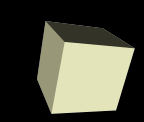
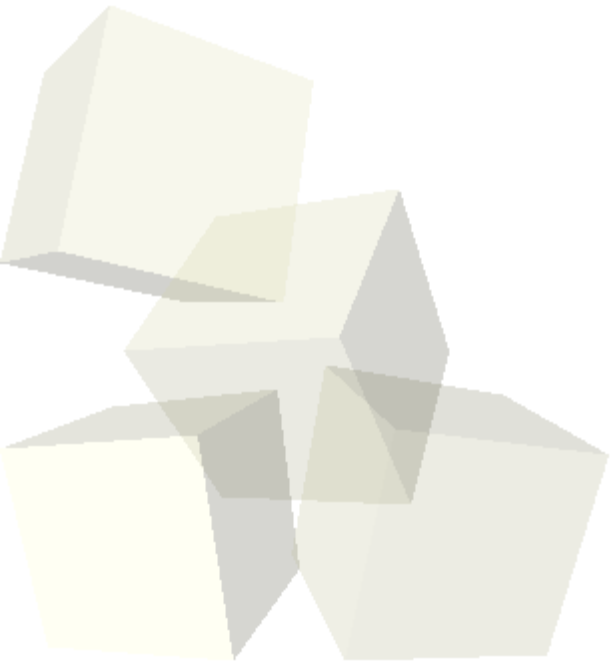
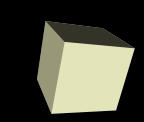


Is it Greedy?

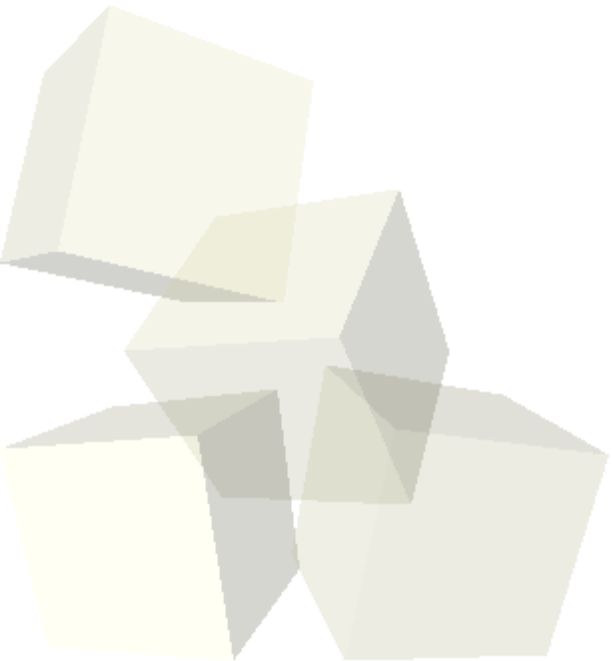
2-9-2006

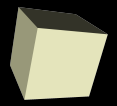




Opening Discussion

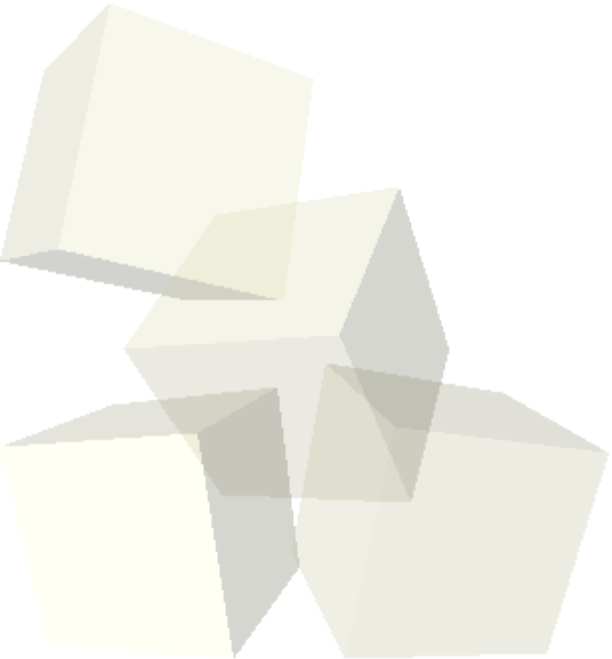
- What did we talk about last class?





Graph Problems

- There are a number of interesting greedy algorithms that we didn't discuss because they fall in the category of graph algorithms. We will be doing graph algorithms later on in the semester and you will see those algorithms then. They include things like minimum spanning tree, shortest path, and clustering.



Developing a Greedy Algorithm

- Greedy algorithms can be developed in different ways. CLR suggests the following steps in the general process of constructing a greedy algorithm.
  - Cast the optimization problem as one in which we make a choice and are left with one subproblem to solve.
  - Prove that there is always an optimal solution to the original problem that makes the greedy choice.
  - Show that once the greedy choice is make we are left with a subproblem with the property such that when we combine an optimal solution of it with the greedy choice we get an optimal solution to the original problem.

Greedy and Dynamic Programming

- CLR covers DP before greedy algorithms because greedy solvable problems are a subset of DP solvable problems. In both cases you need an optimization problem that exhibits optimal substructure. The greedy solvable problems are the ones where you can narrow the number of options at any given step down to one.
- There is no general way to determine if a problem is greedy. Showing a problem has optimal substructure and that is has the greedy choice property are key ingredients.

Greedy-Choice Property

- This is the property that a globally optimal solution can be arrived at by making locally optimal choices.
- When we solve problems with greedy algorithms we are doing a top-down solution. We start with the full problems and make a choice that leads to a particular subproblem of the original problem. We need that subproblem to be part of an optimal solution to the problem as a whole.

Optimal Substructure

- This is something we will come across again with DP. The idea of optimal substructure is that an optimal solution to the entire problem is built from optimal solutions to smaller problems.
- If a problem has optimal substructure then you can find optimal solutions to the subproblems and find the proper combination of the subproblems that gives the optimal solution to the problem as a whole.

Matroid Theory

- We will now take a quick look at a combinatoric theory that can be used to develop greedy solutions to some problems. Unfortunately, it does not work for all problems that have greedy solutions, only a subset of them.
- A matroid is a pair $M=(S,\mathcal{I})$ where S is a finite nonempty set and $\mathcal{I}$ is a nonempty family of subsets of S called the independent subsets.
- If $B\in\mathcal{I}$ and $A\subseteq B$, then $A\in\mathcal{I}$. $\mathcal{I}$ is called hereditary if it satisfies this property.
- M satisfies the exchange property if $A\in\mathcal{I}$, $B\in\mathcal{I}$, and $|A|<|B|$, then there exists $x\in B-A$ such that $A\cup\{x\}\in\mathcal{I}$.

Example Matroids

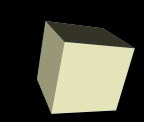
- Matroids were originally part of work on matrices. If S is the rows of a matrix and $\mathcal{I}$ is the set of linearly independent subsets of S you have a matroid.
- Another example of a matroid comes from graph theory. If S is the set of edged on a digraph, then $\mathcal{I}$ can be the set for sets of edges that don't have cycles. This particular matroid leads to the solution of the minimum spanning tree problem.

More Terms

- An extension to an element of the independent sets is an elements that isn't in that set and whose union with the set produces a different element of the independent sets.
- If an element of $\mathcal{I}$ has no extensions is it called maximal.
- All maximal independent subsets of a matroid have the same size.

General Algorithm on a Weighted Matroid

- Start with A and an empty set.
- Sort S from largest to smallest by weight.
- For each x in S do the following
  - If $A \cup \{x\} \in \mathcal{I}$ then $A = A \cup \{x\}$.



Reminders

- You might consider looking at assignment #3.