

Stack, Queues, and Priority Queues: Linked List Based



3-8-2005

Opening Discussion



- What did we talk about last class? Does anyone have any code to show?
- Do you have any questions about the assignment? Remember that assignment #4 is due today.
- Let's look at some code related to stuff from last class real quick.
- Exams and extra credit.

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 to 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.

Testing Code



- Testing of code is a very important topic and something that you should be doing. In some cases, just running the game does a fairly good test, but it isn't always the easiest one to debug.
- Remember that in Java you can put a main in any class. You should put a main in things like your linked list class where all you do is test and print out stuff.

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 assignment #4 is due today and quiz #4 is on Thursday.