CIS 313, Intermediate Data Structures
Winter 2019

CIS 313 Lab 3

Due: Friday, November 22 at 11: 59 pm
This lab involves implementing a Priority Queue using a Max-Heap

Overview
Use a binary Max-Heap to implement a Priority Queue.
Your priority queue class will be a wrapper for the functions in your heap class.
Your heap should be implemented using a list, L.
Recall, if a node is in L[k], then its left child is in L[2k + 1], and its right child is in L[2k + 2]

Fill out all of the methods in the provided skeleton code.
You may add additional private methods, but should not add additional public methods or public fields. You also should not change the name of any of the classes or files.

In particular, the following functionality will be implemented for your priority queue

Note: The code for all these functions has already been provided. Notice that these methods are calling the methods implemented in the heap data structure.

insert
Add priority p to the priority queue.

peek
Return the highest priority from the priority queue

extract_max
Remove and return the highest priority from the priority queue

is_empty
return true if the Priority Queue is empty.

In particular, you will implement the following functionality for your heap

insert
When adding a node to your heap, remember that for every node n, the value in n is greater than or equal to the values of its children, but your heap must also maintain the correct shape. (i.e., there is at most one node with one child, and that child is the left child and that child is as far left as possible). Insert 'data' at the end of the list initially. swap with its parent until the parent is larger or you reach the root

peek
Return the maximum value in the heap

extract_max
Maximum element is first element of the list. swap first element with the last element of the list and remove that element from the list and return it. call __heapify to fix the heap

__heapify
Used to maintain the structure of the heap after an element is added or removed from the heap.

heap_sort()

Note that the heap sort function is outside the class. Initialize a heap using the provided list. Use build_heap to turn the list into a valid heap. Repeatedly extract the maximum and place it at the end of the list. The sorted list should be ascending order. Think carefully about the solution. Appending the extracted elements at the end of the list wont arrange the elements in ascending order. You need to append the first extracted element at the end of the list, the next extracted element at the 2^{nd} last position of the list and so on. Creating helper methods using __heapify might be helpful.

Submission
Compress the mheap.py, pqueue.py and test_lab3.py files and upload in Gradescope similar to the previous programming assignments. The test cases file should contain all the additional test cases that you have written to test your code.

Grading
Similar to previous lab assignments, 70 points will awarded in the test cases and 30 points for style. To earn points for style, your code must be clear enough for us to understand. The rubrics for different functions will be provided in Gradescope.