NTUST, CSIE
Algorithms (CS3001301), Spring 2022

Homework 1

Due date: Mar. 28
Total: 11 pts

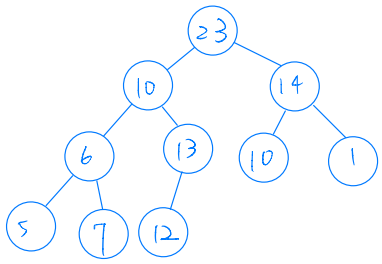
Problem 1.1. (2pts) Is the sequence ⟨23, 10, 14, 6, 13, 10, 1, 5, 7, 12⟩ a max-heap? Can you call MAX-HEAPIFY (*A*, *i*) to make it a max-heap for a single *i*? Why? How to make the sequence to become a heap if your answer to the previous question is no?

① The sequence is not a max-heap.

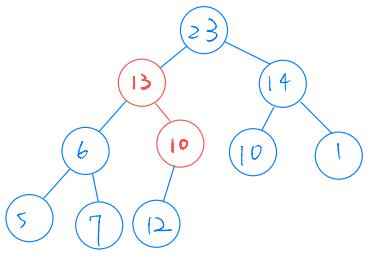
② No.

③ 需做2次 max-heapify

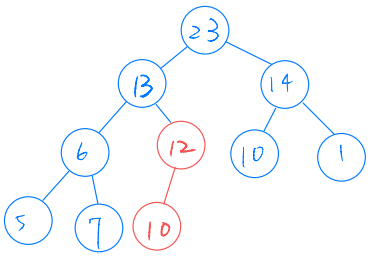
原始:



1次 HEAPIFY:



2次 HEAPIFY:



Problem 1.2. What is the effect of calling MAX-HEAPIFY (*A*, *i*) for *i* > heap-size[*A*]/2?

∵ leaf node 數量 = $\lceil heap\text{-}size[A]/2 \rceil$

∴ 節點 i 一定是 leaf node

又 leaf node 已是 max-heap (leaf node 為無子 node 的 node)

因此只需要看是否需要交換 node, 沒影響

Problem 1.3. (2pts) The operation HEAP-DELETE(A, i) deletes the item in node i from heap A. Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an n-element max-heap.

題目未提及是哪種 heap，因此任意刪除一個 node，
不能直接用最後 node 替換（可能會 increase key），
此時要用 HEAP-INCREASE-KEY 解決，
時間複雜度：$O(\lg n)$

Problem 1.4. (3pts) Given three sorting algorithms: (i) insertion sort, (ii) selection sort, (iii) bubble sort, can we use any of them to find the largest three elements from a sequence? How? What is the complexity? You should provide (the portion of) the codes if you want to make your conclusion clear.

insertion, selection, bubble 都能找到前三大元素，
因為此三種演算法只要排序出來看前三個就行

(i) insertion sort
complexity：$O(n^2)$

```
void insertionSort(int seq[],int n) {
    int i,j,tem;
    //從index[1]開始把元素加到已排序的序列中
    for (i = 0; i < n; i++) {
        tem = seq[i];//暫存
        //從已排序的陣列最後往前找新元素要加入的位置
        //把比新元素大的往後移
        //沒找到比新元素小的就把新元素放到最前面
        for (j = i - 1; (j >= 0 && tem < seq[j]); j--) {
            seq[j + 1] = seq[j];
        }
        seq[j + 1] = tem;
    }
}
```

(ii) selection sort

complexity : $O(n^2)$

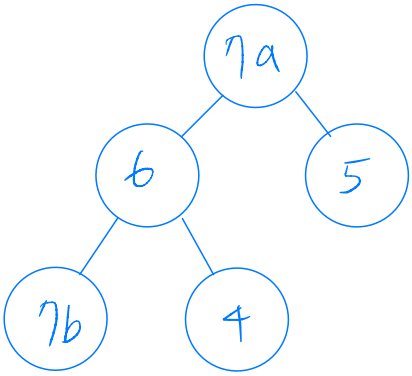
```
void selectionSort(int seq[], int n) {
    int loc, min;
    //由小到大選出前n-1個最小的數字,由前往後放到正確的位置
    for (int i = 0; i < n - 1; i++) {
        min = seq[i];
        loc = i;
        //找出seq[i]~seq[n-1]中的最小數存到min,再把位置暫存到loc
        for (int j = i + 1; j < n; j++) {
            if (seq[j] < min) {
                min = seq[j];
                loc = j;
            }
        }
        //交換
        seq[loc] = seq[i];
        seq[i] = min;
    }
}
```

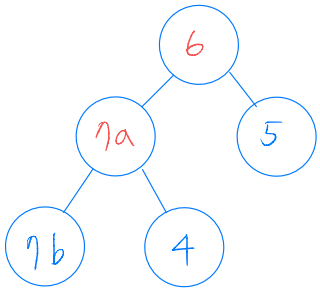
(iii) bubble sort

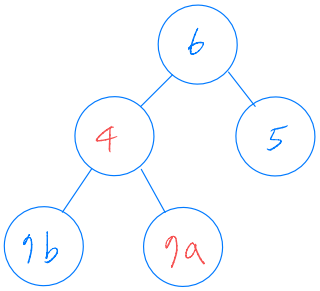
complexity : $O(n^2)$

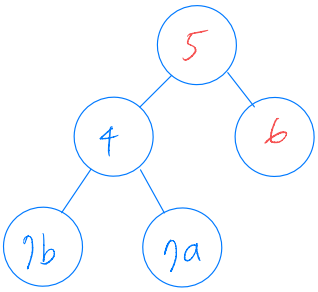
```
void bubbleSort(int seq[], int n) {
    bool flag = false;//紀錄是否曾經交換過
    int tem;
    //把index0~n-i-1個element兩兩比較,大的放到n-i-1
    for (int i = 0; i < n - 1; i++) {//最多掃n-1次
        flag = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (seq[j] > seq[j + 1]) {
                tem = seq[j];
                seq[j] = seq[j + 1];
                seq[j + 1] = tem;
                flag = true;//有交換
            }
        }
        if (!flag) {
            break;
        }
    }
}
```

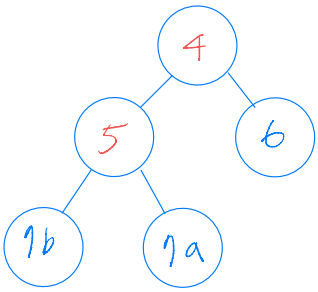
Problem 1.5. (3pts) Name the algorithms that can be implemented in a *stable* way for the given candidates: (i) insertion sort, (ii) merge sort, (iii) selection sort, (iv) bubble sort, (v) heapsort, (vi) quicksort. In your answer (Yes or No) for each of the algorithms, you should provide one or two lines of codes that is the key to make sure the stable property can (or cannot) be ensured.


7a
6
5
7b
4


6
7a
5
7b
4


6
4
5
7b
7a


5
4
6
7b
7a


4
5
6
7b
7a