CSCI-SHU 210 Data Structures

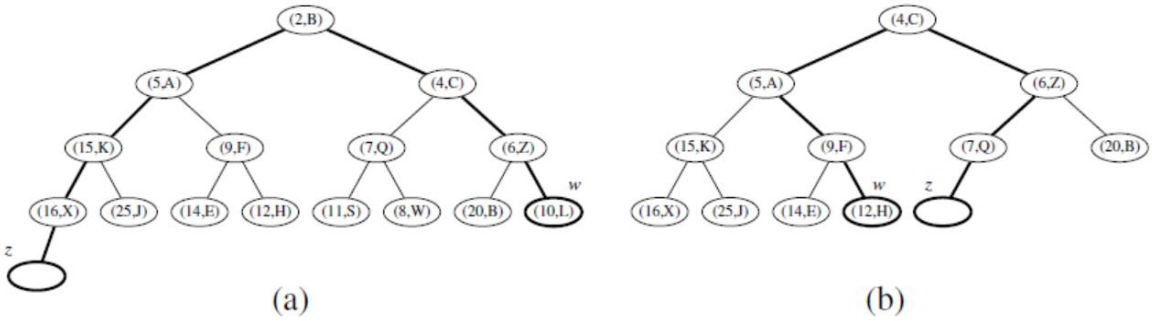
Homework Assignment 8 Heaps, Priority Queues and Hash tables

*Assignment 8 tasks start from line 348 in Assigment8_Heap_Hash.py

Problem 1: Linked representation of a heap (40 points)

Assume we are using a linked representation of a complete binary tree T, and an extra reference to the last node of that tree. Your task is to update the reference to the last node after operations `add()` or `remove_min()` in $O(\log n)$ time, where n is the current number of nodes of T.

Be sure and handle all possible cases, including those illustrated below:



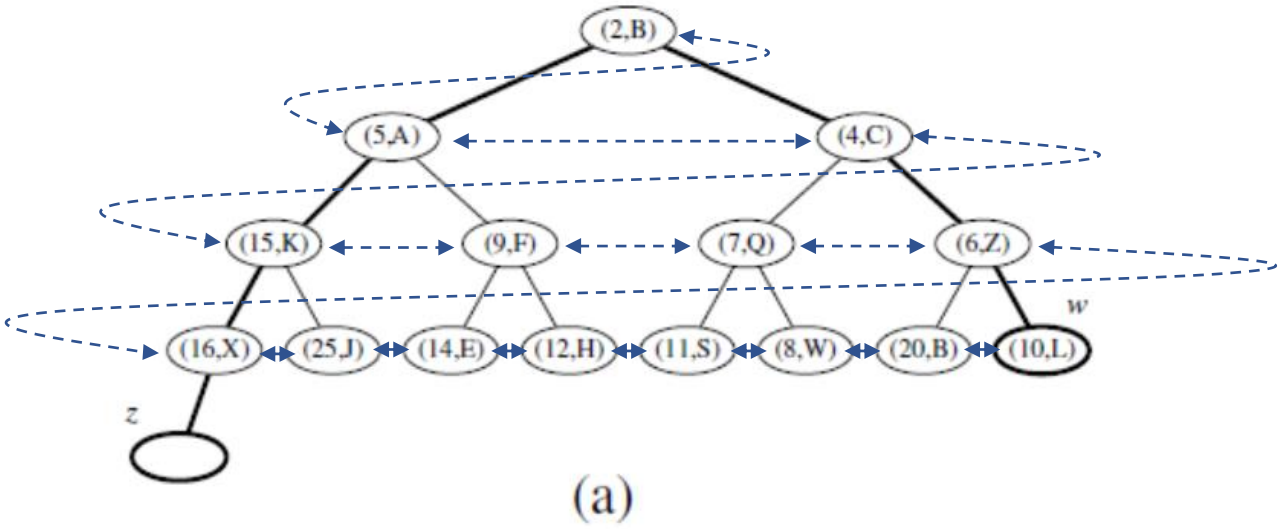
Implementation Requirement:

- You may define other methods. The task is to update the reference to the last node after operations `add()` or `remove_min()` in $O(\log n)$ time.
- No extra storage (e.g. queue, python lists), i.e. $O(1)$ extra memory.
- Hint: write an `after(self, node)` function to find the next **in-order** node.

Problem 2: Linked representation: an alternative way to find the last

(30 points)

An alternative method for finding the last node during an insertion in a heap is to store in each node a reference immediately to its right (wrapping to the first node in the next lower level for the rightmost leaf node). Your task is to **maintain** such references in $O(1)$ time per operation.



(a)

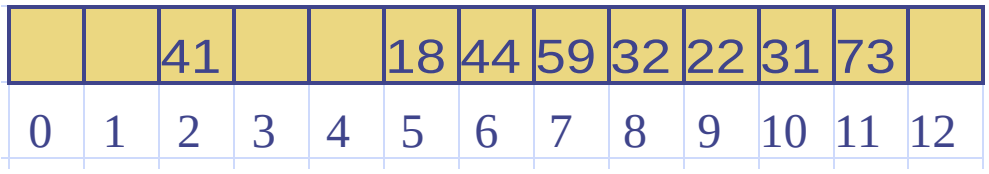
Implementation Requirement:

- You may define other methods. The task is to update the reference to the last node after operations `add()` or `remove_min()` in ***O*(1)** time
- Hint: please include `self._next`, `self._prev` attributes in the TreeNodes to store references to the next (previous) node, just as ADT Doubly Linked Lists.
- No extra storage (e.g. queue, python lists), i.e. ***O*(1)** extra memory.

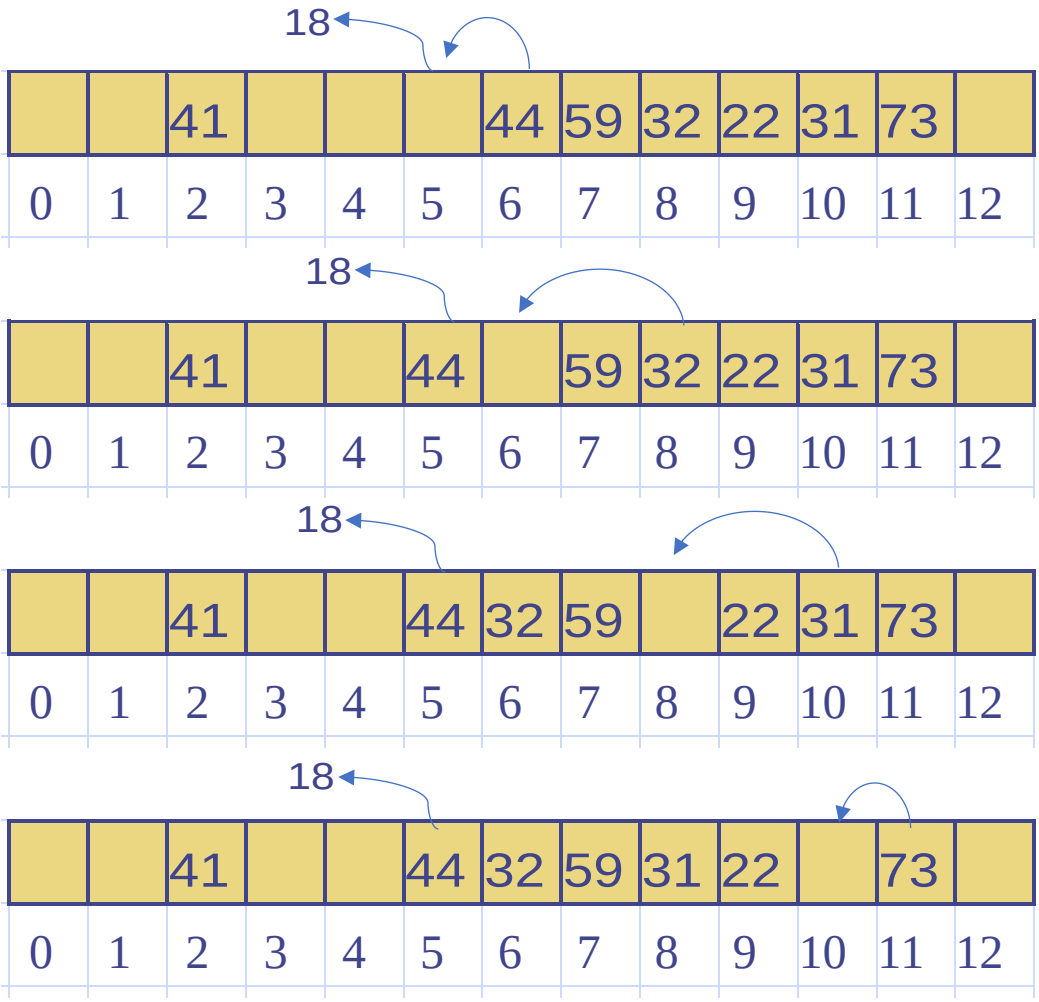
Problem 3: Hash table without _AVAIL object (30 points)

Rewrite the `_bucket_delitem()` method in the `class ProbeHashMap`. It performs a removal from a hash table using linear probing to resolve collisions where we do **not** use a special marker to represent deleted elements. That is, we must rearrange the contents so that it appears that the removed entry was never inserted in the first place.

For example, suppose the hash function is *h(x) = x mod 13*



Given the hash table above, delete 18 will perform the following changes:



Implementation Requirement:

- You need only to rewrite the `_find_slot()` and `_bucket_delitem()` methods
- Time Complexity: ***O(m)***, where *m* is the length of the cluster
- No extra storage (e.g. queue, python lists), i.e. ***O*(1)** extra memory.