Stack, Queues, and Priority Queues: Linked List Based

10-24-2002

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- I'm going to be leaving campus right after class to fly to the D.C. area and I won't have e-mail access during the weekend so asking questions now could be vital.

O(f(n)) Notation

- When we say that an algorithm is of the order of some function of n, what we are saying is that the number of operations it does grows with the input size in the same way that function does.
- It is called an asymptotic notation because it is most accurate for large n and we throw away all coefficients and only keep the largest terms.

Stacks as Linked Lists

- We have looked at how we can implement the Stack interface with an array, but we can also do it with a linked list.
- For a linked list stack, we only need a head pointer and all the pushes and pops go on it or pull from it.
- The main conceptual difference from an array based stack is just which "end" we are pushing to and popping from.

Queues as Linked Lists

- Queues with arrays required a bit of extra thinking to make them circular. They are actually easier with a linked list.
- We keep both a head and a tail. One is the front and the other is the back of the queue. To figure out which is which, think about which one you can easily remove from.
- We make the choice because we want O(1) operations.

The Priority Queue ADT

- An ADT that is slightly more advanced than the Stack or Queue is the Priority Queue. This ADT acts like a queue, but with the added complication that the elements have a priority.
- When elements are removed from it, it is always the highest (or lowest) priority element that is taken out next.
- We want be be able to find that element fast. Fast adds are nice too.

Sorted Linked Lists

- We can easily make a linked list data structure that is sorted by modifying the insert method so that it inserts the new node into the proper position in the list to be sorted.
- Building a sorted linked list is almost like an insertion sort. The problem is that the insert is a O(n) operation.

Using SLLs for Priority Queues

- If we build a sorted list based on priority, then it automatically works as a priority queue. Items are always removed from the front of the list and inserted where they belong in the list.
- This gives fast, O(1), removes, but the adding is O(n). We'll look at a faster alternative later in the semester.

Code

- Now we will look at code for some of these things.

Minute Essay

- How do you think the linked list based queues and stacks compare to those we looked at using arrays?
- Remember that design #5 is due today.
- Have a great fall break.