Bong Lee
Asst08
CS302

Priority Queue (Heap)

Summary

Priority queue data structure is a special kind of queue implemented using a heap. In a priority queue every element has a value representing its priority. The order in which the element is dequeued is based upon the value of its priority. The min/max priority items get dequeued first ignoring the LIFO approach. The priority queue is usually implemented using almost complete binary tree called a heap. The elements are instead into the tree to from a complete tree. The elements are also positioned based on their priority values. The root always contains the max/min value depending on the type of heap. Maintaining the complete tree structure and the order is crucial, therefore special operations are needed to sustain the tree.

Priority Queue vs BST

The biggest advantage of heap is that it can retrieve the top priority element in constant time. On the other hand getting min/max values in a BST requires O(log n). Another great advantage of using a heap is that it is generally easier to implement. It can be implemented using an array without use of pointers. If the size of the data is known the heap can also build itself in O(n) while BST would require O(n log n). The major downside of heap structure is that search operation would not be very efficient. If search is required on the data using a BST would be a much better choice. Heap is a specialized data structure that can do one thing very well which is to retrieve min/max values. If the data requires more diverse set of operations, then BST may be a better choice.

Build Heap Function

Heaps can be built in O(n) time if the size of the data is known. The operation inserts all the data into the tree in the order they are given. Then the function starts heapifying the tree starting from the heapSize/2 node. The process builds a complete heap from the bottom up. If done correctly it can build a heap in O(n) time. The operation can be efficient, but if done incorrectly it may not satisfy the order property of the tree.

Bong Lee
Asst08
CS302

Heap Common Operations

Insert	O(log n)
buildHeap	O(n)
deleteMin	O(c)
reheapUp	O(log n)
reheapDown	O(log n)
resize	O(n)