Ryan Young

10/7/19

CSE 3500 Midterm #1 Review

Priority Queues:

Operations: Addition of Elements, Deletion of Elements and Selection of Element with Smallest Key

All operations in a priority queue are done in O(log(n)) time complexity

Sorting with a priority queue takes O(n\*log(n)) time! This makes it a useful tool to sort things with. This is because we can insert items into a priority queue and then repeatedly call the remove minimum operation. This would be n operations are need and each operation takes log(n) time complexity thus giving us the time complexity to sort ^^^.

Heaps:

At all times must respect the heap property and the balanced binary tree property. When violated the heap must be restored via the Heapify function.

The heaps are listed in an array in a bfs approach ie, the node then its left child, right child. Then it will go to the left child get its left and right child, move back to the right child and get its left then right child. AKA it’s almost a bfs node left right then move to left child get its children then right child get its children and repeat.

Heapify Up Function:

H is almost a heap but a value violated the binary search tree aspect of the heap so we need to recursively swap that node with it’s parent until it restores the validity of the heap. This swapping takes place in log(n) time.

Heapify Down Function:

H is almost a heap but a value violated the binary search tree aspect of the heap so we need to recursively swap the value down aka swap it with its SMALLEST child until the heap is restored. This swapping is done in log(n) time.

Remove Minimum Function:

Pop the root element because this has to be the smallest element in order for a heap to be a heap! Then place the last element at the root. This will now ruin the heap because it will have a higher value than its children. This is fixed by calling heapify down function and thus restoring the heap.

Add New Element Function:

Insert the new element at the end, check if the heap is valid, if not then called the heapify up function until the heap is restored to a heap.

Time Notations:

Big O Notation: Asymptotic Upper Bound. The worst-case running time of a certain algorithm of input size n.

Big Ω Notation: Asymptotic Lower Bound. The best-case running time of a certain algorithm of input size n.

Big Θ Notation: Asymptotic Tight Bound. Found using the limits using Big O and Big Ω.

Properties of Asymptotic Growth:

Transitivity:

If a function f is bounded by a function g that is in turn bounded by h, then h is a bound for f. If a bound applies to two functions, then it applies to their sum.

Sum of functions:

Polynomials, Logarithms, Exponents:

Common Run Times:

Linear Time:

Logarithmic Time:

O(n\*log(n)) Time:

Quadratic, Cubic, O(nk) time

Exponent Time:

Graphs:

Directed:

Connectivity (strong components):

DAGS:

Topological Ordering:

Un-directed:

Traversal:

Bredth First:

Depth First:  
Greedy Algorithms:

Interval Scheduling and partitioning:

The greedy algorithm always stays ahead

Minimize latness:

An exchange argument