COMP 2002 – Assignment 3 (5%)

Due: 11:59pm, Nov. 7, 2022

Learning Objectives

The goal of this assignment is to gain experience with heaps.

Instructions

Download the Python source file `heap.py`. It contains the class definition for an array-based implementation of a Heap.

A Heap is a tree that stores the minimum key on top of the heap, at the root node. Heaps have two properties.

1. A Heap is a complete binary tree.
2. The key of a node is less than or equal to keys of any of its children (heap-order property).

Question 1: (25 pts)

The method, `get_keys(depth)`, returns a list containing the keys of all nodes in a Heap that have depth greater than or equal to the given depth. For example, if a depth of 3 is given, then a list is returned that contains the keys for all nodes that are at depth 3 or greater from the root.

```
def get_keys(depth):
    """
    Returns a list containing the keys for all nodes that have depth
    greater than or equal to the specified depth.
    An empty list is returned in the case of an empty heap or a heap
    without nodes at the specified depth.
    """
```

What is the smallest possible Big Oh for this method (i.e., fastest runtime)?

Implement the `get_keys()` method using this Big Oh into the Heap class given in `heap.py`. Provide your Big Oh answer as a comment in your implemented method with an explanation why this is the fastest possible. A woking solution with running time slower than the theoretical fastest will receive partial marks.

You may add secondary or helper methods, but cannot change the definition of `get_keys()`. You may not change the constructor or otherwise modify the class (for this question).

Question 2: (30 pts)

The TernaryHeap ADT is a Heap built using a complete ternary tree instead of a complete binary tree. A ternary tree has a maximum of 3 children per node, instead of a maximum of 2 children per node as with a binary tree. A TernaryHeap still maintains the heap-order property, such that the TernaryHeap ADT has the following properties.

1. A TernaryHeap is a complete ternary tree.
2. The key of a node is less than or equal to the keys of any of its children (heap-order property).

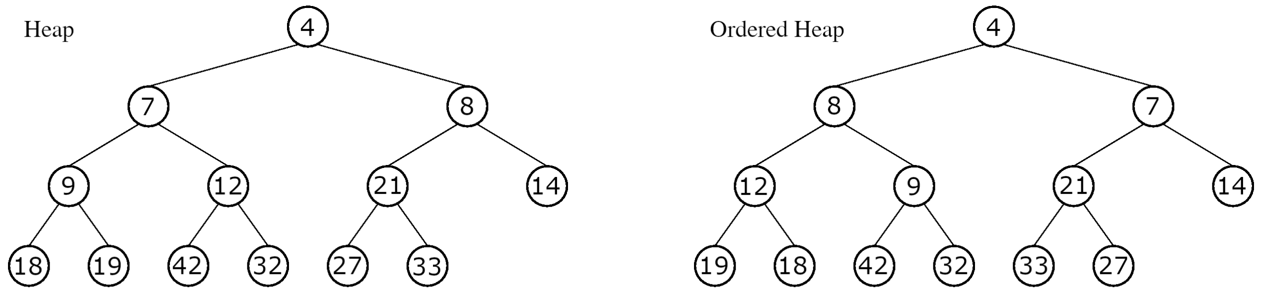
Create a TernaryHeap class in Python using an array-based implementation that implements the `insert(key)` and `remove_min()` operations of the TernaryHeap ADT. You may use the Heap class in `heap.py` as a starting point. The TernaryHeap should maintain the complete ternary tree and heap-order properties.

Question 3: (45 pts)

The OrderedHeap ADT is a Heap with one additional property. In addition to the complete binary tree property and heap-order property, as described in the assignment introduction and in the lecture slides, the OrderedHeap class satisfies the following additional sibling-order property.

1. An OrderedHeap is a complete binary tree.
2. The key of a node is less than or equal to the keys of any of its children (heap-order property).
3. The keys between sibling nodes are ordered such that the key of the left sibling is greater than or equal to the key of the right sibling (sibling-order property).

The figure below shows an example of a set of keys contained in a Heap and an OrderedHeap. Notice that a left sibling stores a larger key than a right sibling for the OrderedHeap.



Create an OrderedHeap class in Python using an array-based implementation that implements the `insert(key)` and `remove_min()` operations of the OrderedHeap ADT. You may use the Heap class in `heap.py` as a starting point and will need to modify up-heap and down-heap bubbling accordingly. Your implementation must maintain the 3 defined properties above.

Submission

Answer the above questions in three separate Python source files, one for each question. For question 1, the source file should contain the Heap class with the added `get_keys()` method. Include the answer to the asked question (smallest possible Big Oh) as a comment. The source file for question 2 will contain the TernaryHeap class, and the source file for question 3 the OrderedHeap class.

Submit a single zip file containing the Python source files through the Assignment submission folder in Brightspace.

Python source code should be `*.py` plain text. The only file types allowed aside from Python source code (`*.py`) are pdfs and plain text (`*.txt`). Do not submit Word documents or rich text format documents. They will not be marked. Only submit a single zip archive. Do not submit files archived in rar format. That may result in your assignment not being graded.

Name all files with the format "firstname_lastname_studentid_...". Make sure to include your name and student ID as comments at the top of all Python source files.

Late submissions will be subject to a 10% penalty for each hour past the deadline.

Attribution

Submissions must represent your independent work.

Submissions should include an attribution section indicating any sources of material, ideas or contribution of others to the submission.

You are encouraged to use any resources to help with your solution, but your solution must represent independent work. If your submitted work includes unacknowledged collaboration, code materials, ideas or other elements that are not your original work, it may be considered plagiarism or some other form of cheating under MUN general regulations 6.12.4.2 and academic penalties will be applied accordingly.

Avoid academic penalties by properly attributing any contribution to your submission by others, including internet sources and classmates. This will also help distinguish what elements of the submission are original. You may not receive full credit if your original elements are insufficient, but you can avoid penalties for plagiarism or copying if you acknowledge your sources.

Github

I encourage you to store and version your work on Github. It is good practice to do so as everyone uses git in the real world.

However, **it is a requirement that git repositories containing assignment material be private.**

University regulations section 6.12.4.2 consider it cheating if you allow your work to be copied. There will be zero tolerance for this.