資料結構第四次作業

班級:資工-乙

姓名:侯秉辰

學號:4B2G0116

一・題目

使用者輸入不等數量的整數值，分別產生 binary search tree 和 max-heap 所產生的結果以陣列的方式與樹狀圖方式呈現出來

二・程式

// 4B2G0116.cpp： 此檔案包含 'main' 函式。程式會於該處開始執行及結束執行。

//

```
#include <iostream>
#include <vector>
#include <queue>
#include <iomanip>
using namespace std;

// Node structure for Binary Search Tree
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

// Functions for Binary Search Tree
TreeNode* insertBST(TreeNode* root, int val) {
    if (!root) return new TreeNode(val);
    if (val < root->val) root->left = insertBST(root->left, val);
    else root->right = insertBST(root->right, val);
    return root;
}

void printBST(TreeNode* root, int space = 0, int level_gap = 5) {
    if (!root) return;
    space += level_gap;
    printBST(root->right, space);
    cout << endl;
    cout << setw(space) << root->val << endl;
    printBST(root->left, space);
```

```
}

// Functions for Max-Heap
void insertHeap(vector<int>& heap, int val) {
	heap.push_back(val);
	int i = heap.size() - 1;
	while (i > 0 && heap[(i - 1) / 2] < heap[i]) {
		swap(heap[(i - 1) / 2], heap[i]);
		i = (i - 1) / 2;
	}
}

void printHeapTree(vector<int>& heap, int i = 0, int space = 0, int level_gap = 5) {
	if (i >= heap.size()) return;
	space += level_gap;
	printHeapTree(heap, 2 * i + 2, space);
	cout << endl;
	cout << setw(space) << heap[i] << endl;
	printHeapTree(heap, 2 * i + 1, space);
}

void printArray(const vector<int>& arr) {
	for (int val : arr) {
		cout << val << " ";
	}
	cout << endl;
}

int main() {
	TreeNode* bstRoot = nullptr;
	vector<int> maxHeap;

	cout << "請輸入一系列整數（以非數字結束輸入，例如輸入 'q'）：" << endl;
	int num;
	while (cin >> num) {
		bstRoot = insertBST(bstRoot, num);
		insertHeap(maxHeap, num);
	}
```

```
        cin.clear(); // Clear the error state caused by non-integer input
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        cout << "\nBinary Search Tree (陣列形式無法直接呈現，但提供樹狀圖表示):"
<< endl;
        printBST(bstRoot);

        cout << "\nMax-Heap (以陣列形式表示):" << endl;
        printArray(maxHeap);
        cout << "Max-Heap (以樹狀圖表示):" << endl;
        printHeapTree(maxHeap);

        return 0;
}

// 執行程式: Ctrl + F5 或 [偵錯] > [啟動但不偵錯] 功能表
// 偵錯程式: F5 或 [偵錯] > [啟動偵錯] 功能表

// 開始使用的提示:
//   1. 使用 [方案總管] 視窗，新增/管理檔案
//   2. 使用 [Team Explorer] 視窗，連線到原始檔控制
//   3. 使用 [輸出] 視窗，參閱組建輸出與其他訊息
//   4. 使用 [錯誤清單] 視窗，檢視錯誤
//   5. 前往 [專案] > [新增項目]，建立新的程式碼檔案，或是前往 [專案] > [新
增現有項目]，將現有程式碼檔案新增至專案
//   6. 之後要再次開啟此專案時，請前往 [檔案] > [開啟] > [專案]，然後選
取 .sln 檔案
```

三．程式說明

Binary Search Tree (BST):

- 使用 TreeNode 結構表示樹節點。
- 輸入的每個數字插入到 BST 中。
- 使用遞迴函數顯示 BST 的樹狀結構。

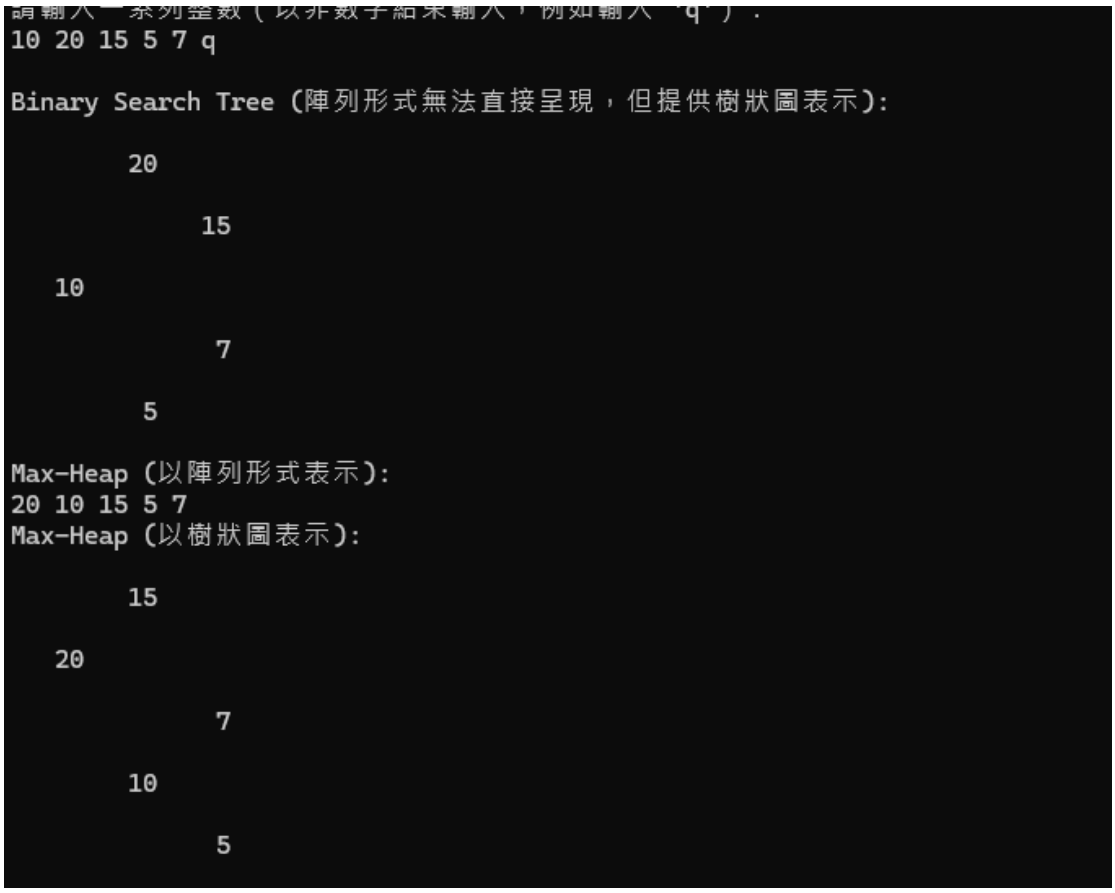
Max-Heap:

- 使用動態陣列表示 Heap。
- 每插入一個數字，都通過調整堆的結構維持 Max-Heap 性質。
- 提供以陣列和樹狀圖的方式顯示 Heap。

輸入與結束條件：

- 使用者可輸入一系列整數，並以非數字（例如 q）作為結束標誌。

四．執行結果



https://github.com/4b2g0116/4B2G0116-