CS280 Notes 1/25

by Christian Sagel

What's the the const throw (*BListException*) on the right side of the node method you exampled?
Don't use that code! Exception specifications used to tag functions. THey are deprecated now. They told you what exceptions your function could throw?

Do the copy and assignment constructors make deep copies or just copy the pointers? (shallow copy)

It's not just ANOTHER linked list.

When did we learn to do binary search?
We did a binary search in one of the vector labs in CS170.

Sorting Algorithms

Bubble sort (exchange sort): A very simple algorithm that works great on small lists but is very inefficient on large lists. Compares adjacent items if they are out of order. (Out of order depending if you want items sorted)
We can modify the algorithm to break out early if we find it to be sorted.
Best case: 0 swaps when data is fully sorted.
Worst case: n squared
Average case: Randomly distributed, (N^2) / 2

Would this algorithm work for linked lists? We could. Swapping will be faster than with arrays since we swap pointers.

Insertion sort: Divide the array into two portions. A sorted portion and an unsorted portion.
How can we start with a sorted list? We partition the first element as the sorted half.
Best case: n comparisons, 0 shifts
Worst case: n(n-1) / 2. same number of compares.
Average case: Kind in the middle like bubble sort.

Will this algorithm work with linked lists? Yes it can. Advantages? Shifting is cheap.
You can use binary search to find insertion points.

Selection sort: Scan through the array looking for the smallest elements in row.
Best case: 0, because sorted

Worst case: n number of swaps. n squared over 2 compares
Average case: It's not n/2.

The comparisons are expensive. It comes down to the cost of
Works for linked lists just fine. Unlike selection sort, we just swap pointers.

On small data sets, bubble sort can blow away quicksort (lol)
A complex algorithm may require too much maintenance from humans.
Sometimes an almost optimal algorithm is better than an optimal one.
If we have insight into the data distribution, we can predict (and use) the average-case analysis instead.

Hashing will scramble data, try to uniformly distribute it. Hashed data will be faster to operate on than skewed data.

Understand the loop to understand whether the algorithm is n squared or log n…
"Browsers all suck." - Mead
"That's not opinion. That's fact." - Mead

Mead is not very happy about the holidays.

Recursive Algorithms and Functions

He will teach it to us like how it was taught to him in Graduate school. He then got ripped a new one.
Take it on faith, like a religion.

Recursive definition: A definition in which something is defined in terms of smaller versions of itself.
Base case: The case for which the solution can be stated non-recursively. You need some code that will stop it from calling itself without some stopping point.
Divide and conquer: Divide the work to be done in smaller and smaller pieces, and conquering those smaller pieces.

There is a whole branch of math that will prove whether code will run as it is.
Mead prefers the hands-on over the theory.

In programming terms, recursion is much like iteration, except instead of jumping to the top of the loop, the function calls itself again.
IF the last statement of the function is the function call itself, it is called tail recursion.

Stack frame: parameters, return address, local variables
There used to be a different between non-optimized and optimized compilers.

Quicksort: Partition the data. Quicksort the left, quicksort the right. Again!
Keep going until there's only 1 element remaining in the arrays.
The idea is: it's easier to sort 2 smaller arrays than 1 larger one.
We have to pick a value in the array, the pivot value.

Typically algorithms check the values of the leftmost, middle and rightmost points and returns the median value from those, uses it as a pivot point and starts partitioning.

Never use a conditional operator (? :) in any code on any exam.
Coming up with a recursive algorithm is harder than actually implementing it.

Iteration/Recursion: You work over a collection of something.
How to print a string?
You print the head, the first character, then the tail, the rest of the string