CSE332 Au 2010 Midterm Review Worksheet

Fill out the run-times for the following algorithms.

For the best case, assume you can setup the data structure however you like for some large value n elements (where n is given to you), and you get to choose the parameter to the operation, if appropriate.

For the worst case, try to come up with the worst possible scenario that could occur for some data structure of size n.

	Best	Worst
Stack (Array) Push; assume that we never need to expand the array-size	Θ(1)	Θ(1)
Stack (Array) Pop	Θ(1)	Θ(1)
Binary Heap Insert (assume we never need to expand the array)	Θ(1)	Θ(logn)
Binary Heap Insert (assume we may have to expand the array)	Θ(1)	Θ(n)
Binary Heap DeleteMin	Θ(1)*	Θ(logn)
Binary Heap Build Heap	Θ(n)	Θ(n)

*The best case for Binary Heap delete is rather tricky, but is constant-time for reasons pointed out during section by a student: Consider a large binary heap with root value 1, whose right sub-tree has a root of 2, but all other values in the right sub-tree are huge; say at least 10^6. The left sub-tree contains values all greater than 2, but less than 10^6. Say the bottom row of the tree is half full. A deleteMin will take that last element, currently in the left subtree, place it at the root and percolate it down. It will be swapped with the 2 of the right sub-tree but stop there, since the other elements of the right sub-tree are all enormous. The result is percolating a single level down, so Θ(1) time.

b. Dictionaries

	Best	Worst
BinarySearchTree Insert	Θ(1)	Θ(n)
BinarySearchTree Find	Θ(1)	Θ(n)
BinarySearchTree Delete	Θ(1)	Θ(n)
AVLTree Insert	Θ(logn)	Θ(logn)
AVLTree Find	Θ (1)	Θ(logn)
AVLTree Delete	Θ(logn)	Θ(logn)

SplayTree Insert	Θ(1)	Θ(n)
SplayTree Find	Θ(1)	Θ(n)
SplayTree Delete	Θ(1)	Θ(n)
HashTable Insert (assume separate chaining)	Θ(1)	Θ(n) †
HashTable Find (assume separate chaining using a linked list)	Θ(1)	Θ(n)
HashTable Rehash (assume separate chaining using a linked list)	Θ(n)	Θ(n)

†This assumes that we do not allow duplicates, and will have to iterate through every item in the linked list at each index in order to make sure that element isn't already there before inserting. If we don't care about duplicates, insert will be constant time.

c. Graphs

	Worst
Topological sort	Θ (|V|+|E|)