**PriorityQueue.cs**

Overview

* An implementation of the Priority Queue data structure

Variables

* data: a list of Tuples, that are composed on an object T and float pair

Functions

* Constructor
  + Declares and initializes the Priority Queue
* Enqueue
  + Input: an item to put into the Priority Queue as well as its weight
  + Places the given item into the Priority Queue and, will resort the Queue so that the item with the highest priority is put in the very front of the Queue. Note: higher priorities are lower numbers. Below is the sorting algorithm:
  + 1. ci = data.count - 1
  + 2. While (ci > 0)
    - 3. pi = (ci - 1)/2
    - 4. If (compare(data[ci], data[pi]) >= 0)
      * 5. Break
    - 6. Swap the items in index ci and pi
* Dequeue
  + Removes the data from the front of the list and resorts the list as necessary
  + Output: data at the front of the list
  + 1. If the data list is empty, return null
  + Remove the item from the front
  + 2. li = data.Count - 1
  + 3. frontItem = data[0]
  + 4. Data[0] = data[li]
  + 5. data.removeAt(li)
  + 6. --li
  + 7. pi = 0
  + Resort
  + 8. While (true)
    - 9. ci = pi \* 2 + 1
    - 10. If ci > li
      * 11. Break
    - 12. rc = ci + 1
    - 13. If (rc <= li && compare(data[rc], data[ci]) < 0)
      * 14. Ci = rc
    - 15. If (compare(data[pi], data[ci]) <= 0)
      * 16. Break
    - 17. Swap the items in index pi and ci
* Count
  + Returns number of items in the Priority Queue
* Compare
  + Compares two given items in the data list and will return a number depending on their priorities
  + Input: first item Tuple, second item Tuple
  + Output: -1 if first item’s priority < second priority
    - 1 if first priority > second priority
    - 0 if first and second are equal
* Peek
  + Returns the item that is at the front of the data list

**Tuple**

Overview

* Tuple data structure that is composed of two items of any data type

Variables

* Item 1
* Item 2

Functions

* Constructor