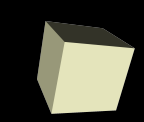
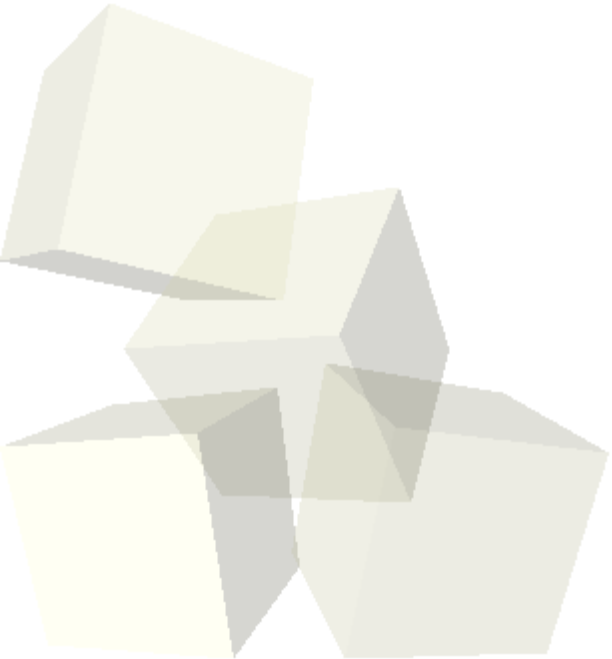


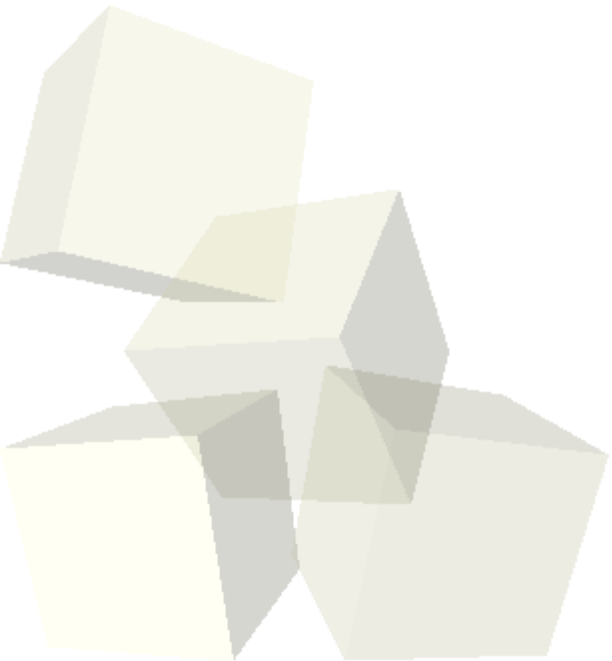
Fibonacci Heaps

2-14-2006



Opening Discussion

- What did we talk about last class?
- Do you have any questions about the the test?
- Do you have any questions about the assignment?



Fibonacci Heaps

- In many ways a Fibonacci heap is like a binomial heap. The primary difference is that the Fibonacci heap is lazy. It intentionally pushes off work until the last possible instant. Unlike with your classes, this can lead to better efficiency for data structures.
- Like the binomial heap, the Fibonacci heap is a collection of trees. In this case it is a collection of trees that are close to being unordered binomial trees.
- An unordered binomial tree is like a binomial tree other than the children to not have to be ordered by degree.

Other Differences

- For various reasons, the lists in a Fibonacci heap are circular doubly linked and the heap keeps track of the minimum root and number of elements.
- The nodes are also enhanced with a bit more data. Obviously more pointers are needed for the doubly linked list aspect. Each node also stored its degree and a boolean marker that is used for some operations.

Adding

- Adding to a Fibonacci heap is quite simple. We do nothing more than add a new node with our data to the root list of the heap. If the new node has a lower value than our current min we make it the min. This operation is O(1).
- Basically, we are adding a U_0 tree to our heap.
- Obviously, a long string of adds will produce nothing more than a linked list where we are keeping track of the smallest one.

Find Minimum and Union

- The peek operation to look at the next item that will come off the heap simply returns the data in the minimum node, which our heap stores a reference to. This is O(1).
- To union to Fibonacci heaps we just connect their root lists together into a single root list and make the minimum be the smaller minimum of the two. Here the fact that we use a doubly linked circular list comes into play to give us O(1) performance.

Extract Minimum

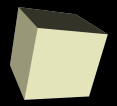
- This is where the real work comes in. Basically the Fibonacci heap has been avoiding work with the other operations and it needs to clean things up a bit here.
- First we act like a binomial heap and add all the children of the min node into the root list while setting their parent pointers to null. Next we remove the minimum node from the root list.
- If our root list is empty we set min to null. Otherwise we set min to the element on the right of the old min and call a consolidate. That is the function that will do the real work. That function in O(log n)

Consolidate

- Obviously at some point we have to bring some of the trees in our forest together. This is where that happens. We begin with an array of node pointers, A, that has $\log_2 n$ elements in it.
- We walk through the root list and check the degree of each root. While A[degree] is not null merge the two trees together preserving heap order. We set A[degree] to null and continue the while loop. After all mergers we set A[degree] to the degree of the node we ended up with.
- Last we walk through our array and set min to point to the root with the smallest value.

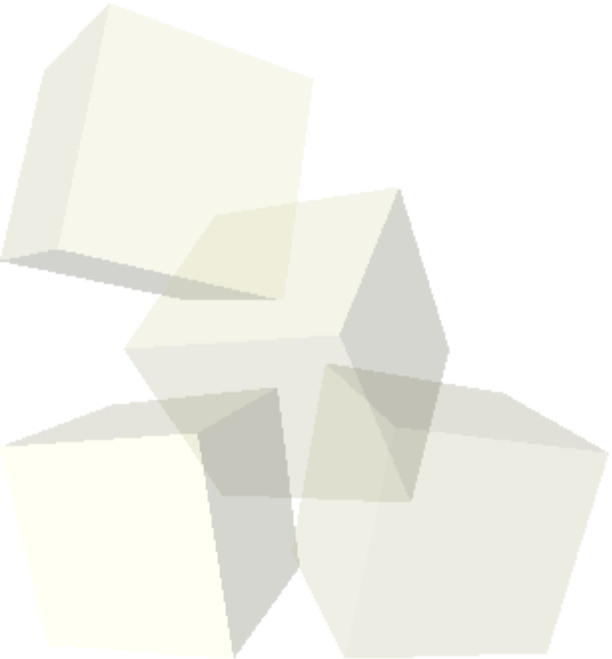
Decreasing Keys and Deleting

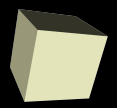
- This is where we get to use our marker. Up to this point we have just set markers to false at creation and every time we make a node a child of a different node.
- To decrease a key we set the new key and see if it has violated heap order. If it has we cut that node out and put it in the root list. We also do a “cascading-cut” on the parent. If the parent is not marked we simply mark it. If it is marked, we cut it and recursively go to its parent. This turns out to be O(1).
- A delete can be done by decreasing the key to a minimum value and then doing extract-min.



Performance in Java

- I spent the weekend doing some Java code and trying to get the most performance out of it as possible. I want to look at some of the subtle changes that were made to try to cut seconds off of a runtime.





Reminders

- Remember to turn in your test on Thursday.
- We will talk about dynamic programming next class.
- I will get you timings for my Fibonacci heaps as soon as possible.