DS HW8

Deadline(107/12/11)

作業注意事項:

1. 請同學 HW8 與 HW9 分開寫，繳交作業時需分開繳交(會有兩個箱子分別給同學繳交 HW8&9)。
2. 助教會在 12/11(二)的 13:20~13:30 到教室收作業，如果同學無法在這段期間繳交，可以在當天 17:30 前交到 ED817 lab 外面的箱子。

手寫題

2. Balance the AVL tree in Figure 8-18. Show the balance factors in the result.

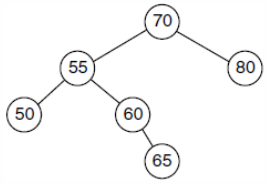

FIGURE 8-18 Figure for Exercise 2

4. Add 68 to the AVL tree in Figure 8-19. The result must be an AVL tree. Show the balance factors in the resulting tree.

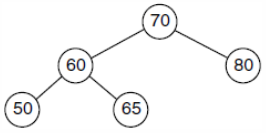

FIGURE 8-19 Figure for Exercises 3 and 4

6. Create an AVL tree using the following data entered as a sequential set. Show the balance factors in the resulting tree:

```
14 23 7 10 33 56 80 66 70
```

8. Create an AVL tree using the following data entered as a sequential set. Show the balance factors in the resulting tree:

```
80 70 66 56 33 23 14 10 7
```

10. Insert 44 and 50 into the tree created in Exercise 7.

(第 7 題僅供參考，不用寫)

7. Create an AVL tree using the following data entered as a sequential set. Show the balance factors in the resulting tree:

```
7 10 14 23 33 56 66 70 80
```

14. Write an iterative version of Algorithm 8-1, "AVL Tree Insert."

ALGORITHM 8-1 AVL Tree Insert

```
Algorithm AVLInsert (root, newData)
Using recursion, insert a node into an AVL tree.
   Pre    root is pointer to first node in AVL tree/subtree
          newData is pointer to new node to be inserted
   Post   new node has been inserted
   Return root returned recursively up the tree
 1 if (subtree empty)
   Insert at root
   1  insert newData at root
   2  return root
 2 end if
 3 if (newData < root)
   1  AVLInsert (left subtree, newData)
   2  if (left subtree taller)
      1  leftBalance (root)
   3  end if
 4 else
   New data >= root data
   1  AVLInsert (right subtree, newPtr)
   2  if(right subtree taller)
      1  rightBalance (root)
   3  end if
 5 end if
 6 return root
end AVLInsert
```

DS HW9

Deadline(107/12/11)

手寫題

2. Make a heap out of the following data read from the keyboard:

```
23 7 92 6 12 14 40 44 20 21
```

6. Apply the delete operation to the heap in Figure 9-21. Repair the heap after the deletion.

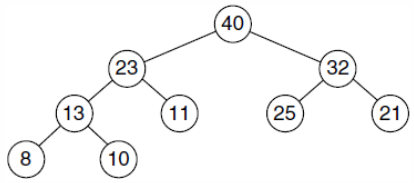

FIGURE 9-21 Heap for Exercises 5, 6, and 7

12. Which of the following sequences are heaps?
    a. 42 35 37 20 14 18 7 10
    b. 42 35 18 20 14 30 10
    c. 20 20 20 20 20 20

14. Show the resulting heap after 33, 22, and 8 are added to the following heap:

```
50 30 40 20 10 25 35 10 5
```

程式題

30. Modify Project 29 to determine the efficiency of the reheap up and reheap down algorithms only. Again, analyze the data and prepare a short report of your conclusions regarding their efficiency.

(29 題僅供參考，不用寫)

29. Our study of tree algorithmics has shown that most tree structures are quite efficient. Let's examine the efficiency of heaps. Modify the heap ADT developed in Section 9.3 to determine the complexity of building a heap. For this program measure efficiency as the number of data moves necessary to build the heap.

    To determine a pattern, run your program with arrays filled with random numbers. Use five different array sizes: 100, 200, 500, 1000, and 2000. Then analyze the heuristics developed in these runs and determine which big-O notation best applies. Prepare a short report of your findings with appropriate tables and graphs.