CSC 207 2014S: Extra Session, Week 9

Overview

- Admin
- You ask questions.
- I try to give answers, or at least direct you in the right direction.

Admin

- Cool tech conference today.
- CS table tomorrow.
- New Q&A in exam.

On problem #3, what do I do with the fact that I'm being forced to treat nodes as objects, which means I can't get the data field?

```
Node myNode;
T value = myNode.next[height].data;
```

Problem: next is an array of objects, so we get type errors.

Solution: Cast and note unsafety.

```
Node nextNode;
nextNode = (Node) myNode.next[height];
T value = nextNode.data;
```

Note that you may have to suppress warnings

```
@SuppressWarnings({"unchecked"})
```

Note that Sam wrote a helpful method called `next`

_Can I use `myNode.next(height).data`?

Yes, that's the way to use the method.

For problem 4, we need an upper-bound and a lower bound. The algorithm is supposed to be O(n). How do we get the upper bound other than starting at the front and iterating until we run out of elements?

That's the only way I know, but that's O(n). Try to do it only once.

Should I use iterators for problem 1?

Probably not.

Why do you provide a range class in problem 1?

The DNF algorithm needs to communicate the bounds of small and large back to Quicksort. Two strategies for doing so. Strategy one: Make a separate DNF method and have it return a Range object. Straetgy two: Shove the code for DNF in the middle of Quicksort. Then you know what small and large are.

Why am I getting a Stack Overflow?

"Stack overflow" means "you recursed much more than is reasonable." Guesses: Not shrinking array, insufficient base case, ...

Do we have to use `previous` in the last problem?

Almost certainly.

How do I get a private repo?

http://github.com/education

_What is the relationship between `SkipList` and `SortedList` and `SortedSkipList`?

`SkipList` is a class/data structure designed to implement the Skip List data structure. We can use them for all sorts of purposes.

`SortedList` is an interface/ADT designed to describe what we mean when we say "a sorted list - a list that you iterate from largest to smallest or smallest to largest"

`SortedSkipList` is an adapter that makes skip lists meet the sorted list interface.

Do we have to implement the iterator remove method with the skip list remove method?

No, but I think it will be easier.

Do we have to look for the next largest element at each call to `next`?

Think about the underlying design of skip lists. It won't be O(1) if you look at every element.

Can you explain the kthSmallest algorithm?

I did an example on the board.

For problem 4, how do I know the position?

Amazingly, `ListIterator` objects include `nextIndex` and `prevIndex` methods.