Yutong He

Professor Ted Pawlicki

CSC 172

April 12, 2016

Skew Heap

I. Introduction and Purpose

While we consider binary heap as a perfect, elegant and efficient data structure, it is hard to merge two heaps. Since heaps are constructed by using arrays, to merge two heaps requires to expand one array, copy another one into it and re-heapify it, which takes O(N)'s running time to do so. It seems pretty difficult to use only array to implement merging. For this reason, a linked data structure is designed to efficiently support merging – skew heap.

Skew heap is a self-adjusting heap data structure implemented as a binary tree, or we can say, a binary heap-ordered tree. Like heap, it contains a priority-associated relationship between parents and children; like binary tree, it retains the structural retrievable capability among parents and children; and with relatively simple method and implementation, it is able to adjust itself for potential merging without any balance information. In this paper, I am going to discuss the methods and functionality of skew heap, the primary difference from related common data structures, and the academic and industry application of skew heap.

II. Difference From Other Data Structures

i. Ancestors: Binary Tree and Heap

As mentioned before, a skew heap is both a binary tree and a heap.

Unlike heap, skew heap is not an array-based data structure but a node-based data structure. For insertion, heap is inserted by adding elements to the array, expanding the array, and bubbling up while skew heap is inserted by merging a one-node skew heap with another heap. For deletion, the root of heap is deleted by replacing the root by the last element in the array and bubbling down while the root of skew heap is deleted by merging the left and right child and replacing the root. Additionally, because of its consistently swapping merging algorithm, skew heap may not be completely left-oriented like heap.

Unlike binary tree, skew heap satisfies heap priority rules to manipulate its nodes.

While binary tree is a relatively abstract concept, skew heap is a specific data structure with special implementation.

Overall, the skew heap is different from its ancestor data structures basically because of its unique merge method, which I will elaborate later in section III.

ii. Twin Sibling: Leftist Heap

Both designed for the purpose of merging to heaps, having the same heap-order property and constructed as binary trees, skew heap and leftist tree are like twin siblings – look similar, sometimes exactly the same, but essentially are two different data structures.

Speaking of leftist heap, we have to mention the null path length, npl(X), which defines the length of the shortest path from X to a node with no or only one children(Weiss 241). Closely associated with the null path length, the leftist heap property is that for every node X in the heap, npl(X.left) is greater or equal to npl(X.right). When merging two heaps, we recursively merge the right subtree of the tree with higher priority root with the tree with the lower priority root. After merging, because we need to maintain the leftist heap property, the null path length of the new right subtree decides whether we need to swap the left and right child of the root. In other word, if the new npl(root.right) is greater than npl(root.left), it is the time when we swap the left and right child of the root to fix the root for its leftist heap property. Because we only need to operate portions of the whole heaps to complete the merge and constant time work is performed at every recursive call, we can obtain O(logN) time bound to merge to heaps.

While leftist heap has the balance field to decide whether and when to swap the children, skew heap, which is a self-adjusting version of leftist heap (Weiss 249), possess no structural constrains. There is no balance information stored in skew heap node and when merging, we always swap the two children of the higher priority root. As a result, a skew heap may have longer right path than left path, which is not acceptable in binary heap and leftist heap. As for running time, even though it seems that the worse-case is O(N) because of the arbitrarily long right path, amortized complexity analysis demonstrates that the worst-case total running time of its operation is O(logN)[1]. Additionally, because skew heap does not contain the balance information, it takes up less memory. Skew heap also has the advantage

[1] See Chapter 11 (11.3) in *Data Structure And Algorithm In Java* (Weiss) for more details.

of less extra space to maintain the path length and fewer tests to determine the swapping. For these reasons, skew heap is a relatively efficient data structure that is simple to understand and implement. However, the recursion might fail due to lack of stack space because of over-long right path.

III. Methods and How Do They Work

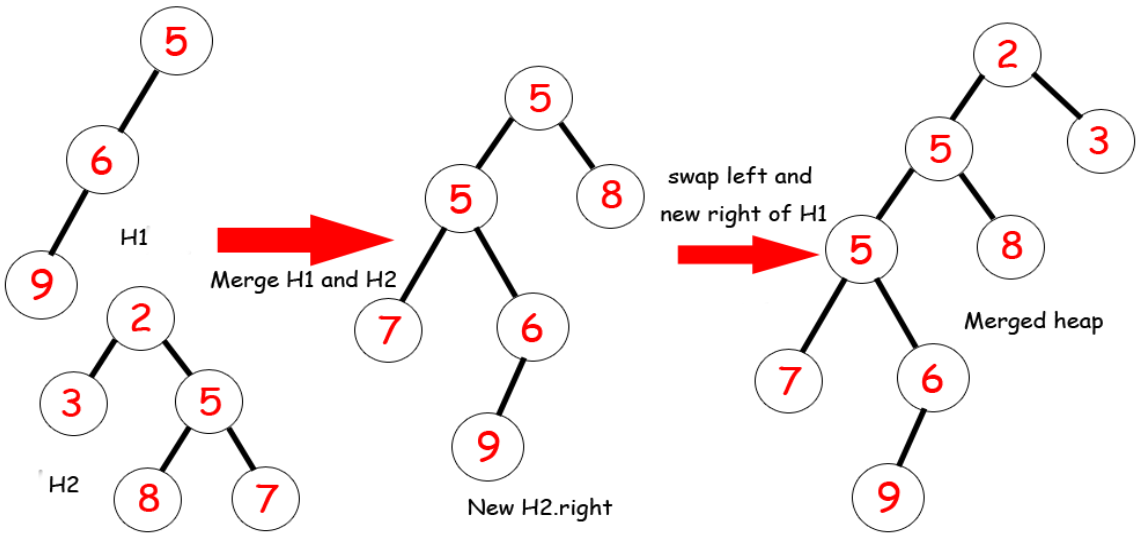
The essential method of skew heap is merge(). Actually, it is safe to say that skew heap is the data structure constructed based on the merge method. In this section, I am going to introduce the node class, the heap class, insert, is empty, find min, delete min[2], merge two heaps and print methods and elaborate the merge method and the relationship between merge and all the other methods.

i. Node Class and Heap Class

The node class of the skew heap is basically the same as the one of binary tree – generic, with data, left and right child, and with three different constructors which takes different parameters to construct the heap node. I also put the three printing methods in the node class and call them in the heap class just like in regular tree class.

The heap class of the skew heap contains two fields: the root node and the integer size. When constructing a new skew heap, the constructor sets the root to be null and the size to be zero. To test whether the skew heap is empty, just check whether the root is empty. General getters and setters are also included in the class.

ii. Merge Method



[2] In my algorithm, my skew heap is a min heap, in which the less the value, the higher the priority.

We merge two skew heaps node by recursively call the merge method, which takes two skew heap nodes as parameters and returns a skew heap node.

Basic Step: if one of the skew heaps nodes is null, return the other one.

Recursive Step: if the first skew heap node (we call it n1) has higher or equal priority, which mean n1 has the smaller or the equal integer value in min heap, than the second heap node (we call it n2), merge the right child of n1 and n2 and let the new merged skew heap node be the new right child of n1. Then swap the left and right child of n1. Return n1 and end the method. If n1 has lower priority than n2, return the skew heap node created by calling merge method again but put n2 and n1 reversely in the parenthesis (i.e. originally calling merge(n1, n2), but now calling merge(n2, n1)).

iii. Insert, Delete Min, Find Min, and Merge Two Heaps Method

We use merge method to manipulate the above method in skew heap. All of these methods can keep track of the size if necessary.

Insert: The insertion in skew heap is just a special case of merging – taking the new data as parameter and merging the new one-node skew heap with a normal skew heap.

Delete min: The deletion in skew heap is simply merging the left and right child, replacing the root by new merged heap, and returning the original root data.

Find min: The lookup or peek method in skew heap is simply returning the data of the root of the skew heap.

Merge two heaps: The merge two heaps method is a static method which creates a new skew heap instant and set the root of the new skew heap be the merged skew heap node of two roots of the two skew heaps it takes as parameters and return the new skew heap.

IV. Academic and Industrial Application

Because skew heap is essentially a heap structure, it is helpful to use skew heap in some common applications of heap where we may need to merge two heaps such as selection algorithm which can get access to the max or min in constant time. However, since it is not array-based data structure, some of the applications of heap such as heap sort, priority queue are not available for skew heap.

Work Cited: Weiss, Mark Allen. *Data Structure and Algorithm Analysis in Java (Third Edition).* Upper Saddle River: Pearson. 2012. Print.