HW14 Problem Set

CS-3020

Assignment
Instructor Assigned

HW14-1: (Instructor Assigned #A)

Design an abstract class **Heap** and two concrete classes, **StaticHeap** and **DynamicHeap**, that inherit from it. The public interface required by the **Heap** class is as follows:

Heap(count=0, type=Heap.MINHEAP): Creates a heap that can hold "count" items and places the minimum value at the root of the heap. The alternative is **Heap.MAXHEAP** which places the maximum value at the root of the heap.

Remove(value): Finds 'value' in the heap and removes it. Throws a **HeapEmptyException** if the heap contains no data and a **ValueNotInHeapException** if the heap is not empty but does not contain the requested value.

Add(value): Adds 'value' to the heap. Throws **HeapFullException** if the heap cannot accept any more data.

Pop(): Removes the value presently at the root of the heap and returns it.

Capacity: Returns the total amount of data the heap is currently able to hold.

Count: Returns the total amount of data presently stored in the heap.

Peek: Returns the value presently at the root of the heap.

The three properties are read-only – the user cannot assign values to these properties.

The **DynamicHeap** must be able to grow, if possible, when more data is added than it can presently hold.

The **StaticHeap** must be given a fixed size at instantiation, the default of which is 255 elements.

You may implement the **DynamicHeap** any way you choose and performance is not a factor in your grade. However, your **StaticHeap** should be designed for performance and this will affect your grade.

HW14 Problem Set

CS-3020

HW14-2: (Instructor Assigned #B)

Design and implement class **`RunningHeap`** which inherits from **`Heap`** and implements the **`IRunningStatistic`** interface (which you are to define). The interface has the following public elements:

`Flush(value=0.0)`: Sets all of the values in the data structure to the specified value.

`New(value)`: Adds a new value to the data structure, removes the oldest, and returns the new value of the running statistic.

`Value(n)`: Returns the value of the nth most recent data value. Value(0) returns the most recent value.

`Bit(n)`: Returns True if the nth most recent data value is above the present threshold, False otherwise.

`Threshold`: Returns the current value of the running statistic.

The property is read-only – the user cannot assign values to these properties.

The constructor for RunningStatistic should take two arguments:

```
RunningHeap(size = 1048576, percentile = 66.7)
```

The size (defaults to 2^{10}) is the total amount of data that can be stored in the structure. This need not be adaptable. The percentile is the ranking of the value that serves as the threshold. A value of 66.7 means that 66.7% of the data presently in the heap is less than or equal to the threshold value while the remainder is greater than or equal to the threshold value.

Grading Rubric

Each problem is worth 10 pts (score will be recorded as a percentage of that amount)

10% Properly submitted
10% Properly named
20% Adequate comments
10% Runs
20% Produces correct output
30% Effort evidenced by the submitted work