CS 313 - Project 2
Max-Heap Priority Queue

February 2, 2023

1 Project Overview

For this lab, we will be taking what we've learned so far up a notch and learn how to implement a max-heap priority queue. Priority queues are another commonly used Abstract Data Type (ADT) due to the fact that they are self-sorting (they sort themselves) and that their time complexity ranges from constant $O(1)$ to linear $O(n)$ depending on the operation and how they are implemented. Our central task for this lab is to implement the max-heap variety of priority queues.

2 Program Requirements:

2.1 Write the priority queue class.

You will need to write a **PriorityQueue** class. To receive any credit, your class must follow the specifications below exactly:

- Class Name: **PriorityQueue**
- Methods:
  - $\_\_init\_\_(<PriorityQueue> self, <Int> capacity) \rightarrow <Nonetype> None$
    - **Complexity:** $O(1)$.
    - **Valid Input:** An integer from $(0, \infty]$.
    - **Error Handling:** Raises a **PriorityQueueCapacityTypeError** if the capacity is of the wrong type. Raises a **PriorityQueueCapacityBoundError** if the capacity is negative or 0.
    - Instance variables:
      - $<list>$ *_heap* This is where the heap will be stored. Our heap takes the form of a list.
      - $<Int>$ *capacity*: This variable contains the total number of items that can be fit into the queue.
      - $<Int>$ *currentSize*: This variable contains the current number of items in the queue.
      - Any other instance variables you want.
  - $insert(<PriorityQueue> self, <Tuple> item) \rightarrow <Bool> returnValue$
    - **Complexity:** $O(lg(n))$.
    - **Valid Input:** A tuple $(priority, item)$ containing the following:
      - *priority*: A positive integer in the bound $(0, \infty]$.
      - *item*: Any Python object.
    - **Error Handling:**
      - Raises a **QueueIsFull** exception if the insert method is called when the queue is full.

    - · Raises a **InvalidInputTuple** exception if the input tuple does not satisfy the valid input requirements.
  - ∗ **Description:** This method will add a tuple to the queue based on its priority, then return True upon successfully adding it to the heap. If any other error happens, raise either **QueueIsFull** or **InvalidInputTuple** as required above.
  - ∗ **Note:** When adding a node to your heap, remember that for every position i, the priority of i must be greater than or equal to the priorities of its children, but your heap must also maintain the correct shape. (i.e., for any position i there can be at most two children, and the parent has a greater priority than all subsequent nodes.

- $extractMax(< PriorityQueue > self) \rightarrow < Tuple > returnValue$
  - ∗ **Complexity:** $O(lg(n))$.
  - ∗ **Error Handling:** Raises a **QueueIsEmpty** exception if the *extractMax* method is called when the queue is empty.
  - ∗ **Description:** This method will remove and return the tuple with the highest priority. **Note:** Do not forget to reorder the heap after extraction. (i.e., call $maxHeapify$)
- $peekMax(< PriorityQueue > self) \rightarrow < Tuple > returnValue$
  - ∗ **Complexity:** $O(1)$.
  - ∗ **Description:** This method will return the tuple with the highest priority if the queue is not empty. Otherwise, it will return False and not raise exceptions.
- $isEmpty(< Queue > self) \rightarrow < bool > returnValue$
  - ∗ **Complexity:** $O(1)$.
  - ∗ **Note:** This method will return True/False depending on if the priority queue is empty or not.
- $isFull(< Queue > self) \rightarrow < bool > returnValue$
  - ∗ **Complexity:** $O(1)$.
  - ∗ **Note:** This method will return True/False depending on if the priority queue is full or not.

2.2 Extra Credit:

If you are looking for general bonus points (10%): Implement a heap-sort method (you will already have a max-heapify). The details of this method are given below:

- $heapSort(< PriorityQueue > self, < list > lst) \rightarrow < list > returnValue$
  - **Complexity:** $O(nlg(n))$.
  - **Valid Input:** A list of tuples (a, b) where a is the priority and b is the item.
  - **Error Handling:** Raise the **InputError** on invalid input.
  - **Description:** This method will sort the given list of tuples.
  - **Note 1:** Extra credit is all or nothing.
  - **Note 2:** Do not forget to do input validation! lst must be a list and each item in the list must be a tuple in the same format as specified in the insert method. If it's not, raise the exception.
  - **Note 3:** This extra credit can either be stunningly easy or difficult based on how you think about the methods you've already developed.

3 Submission Requirements:

Submit a single file **p2.py** to canvas.

4 Grading:

Your work will be graded along three primary metrics: Correctness, Completeness, and Elegance.

- Correctness: (60 points)
  - You wrote the class methods as specified and they meet the complexity requirements.
  - You utilize a list to build your heap.
  - You implemented the priority queue with a max-heap.
  - Your classes are robust, fault-tolerant and follow the specified behavior on invalid input.
- Completeness (25 points)
  - The program contains a class named: PriorityQueue
  - The class contains methods as defined above.
  - The method signatures were implemented as specified.
- Elegance: (15 points)
  - See the programming guide posted on canvas for information on elegance.

5 Remarks:

- The methods listed above are the only methods that will be tested. However, you may add additional methods as you please. I would recommend the following:
  - $\_maxHeapify$ – to restructure your heap.
  - $\_swap$ – to swap parent and child tuples.
  - $\_getParent$ – to get the parent tuples index.
- Pay attention to the complexity bounds mentioned in the method descriptions. Your implementation must run within these bounds
- For this assignment, you must use a list to implement your heap. The use of any other built-in data structures is explicitly forbidden. Using them will result in a 0 for the assignment.
- You must implement the priority queue class as a max-heap priority queue as discussed in the book (chapter 6).