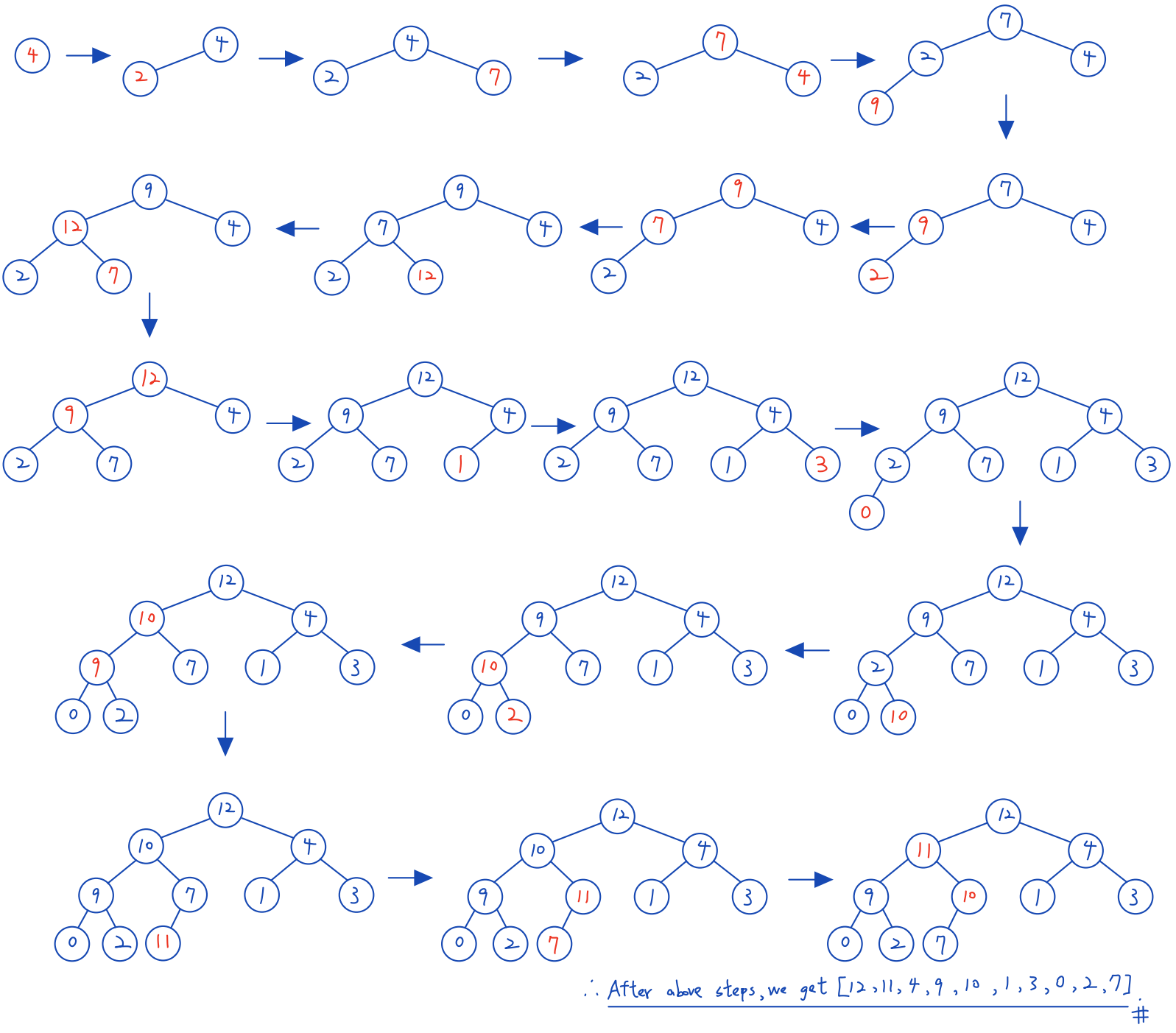
Module 7 Problem Set Hsuan-You(Shaun) Lin

1. My Answer: O(n log(n))
My reason:

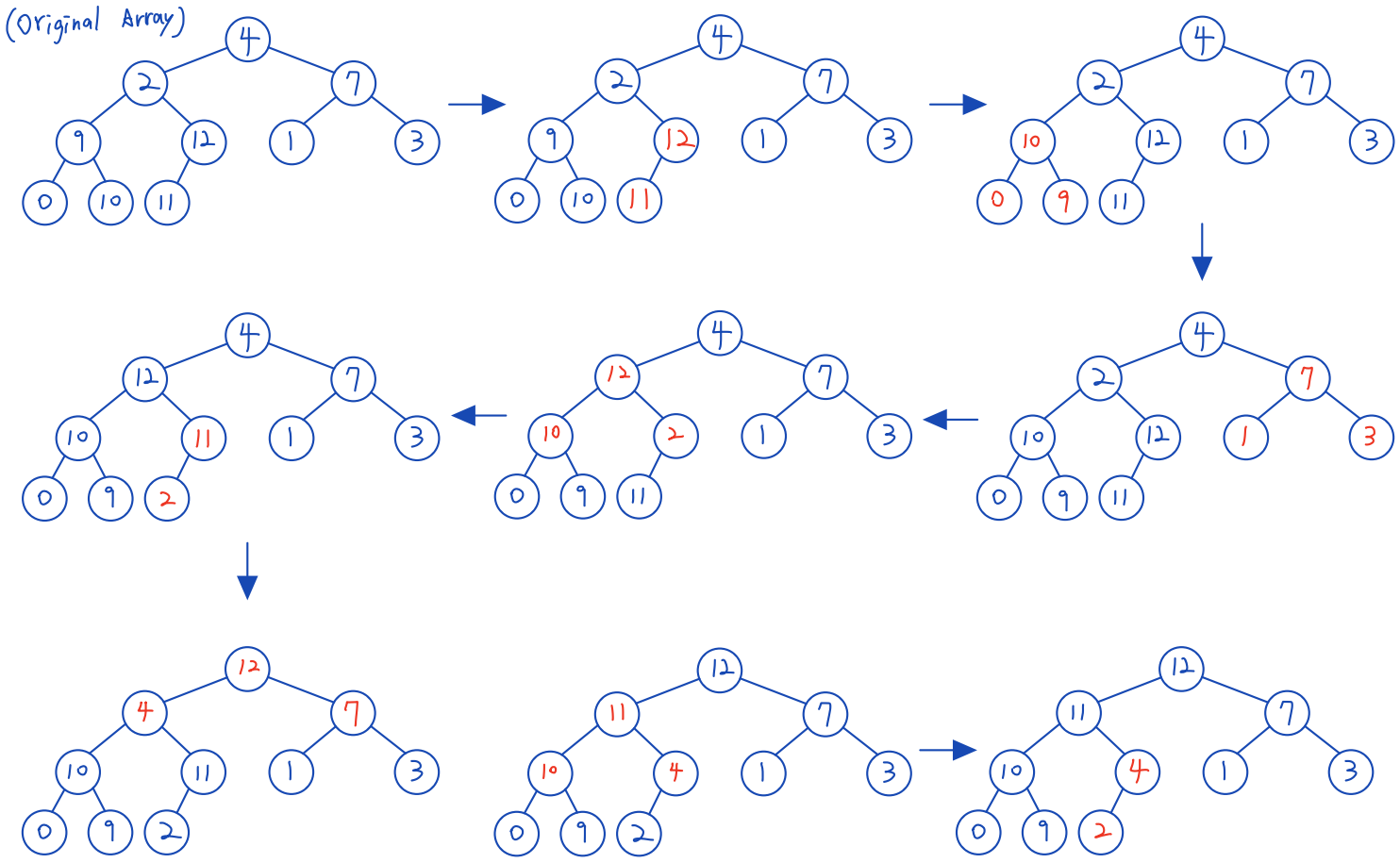
In the "build_heap" algorithm, we iterate through all element in the array and implement the "heap_insert" algorithm each time, every "heap_insert" operation will have complexity of O(log(n)), so we can get upper bound overall complexity of O(n) * O(log(n)) = O(n log(n)).

---

2. My Answer: [12, 11, 4, 9, 10, 1, 3, 0, 2, 7]
My steps:



3. My Answer: [12, 11, 7, 10, 4, 1, 3, 0, 9, 2]
My steps:

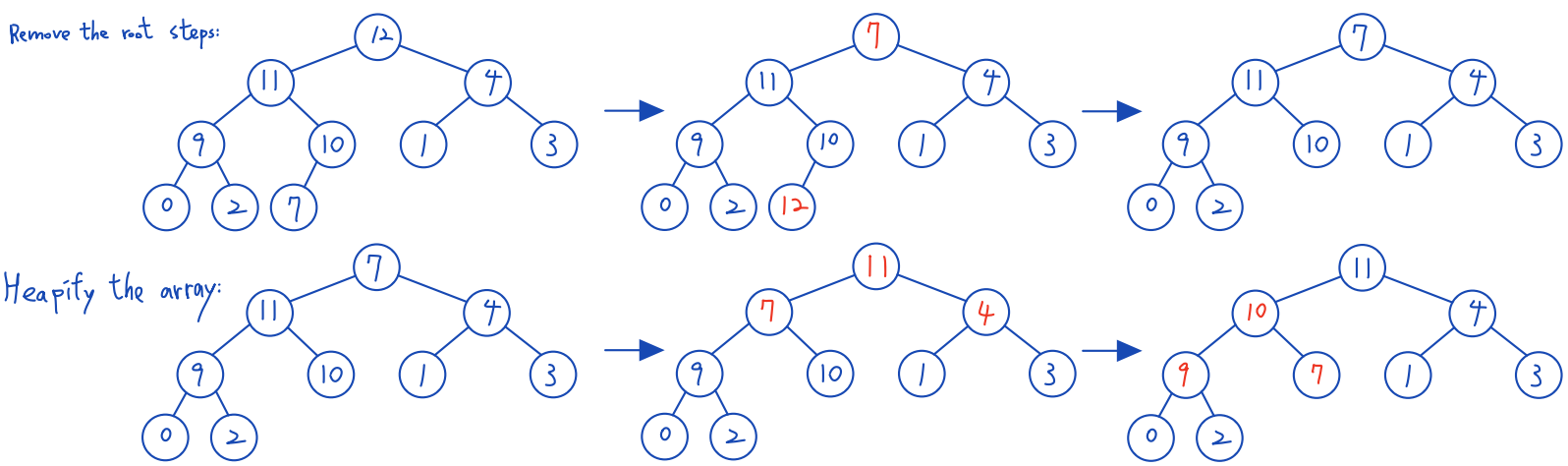


∴ After above steps, we get [12, 11, 7, 10, 4, 1, 3, 0, 9, 2]. #

---

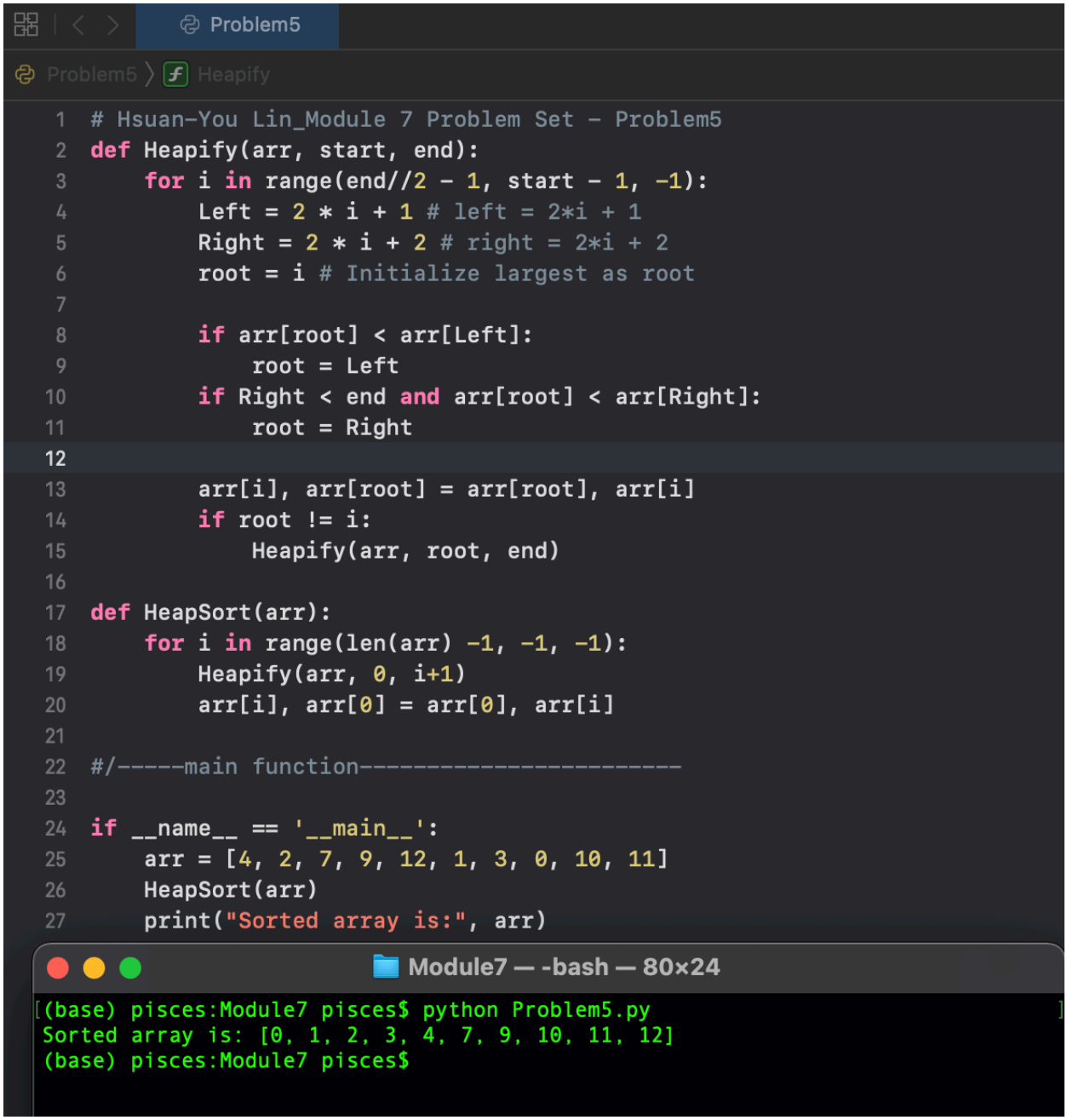
4. My Answer: [11, 10, 4, 9, 7, 1, 3, 0, 2]
My steps:

from problem 2 we got [12, 11, 4, 9, 10, 1, 3, 0, 2, 7]



∴ After above steps, we get [11, 10, 4, 9, 7, 1, 3, 0, 2]. #

5. My pseudocode:



---

6. My Answer: O(n log(n))
My reason:

In my problem 5 pseudocode used a loop "for i in range(len(arr), -1, -1, -1)", which means it has the worst case when the heapify operation requires sinking through the tree's length, let length = n, so we got complexity of O(log(n)).

Thus, we can calculate the worst case complexity of my method is O(n log(n)).