Topic 4: Asymptotic Analysis of Basic Set ADT, Section 1 Group __

Group Members Present: Firstname Lastname <uhemail@hawaii.edu>

- Name
- Name
- Name
- Name

---

1. (3 pts) For each of the four types of lists in the following table, what is the asymptotic worst-case running time for each dynamic set operation listed given a list of length *n*? **Fill in using Θ**

- ***k*** is the **key** (set element) and ***p*** a **position** in the data structure (see web notes)
- Assume that **keys are unique**.
- **Search returns a position.** We don't have to search for the item if we have a position p for it.
- Sorted lists are sorted in **ascending order** of the relation "<"
- Predecessor, successor, minimum and maximum are **with respect to ordering of keys in the set under "<"** (which may differ from the ordering of the list data structure).

	Unsorted, Singly Linked (no tail pointer)	**Sorted, Singly Linked** (no tail pointer)	**Unsorted, Doubly Linked Sentinel and Tail pointer**	**Sorted, Doubly Linked with Sentinel and Tail pointer**
minimum()	Θ(____)	Θ(____)	Θ(____)	Θ(____)
maximum()	Θ(____)	Θ(____)	Θ(____)	Θ(____)
Using key k (set element):				
insert(k)	Θ(1)	Θ(n)	Θ(1)	Θ(n)
search(k)	Θ(n)	Θ(n)	Θ(n)	Θ(n)
delete(k)	Θ(____)	Θ(____)	Θ(____)	Θ(____)
successor(k)	Θ(____)	Θ(____)	Θ(____)	Θ(____)
predecessor(k)	Θ(____)	Θ(____)	Θ(____)	Θ(____)
Using position p (referencing a linked list cell containing a key):				
delete(p)	Θ(n)	Θ(n)	Θ(____)	Θ(____)
successor(p)	Θ(____)	Θ(____)	Θ(____)	Θ(____)
predecessor(p)	Θ(____)	Θ(____)	Θ(____)	Θ(____)

Justifications (not graded, but in case you want to explain assumptions):

minimum():

maximum():

delete(k):

successor(k):
predecessor(k):
delete(p):
successor(p):
predecessor(p):

2. Printing Binary Trees (2 pts)
(a) Write a Θ(n)-time recursive procedure that, given an n-node binary tree, prints out the key of each node of the tree in **preorder.** Assume that trees consist of vertices of class **`TreeNode`** with instance variables **`parent`**, **`left`**, **`right`**, and **`key`.** Your recursive procedure takes a **`TreeNode`** as its argument (the root of the tree or subtree being considered by the recursive call).

```
printTreeNodes(TreeNode root)
1
2
3
4
5
```

(b) Prove that your algorithm is Θ(n)-time. If you run into difficulties ask for a hint.

Challenge Problems (if you have time)

3. Which of the Θ results in the ADT runtime table must be changed if we do not assume unique keys, and why? (Describe the worst case scenario.)

4. Modify `printTreeNodes` to print out the nodes in in-order:

5 Modify `printTreeNodes` to print out the nodes in postorder:

If you finish early, you may also discuss these problems (but don't type in solutions):

- How would you rewrite your printTreeNodes procedure to use iteration (for or while loop) and a stack

rather than recursion?

- How would you modify your printTreeNodes procedure to print out the nodes of an N-ary tree using the left-child right-sibling representation in arbitrary order?