節點
Binary Tree	Complete Binary Tree	樹必須由上至下，左至右填滿
Binary Tree	Binary Search Tree	左子節點值<父節點值<右子節點值
BST	AVL Tree	左右子樹高度差小於等於 1
BST	Red-Black Tree	使用顏色與顏色分佈規則區分節點
Binary Tree	Max Heap	父節點大於等於所有子節點，必須是 Complete Binary Tree
Binary Tree	Min Heap	父節點小於等於所有子節點，必須是 Complete Binary Tree

Student ID: Student Name:

Section 3. Tree Construction 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: https://trees-visualizer.netlify.app/trees

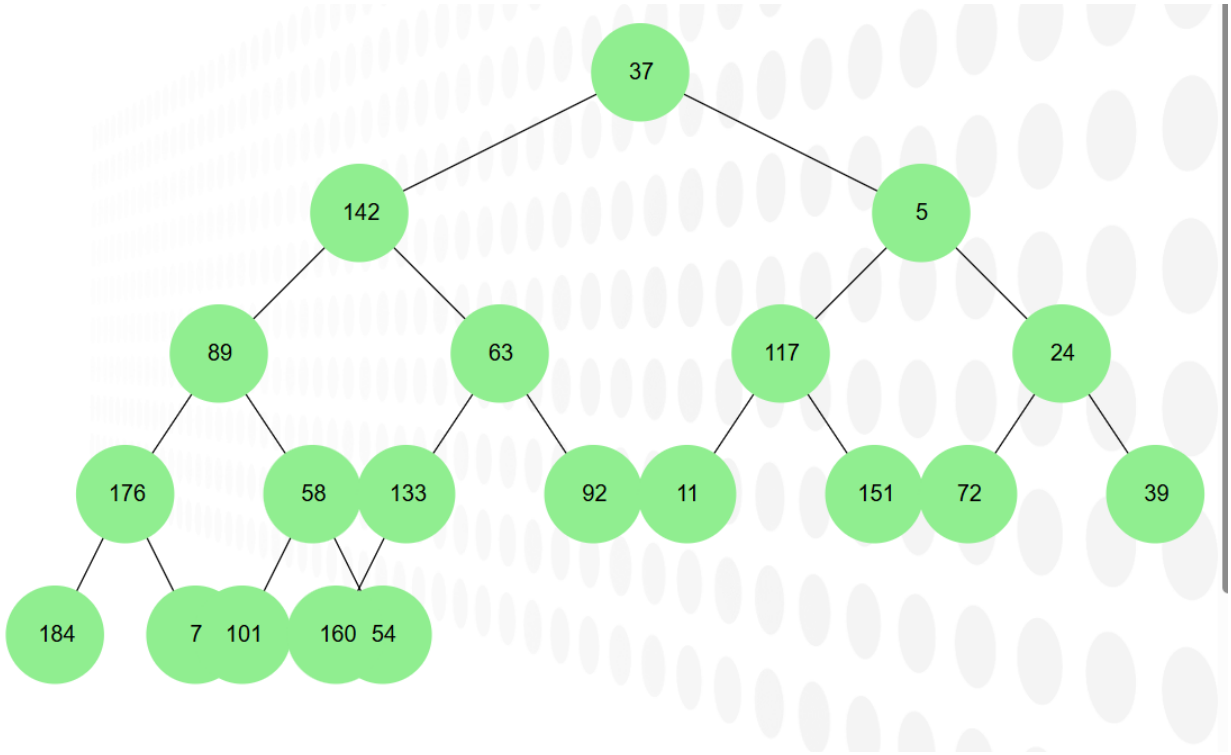
Construction / insertion description:

輸入時要把頓號去掉:

然後按 BINARY TREE VISUALIZATION 就會出來了

所有的點會按照輸入順序從左到右放入樹:

Screenshot of Binary Tree (paste below):



3.2 Complete Binary Tree

Tool name / URL: https://trees-visualizer.netlify.app/trees

Construction / insertion description:

Student ID: Student Name:

上面的 Binary tree 就算是 complete binary tree 了， AI 沒有推薦我其他選項，我也覺得沒有必要再找其他特別標註是用” complete binary tree”建樹的網站來用，這邊就不貼重複的截圖了。

Screenshot of Complete Binary Tree (paste below):

同上

3.3 Binary Search Tree (BST) Tool

name / URL:

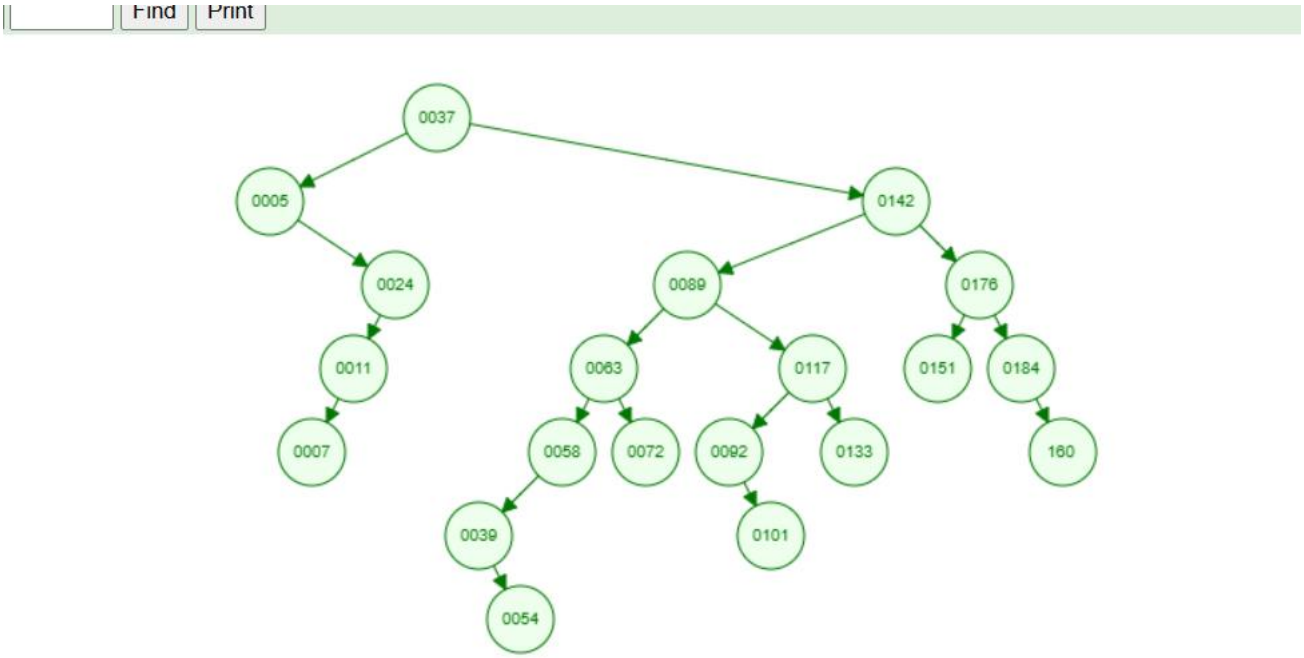
https://www.cs.usfca.edu/~galles/visualization/BST.html

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

用給定的順序輸入(37 142 5 89 63 117 24 176 58 133 92 11 151 72 39 184 7 101 54 160)

一次只能輸入一個數字，然後可以看判斷過程(大於往右放，小於往左放)

Screenshot of BST (paste below):



3.4 AVL Tree Tool

name / URL:

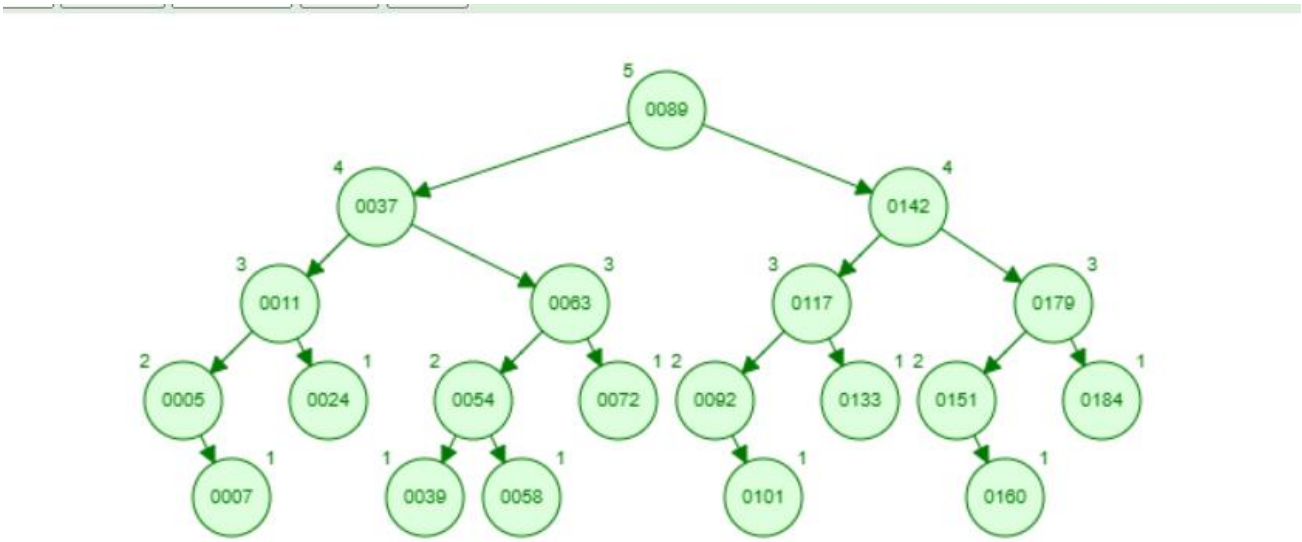
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Insertion & balancing description:

用給定的順序輸入，同樣每次只能輸入一個數字，每次加入一個點除了做 BST 判斷外，會額外記一個高度值再點的左上角，用來判斷是否需要調整位置，如果高度有需要調整，通常是移動最下面那個點與其父節點(這應該就是做旋轉)

Student ID: Student Name:

Screenshot of AVL Tree (paste below):



3.5 Red-Black Tree Tool

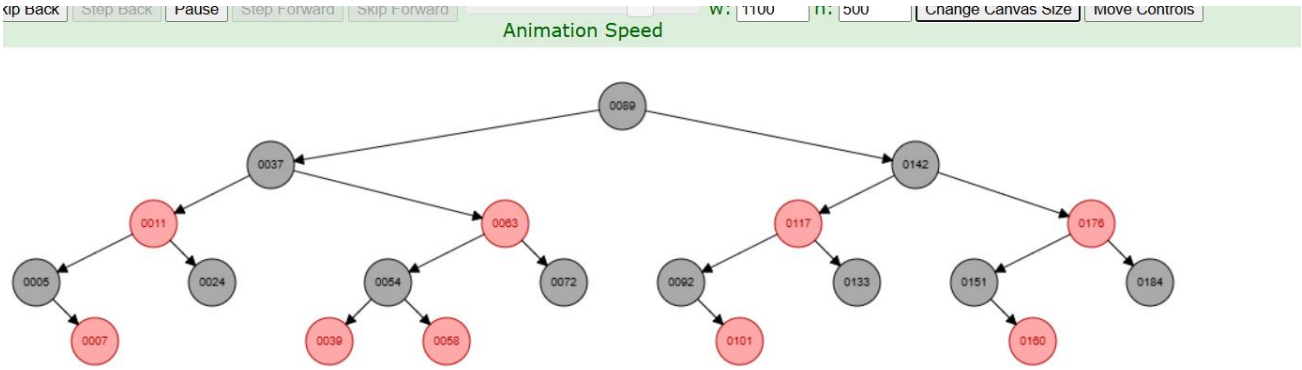
name / URL:

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

Insertion & balancing description:

用給定的順序輸入，同樣每次只能輸入一個數字，輸入時除了根以外初始都是紅色，先做 BST 放入後才開始判斷是否符合紅黑樹的規則，通常是做改變其與親代的顏色，也有做一個超大旋轉導致根換成其他值的事件發生。

Screenshot of Red-Black Tree (paste below):



Student ID: Student Name:

3.6 Max Heap Tool

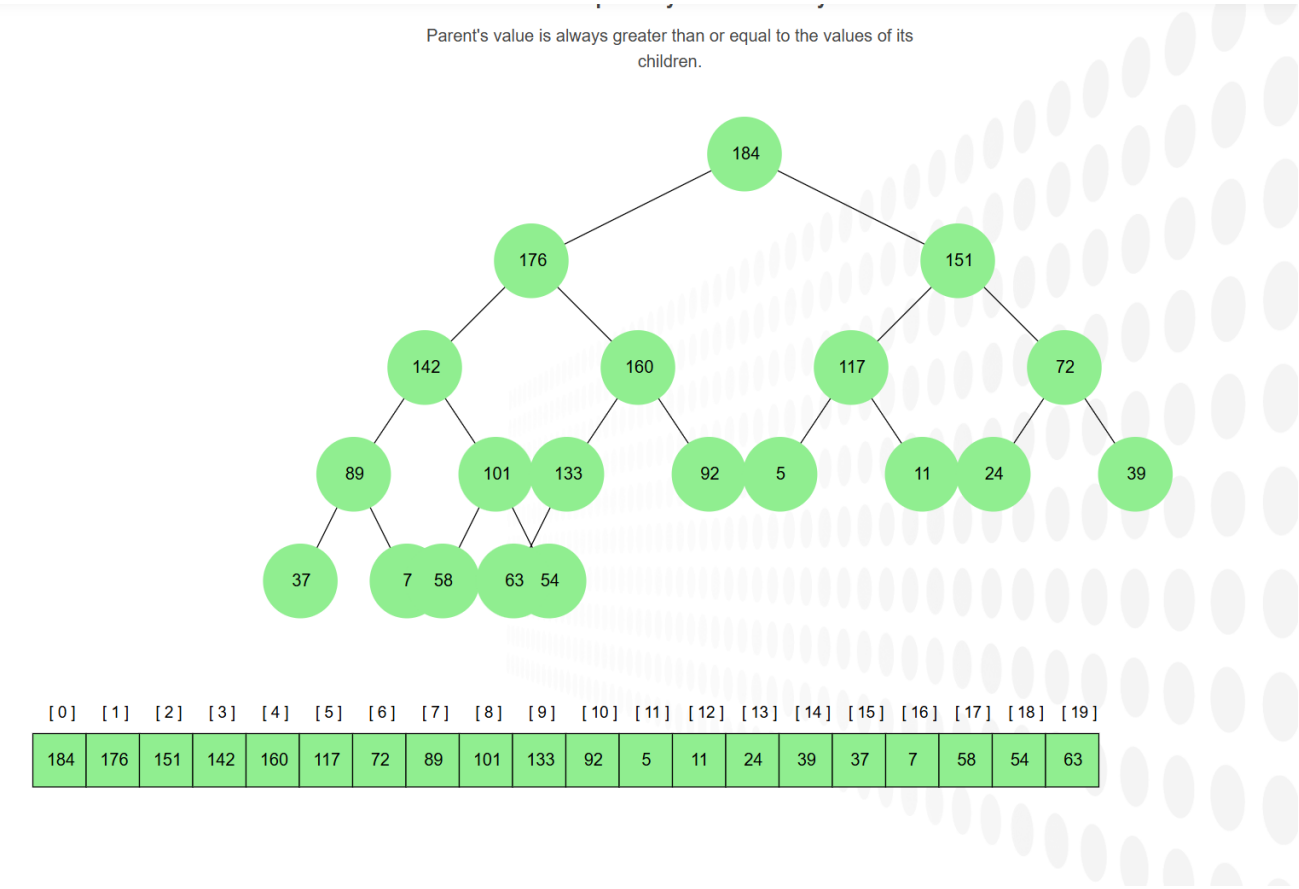
name / URL:

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

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

這個是直接輸入一整串數字的，我也有嘗試把數字一個一個分開往後加入，太麻煩了做了 5 個就放棄了，雖然這個網站沒有過程，不過可以合理推測大概和 min heap 一樣，只是改成判斷大於。

Screenshot of Max Heap (paste below):



3.7 Min Heap

Tool name / URL:

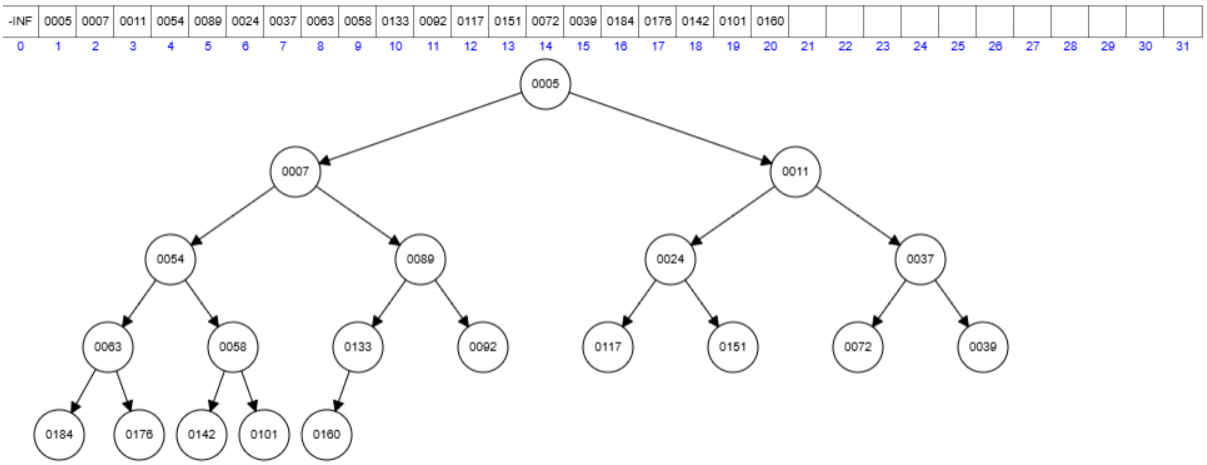
https://www.cs.usfca.edu/~galles/visualization/Heap.html

Construction / heap-building description:

每次輸入時先直接放入樹裡，再判斷是否小於父節點，並依次做交換。

Screenshot of Min Heap (paste below):

Student ID: Student Name:



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	運算式樹	Leaf 是數字，其餘是運算子的話，可以透過 pre-order 遍歷樹輸出運算式
Complete Binary Tree	Heap 實作	因為除了最後一層都可以完全填滿的特性，可以當作 array 儲存值(或是使用 array 當作樹，減省存 pointer 的空間)。
Binary Search Tree	電話簿或 ID 的查找系統	因為儲存特性，查找資料的平均時間為 O(logn) ，比一個一個比對的 O(n) 快
AVL Tree	資料變動少的系統	樹高會因為特性所以比較矮，查詢時間能比紅黑樹更快，但因為高度嚴格限制，插入或刪除時整理樹的時間比較久，不適合多次變動。
Red-Black Tree	Map、Set	因為特性關係，高度限制比 AVL 寬鬆，適合多讀寫刪除的資料結構
Max Heap	工作排程	把重要的工作排更大的數字就可以用 O(1)抓到優先權最高的資料
Min Heap	Dijkstra(最短路徑)	找最短路徑時先找目前距離最小的那個點，用 min 可以達成 O(1)

Student ID: Student Name:

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，因為 AVL 的高度理論上最矮，在搜尋上會比其他的還要快。

Frequent insertions and deletions? Why?

Red-Black tree，他觸發節點變動(旋轉)的可能比較小，因為在整理樹時還有一個"只變色"的方案作為緩衝，可以在整理樹時減少很多時間。

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，首先比起 tree 他少了 pointer 要儲存，而且搜尋也可以達到 O(logn)，因為不會再變動也不用擔心重新 sort 需要的 O(n^2)。

Section 6. AI Usage Log (Required)

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

Index	Date / Time	AI Service (ChatGPT, Gemini, etc.)	Your Full Prompt / QuesOon
0	12/28 23:38	Gemini	作業 PDF
1	12/29 23:38	Gemini	請告訴我這份作業想要我做甚麼
2	12/29 00:00	Gemini	好 請先給我 1 和 2 的資料
3	12/29 00:35	Gemini	一般樹 (General Tree) 和 graph 的差異在哪?
4	12/30 23:26	Gemini	Max Heap 有 Complete Binary Tree 的特徵?
5	12/30 23:28	Gemini	請推薦我第三大題可以用的網站
6	12/30 00:55	Gemini	可以給我第四題的一些例子嗎
7	12/30 01:42	Gemini	AVL 樹為什麼適合資料變動少的? 意思是他不適合多次插入與刪除嗎?
8	12/30 01:46	Gemini	為什麼 min heap 適合最短路徑?
9	12/30 01:49	Gemini	請幫我將 section 5 的題目翻譯成中文
10	12/30 01:54	Gemini	為什麼紅黑樹觸發旋轉比較少?

You may extend this table as needed.