Student ID: Student Name: 李奕翔

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: Student Name: 李奕翔

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: Student Name: 李奕翔

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.

1. General Tree

   Definition:最基本的樹狀結構，節點可以有任意數量的子節點，且沒有特定順序限制 。

Student ID: Student Name: 李奕翔

2. Binary Tree

Definition:

一種特殊的樹，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，且最後一層的所有節點都盡可能地**靠左排列** 。

4. Binary Search Tree (BST)

Definition:一種有順序的二元樹，滿足：左子樹所有節點值 < 根節點值 < 右子樹所有節點值 。

5. AVL Tree

Definition:

一種**自我平衡**的 BST，任何節點的左右子樹高度差（平衡因子）最多為 1

6. Red-Black Tree

Definition:

另一種自我平衡 BST，透過節點顏色（紅或黑）與特定規則確保樹的高度大約維持在 O(logn) ，平衡要求比 AVL 寬鬆，但適合頻繁插入刪除 。

7. Max Heap

Definition:

一種完全二元樹，父節點的值永遠**大於或等於**其子節點的值 。

8. Min Heap

Definition:

種完全二元樹，父節點的值永遠**小於或等於**其子節點的值 。

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):

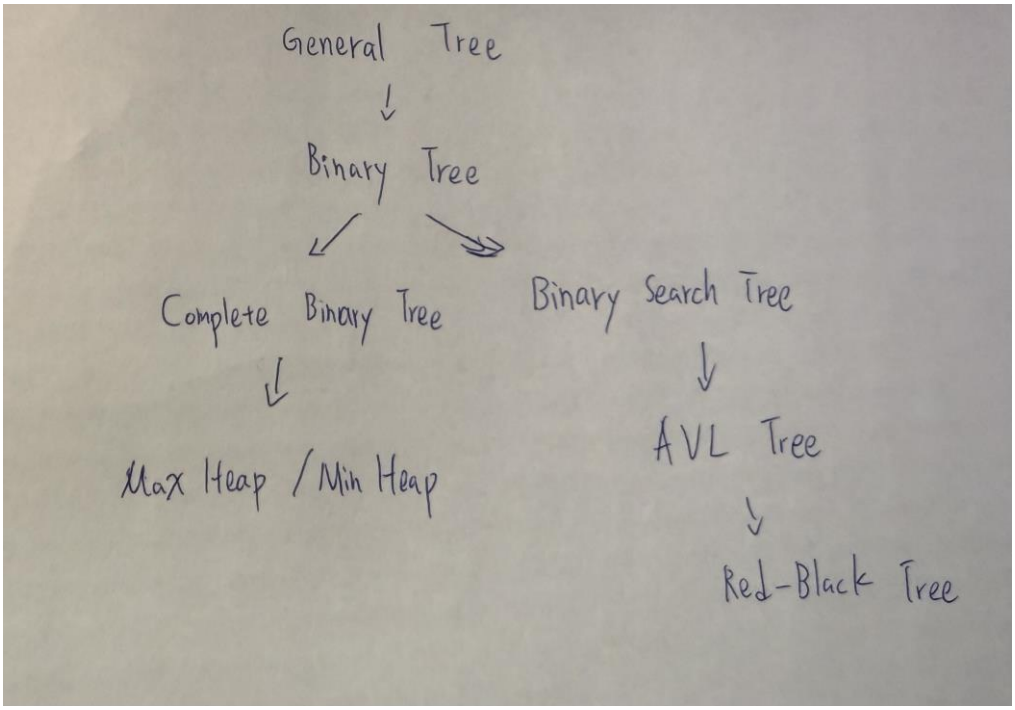
General Tree → Binary Tree → Complete Binary Tree

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

Binary Tree → Max Heap / Min Heap

Your Diagram:

Student ID: Student Name: 李奕翔



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	節點必須按層填滿，且最後一層需靠左對齊。
Binary Tree	Binary Search Tree	增加順序規則：左子節點 < 父節點 < 右子節點 。
BST	AVL Tree	增加高度平衡限制：左右子樹高度差不可超過 1 。
BST	Red-Black Tree	增加顏色屬性與特定旋轉規則，確保概略平衡。
Binary Tree	Max Heap	限制必須是完全二元樹，且父節點大於等於子節點
Binary Tree	Min Heap	限制必須是完全二元樹，且父節點小於等於子節點

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: Student Name: 李奕翔

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:

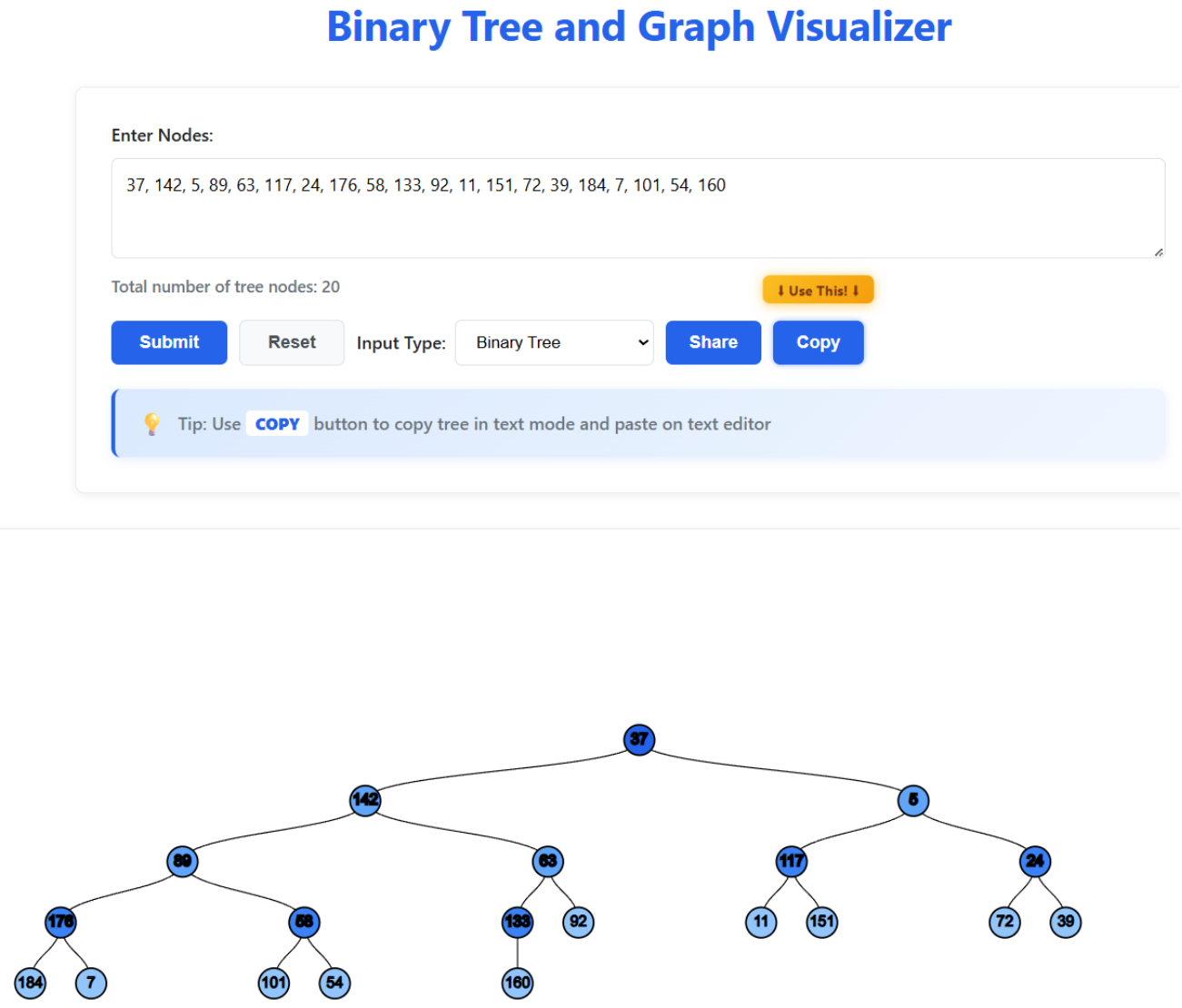
binary tree visualizer and converter /

Binary Tree Visualizer and Converter

Construction / insertion description:

依照給定的整數序列順序插入。

Screenshot of Binary Tree (paste below):



3.2 Complete Binary Tree

Tool name / URL:

graphviz online /

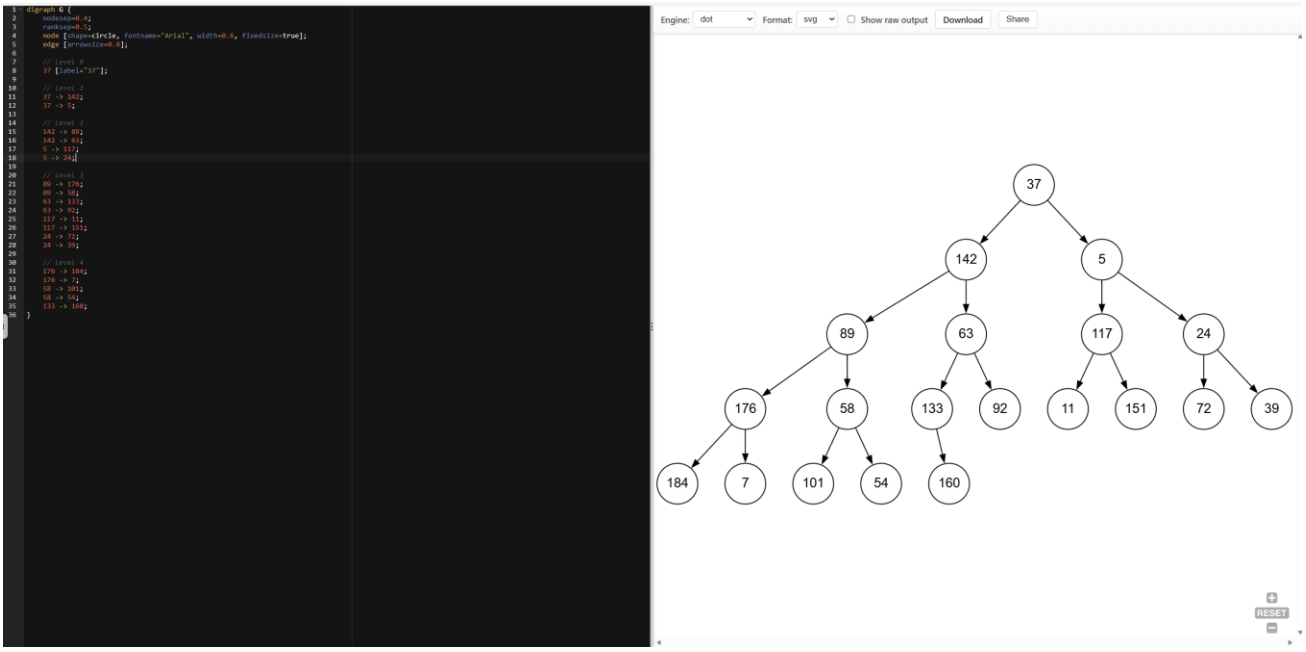
Student ID: Student Name: 李奕翔

Data Structure Visualization

Construction / insertion description:

沒看到有 complete binary tree visualizations 我用 graphviz online 然後按照 LEVEL 輸入

Screenshot of Complete Binary Tree (paste below):



3.3 Binary Search Tree (BST)

Tool name / URL:

University of San Francisco Data Structure Visualization/

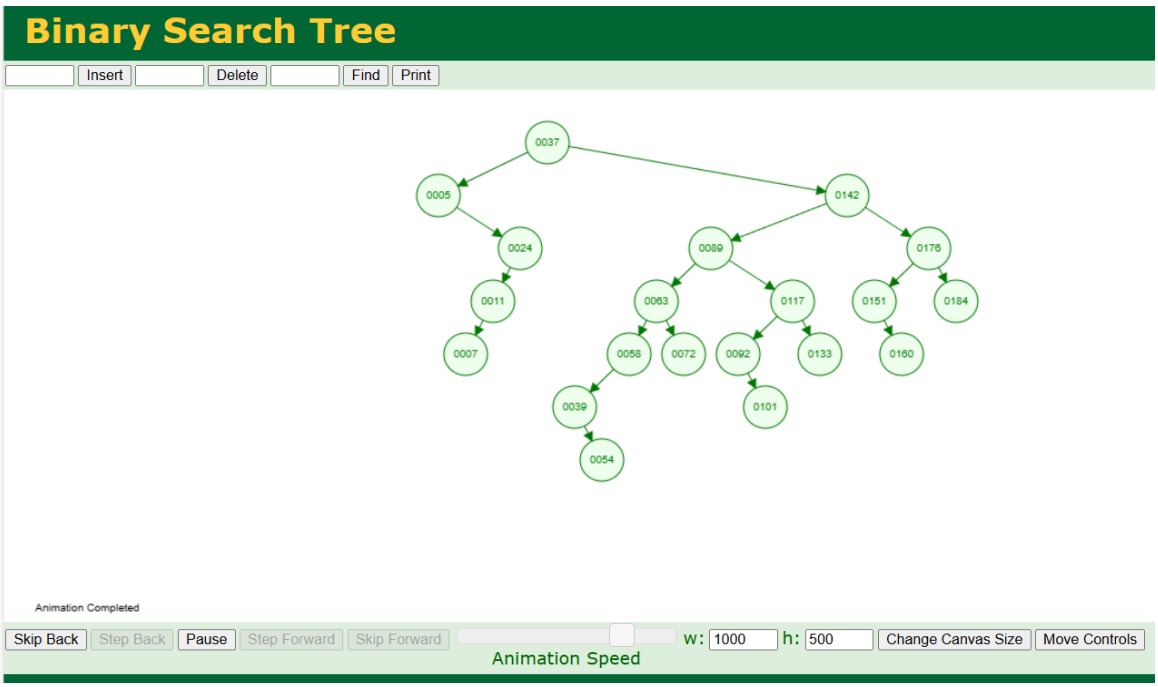
Data Structure Visualization

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

依照給定的整數序列順序插入。

Screenshot of BST (paste below):

Student ID: Student Name: 李奕翔



3.4 AVL Tree

Tool name / URL:

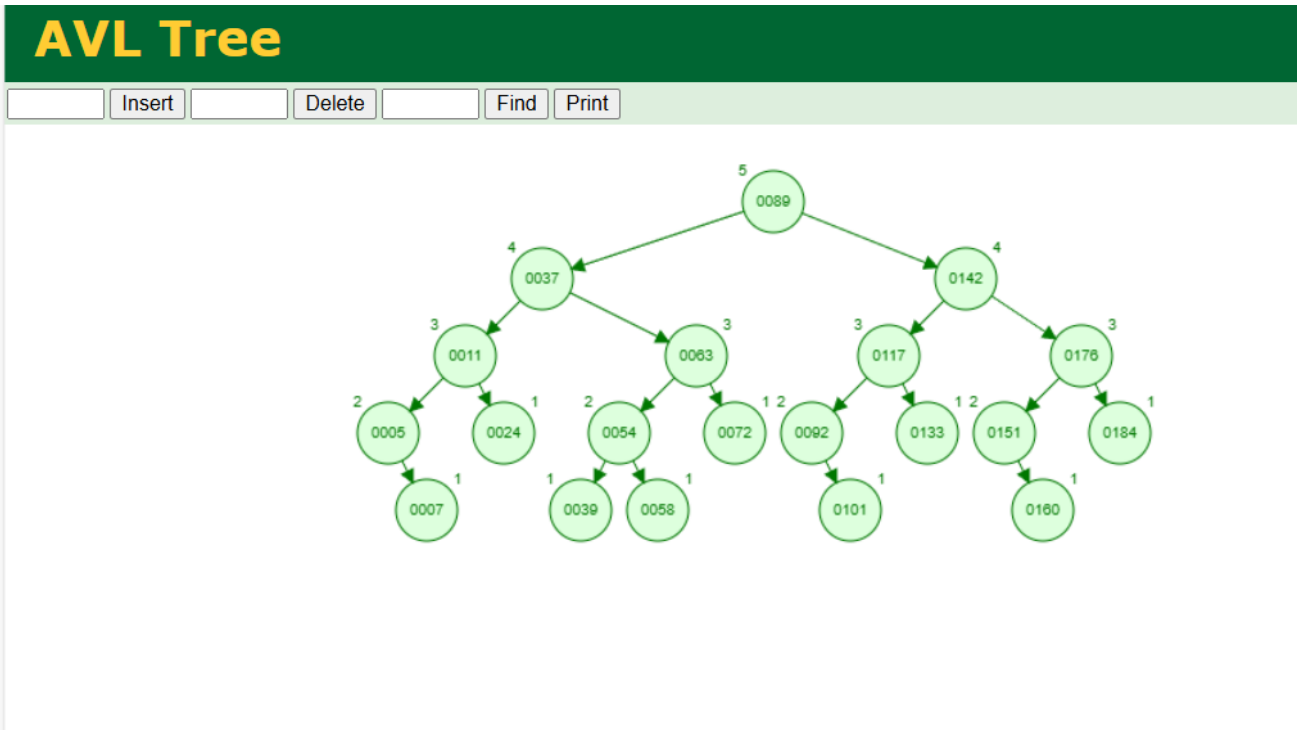
University of San Francisco Data Structure Visualization/

[Data Structure Visualization]

Insertion & balancing description:

依照給定的整數序列順序插入。

Screenshot of AVL Tree (paste below):



Student ID: Student Name: 李奕翔

3.5 Red-Black Tree

Tool name / URL:

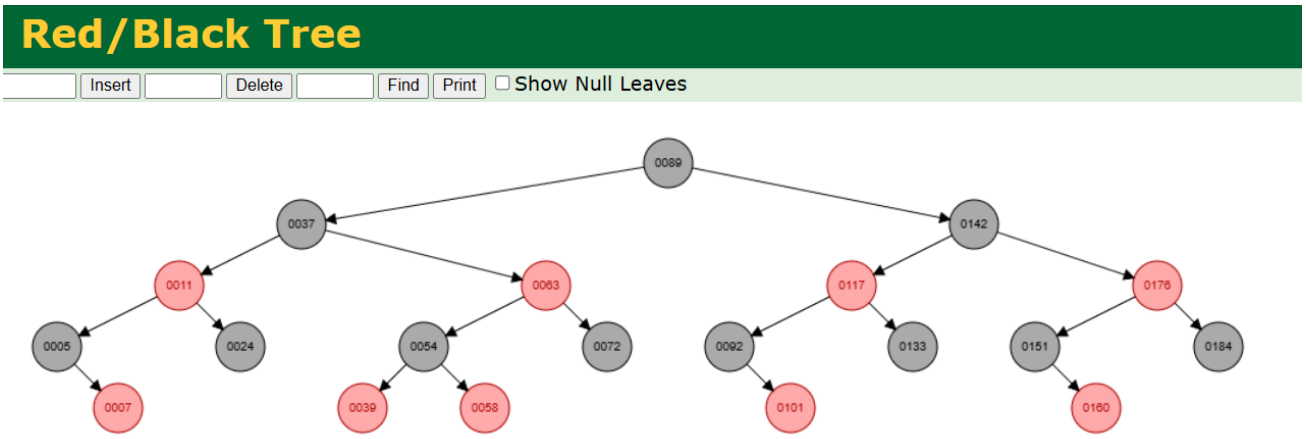
University of San Francisco Data Structure Visualization/

Data Structure Visualization

Insertion & balancing description:

依照給定的整數序列順序插入。

Screenshot of Red-Black Tree (paste below):



3.6 Max Heap

Tool name / URL:

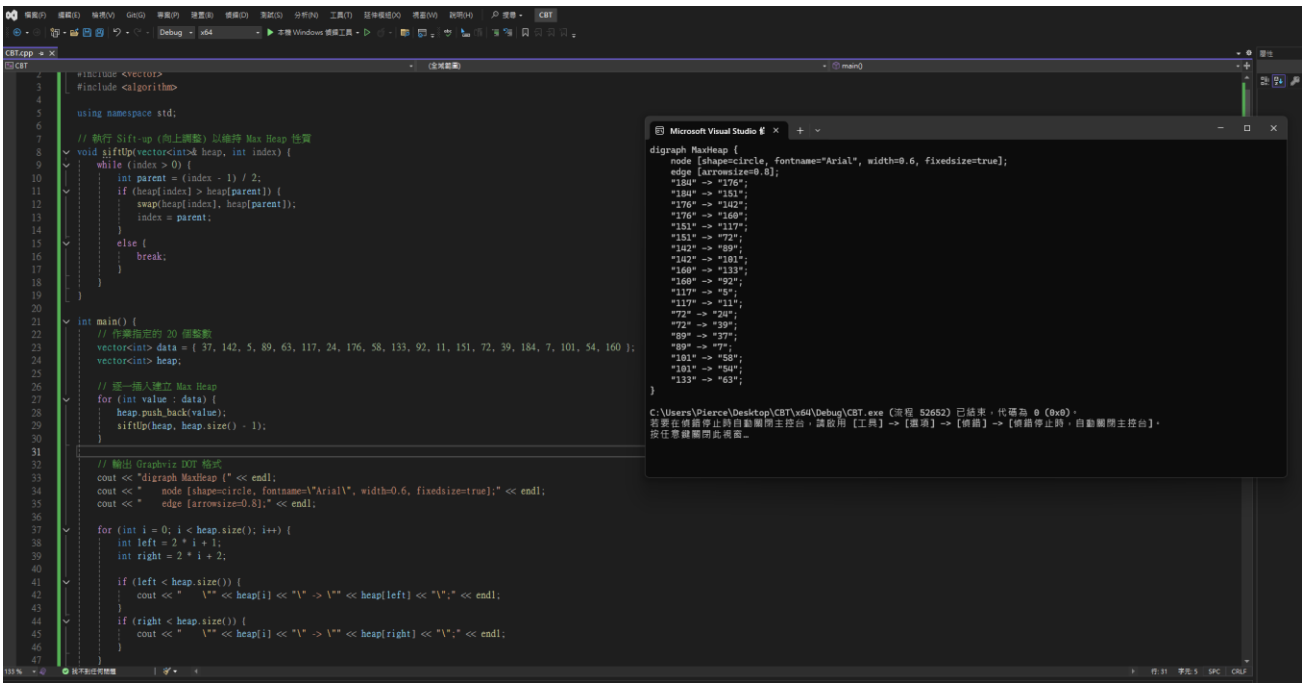
University of San Francisco Data Structure Visualization/

Data Structure Visualization

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

沒看到有 complete binary tree visualizations 我用 C++寫出 DIAGRAM 再丟入 graphviz online 生成圖片

Student ID: Student Name: 李奕翔

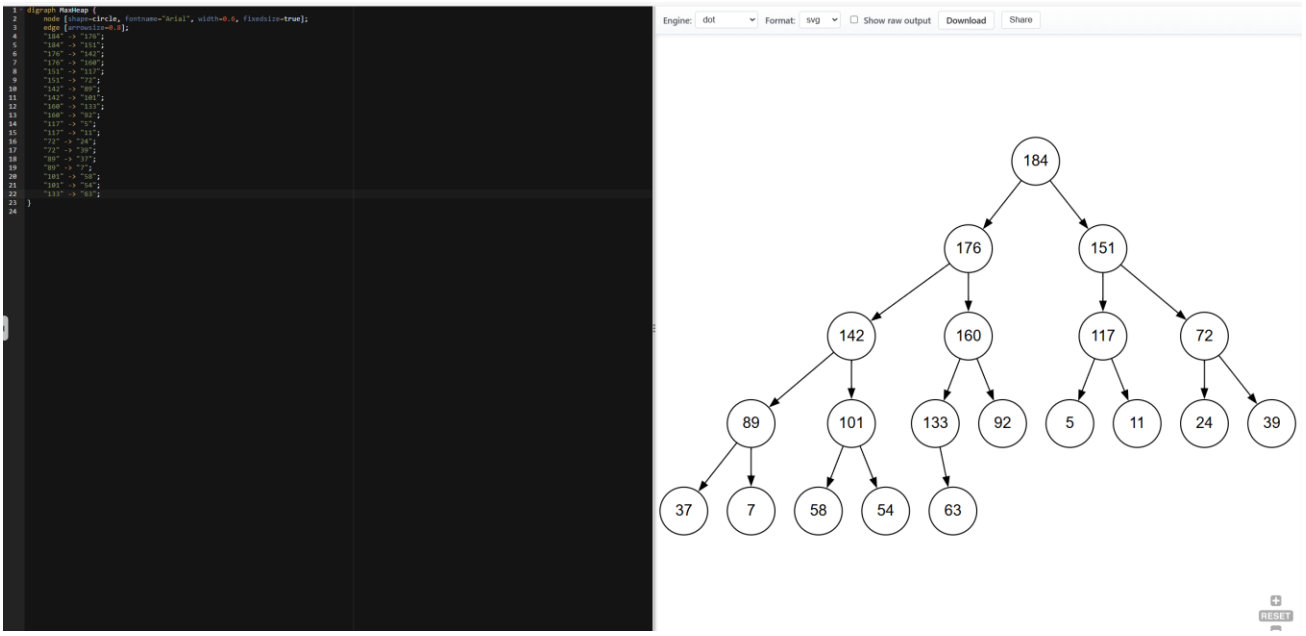


```
#include <vector>
#include <algorithm>

using namespace std;

// 執行 Sift-up (向上調整) 以維持 Max Heap 性質
void siftUp(vector<int>& heap, int index) {
    while (index > 0) {
        int parent = (index - 1) / 2;
        if (heap[index] > heap[parent]) {
            swap(heap[index], heap[parent]);
            index = parent;
        }
        else {
            break;
        }
    }
}

int main() {
    // 作業指定的 20 個整數
    vector<int> data = { 37, 142, 5, 89, 63, 117, 24, 176, 58, 133, 92, 11, 151, 72, 39, 184, 7, 101, 54, 160 };
    vector<int> heap;

    // 逐一插入建立 Max Heap
    for (int value : data) {
        heap.push_back(value);
        siftUp(heap, heap.size() - 1);
    }

    // 輸出 Graphviz DOT 格式
    cout << "digraph MaxHeap {" << endl;
    cout << "    node [shape=circle, fontname=\"Arial\", width=0.6, fixedsize=true];" << endl;
    cout << "    edge [arrowsize=0.8];" << endl;

    for (int i = 0; i < heap.size(); i++) {
        int left = 2 * i + 1;
        int right = 2 * i + 2;

        if (left < heap.size()) {
            cout << "    \"" << heap[i] << "\" -> \"" << heap[left] << "\";" << endl;
        }
        if (right < heap.size()) {
            cout << "    \"" << heap[i] << "\" -> \"" << heap[right] << "\";" << endl;
        }
    }
    cout << "}" << endl;

    return 0;
}
```

```
Microsoft Visual Studio
digraph MaxHeap {
    node [shape=circle, fontname="Arial", width=0.6, fixedsize=true];
    edge [arrowsize=0.8];
    "184" -> "176";
    "184" -> "151";
    "176" -> "142";
    "176" -> "160";
    "151" -> "117";
    "151" -> "72";
    "142" -> "89";
    "142" -> "101";
    "160" -> "133";
    "160" -> "92";
    "117" -> "5";
    "117" -> "11";
    "72" -> "24";
    "72" -> "39";
    "89" -> "37";
    "89" -> "7";
    "101" -> "58";
    "101" -> "54";
    "133" -> "63";
}

C:\Users\Pierce\Desktop\CBT\x64\Debug\CBT.exe (程序 52652) 已結束，代碼為 0 (0x0)。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...
```

依照給定的整數序列順序插入。

Screenshot of Max Heap (paste below):



3.7 Min Heap

Tool name / URL:

University of San Francisco Data Structure Visualization/

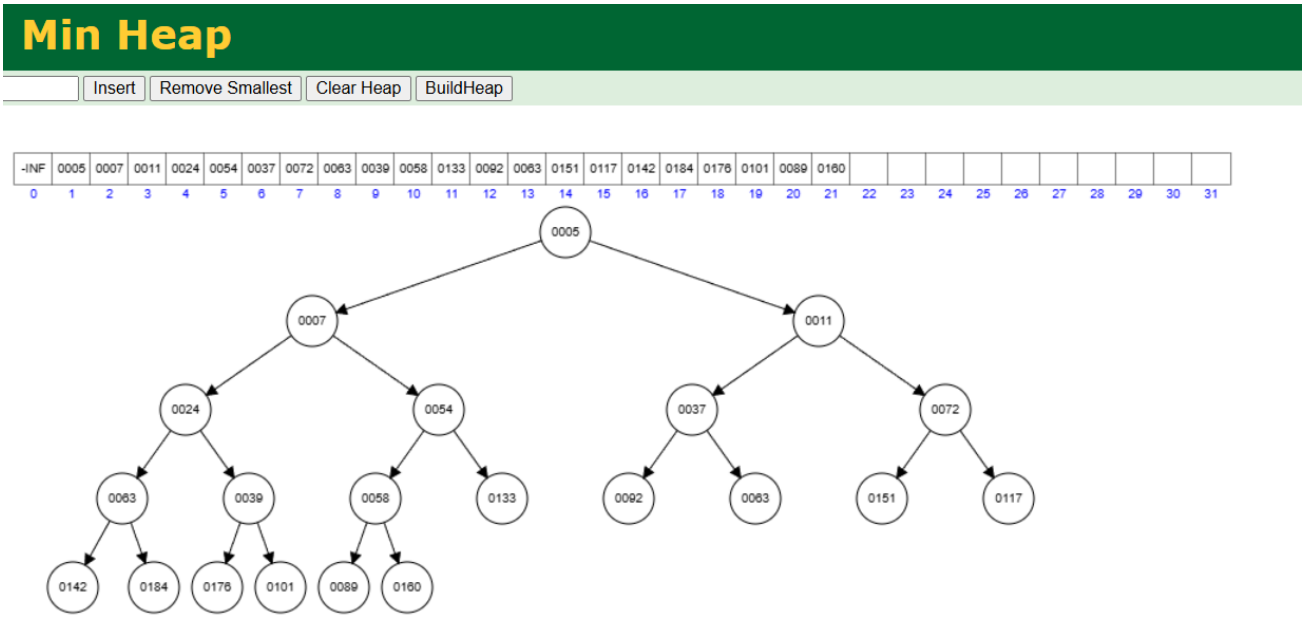
Data Structure Visualization

Construction / heap-building description:

Student ID: Student Name: 李奕翔

依照給定的整數序列順序插入。

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	Expression tree	It naturally represents arithmetic expressions
Complete Binary Tree	Heap implementation	Easy to store and manage using arrays
Binary Search Tree	Dictionary	Allows efficient searching of ordered data
AVL Tree	Database indexing	Strict balance improves search performance
Red-Black Tree	C++ map	Guarantees O(log n) operations even with updates
Max Heap	Priority scheduling	Fast access to the highest priority element
Min Heap	Task scheduling	Fast access to the smallest element

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?

If most operations are searches, I would choose an AVL tree because it maintains strict balance, which results in faster search performance.

Frequent insertions and deletions? Why?

Student ID: Student Name: 李奕翔

For frequent updates, a Red-Black tree is more suitable because it requires fewer rotations and handles insertions and deletions efficiently.
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?
If the data is static and no updates are needed, I would prefer a sorted array with binary search since it is simple and efficient.

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	Gemini	Gemini 幫我提供一般樹的定義，以及完整的二元樹、二元搜尋樹、AVL 樹和紅黑樹的定義。
2	12/29	Gemini	Help me organize the structure of my tree assignment report.
3	12/29	Gemini	3.2 Complete Binary Tree 3.6 Max Heap 我用 gemini 讓他幫我寫 code 讓我能生出 DIAGRAM 再丟入 graphviz online 生成圖片
4	12/29	Gemini	What is a binary search tree and how does insertion work step by step?