

CS 201 Data Structures Library Phase 3 Due 4/20

Phase 3 of the CS201 programming project, we will be built around heaps. In particular, you should implement both a standard binary heap and binomial heap. You will implement a class for each heap type.

You should create a class named `Heap` for the binary heap with public methods including the following (`elmttype` and `valuetype` are types from the template). You should use your dynamic array class for the heap storage.

Function	Description	Runtime
<code>Heap();</code>	Default Constructor. The Heap should be empty	$O(1)$
<code>Heap(keytype k[], valuetype V[], int s);</code>	For this constructor the heap should be built using the arrays <code>K</code> and <code>V</code> containing <code>s</code> items of <code>keytype</code> and <code>valuetype</code> . The heap should be constructed using the $O(n)$ bottom up heap building method.	$O(s)$
<code>~Heap();</code>	Destructor for the class.	$O(1)$
<code>keytype peekKey();</code>	Returns the minimum key in the heap without modifying the heap.	$O(1)$
<code>valuetype peekValue();</code>	Returns the value associated with the minimum key in the heap without modifying the heap.	$O(1)$
<code>keytype extractMin();</code>	Removes the minimum key in the heap and returns the key.	$O(\lg n)$
<code>void insert(keytype k, valuetype v);</code>	Inserts the key <code>k</code> and value <code>v</code> pair into the heap.	$O(\lg n)$
<code>void printKey()</code>	Writes the keys stored in the array, starting at the root.	$O(n)$

Your class should include proper memory management, including a destructor, a copy constructor, and a copy assignment operator.

You should create a class named `BHeap` for the binomial heap with public methods including the following (`elmttype` and `valuetype` are types from the template). You should use dynamic allocation for the binomial trees.

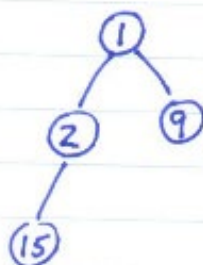
Function	Description	Runtime
<code>BHeap();</code>	Default Constructor. The Heap should be empty	$O(1)$
<code>BHeap(keytype k[], valuetype V[], int s);</code>	For this constructor the heap should be built using the arrays <code>K</code> and <code>V</code> containing <code>s</code> items of <code>keytype</code> and <code>valuetype</code> . The heap should be constructed using repeated insertion.	$O(s)$
<code>~BHeap();</code>	Destructor for the class.	$O(n)$
<code>keytype peekKey();</code>	Returns the minimum key in the heap without modifying the heap.	$O(\lg n)$
<code>valuetype peekValue();</code>	Returns the value associated with the minimum key in the heap without modifying the heap.	$O(\lg n)$

keytype extractMin();	Removes the minimum key in the heap and returns the key.	$O(\lg n)$
void insert(keytype k, valuetype v);	Inserts the key k and value v pair into the heap.	$O(\lg n)$
void merge(BHeap<keytype, valuetype> &H2);	Merges the H2 into the current heap	$O(\lg n)$
void printKey()	Writes the keys stored in the heap, printing the smallest binomial tree first. When printing a binomial tree, print the size of tree first and then use a modified preorder traversal of the tree. See the example below.	$O(n)$

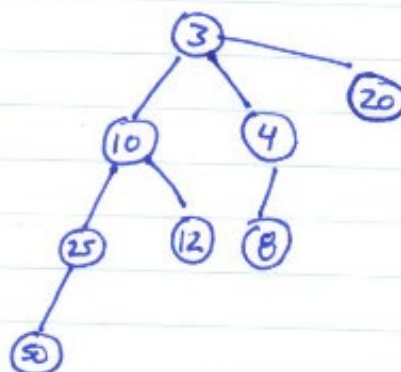
B_0

(6)

B_2



B_3



Print key()

B_0
6

B_2
1, 2, 15, 9

B_3
3, 10, 25, 50, 12, 4, 8, 20