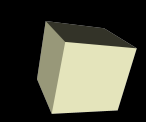
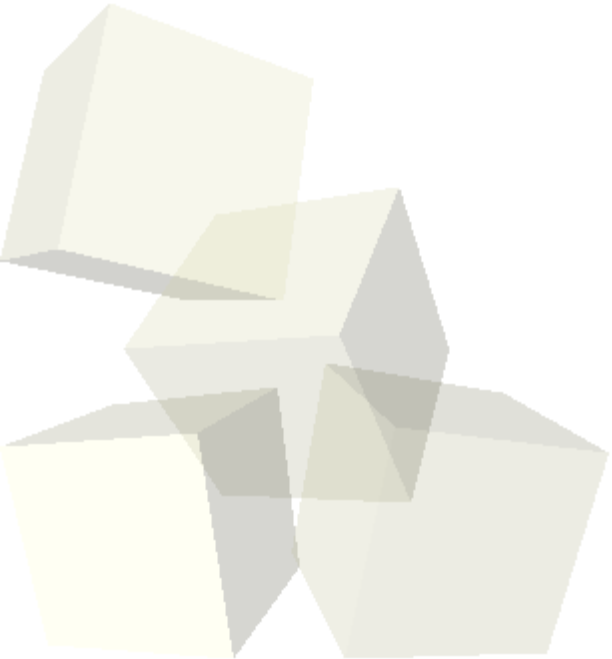


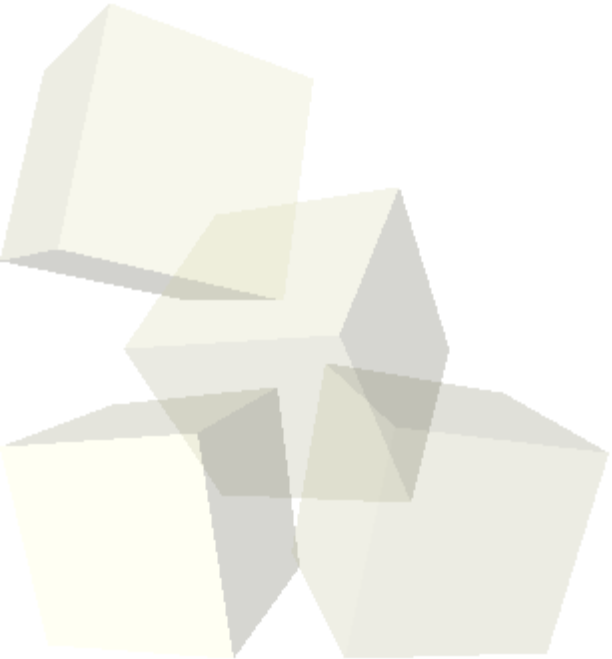
Approximation Algorithms

4-20-2006



Opening Discussion

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



Approximation Algorithms

- As you know, some problems simply aren't tractable for finding exact solutions. In optimization problems we can instead look for approximations.
- An approximation algorithm is most useful when we can prove that the approximation is within a certain range of the truly optimal solution. We describe a p-approximation as one where it finds a solution such that the ratio of the approximation and the actual solutions is no greater than p.
- So a 2-approximation algorithm returns a solution that is within a factor of 2 of the actual solution.

Vertex Covering

- The problem here is that we are given an undirected graph and we want to find the smallest set of vertices such that every edge has at least one vertex in that set.
- Obviously we could find the actual solution in 2^n time with a brute force search.
- A 2-approximation algorithm can be made by consistently taking edges such that neither vertex on the edge is currently in the set and adding their vertices to the set.
- This is provably a 2-approximation because a true solution would have to have at least one vertex from each of the selected edges.

TSP

- You are all familiar with the traveling salesman problem and the fact that brute force takes factorial time while even memoizing requires exponential time and space.
- We can, however, find a polynomial 2-approximation algorithm iff the graph obeys the triangle inequality.
- The algorithm is simple, first build a minimum spanning tree, then walk the tree in preorder and have your circuit include the vertices in the order they are visited.
- The MST can't be longer than the optimal solution and the circuit is smaller than two MSTs.

TSP without Triangle Inequality

- If you don't have the triangle inequality, TSP can only be approximated in polynomial time if P=NP.
- The reason is that you can design an instance of TSP such that the approximation algorithm would be forced to find a Hamiltonian circuit on a general graph.
- If you had such an approximation algorithm you could find Hamiltonian cycles in P time, but the Hamiltonian cycle problem is NP-complete.

Set Covering

- Imagine you have a set, X, along with a set of subsets, F, of that set such that every element of X is in at least one element of F.
- The question is, what is the minimal number of elements of F that are required to “cover” X?
- We can use a greedy method to approximate this where we get a new set each time such that it includes the most uncovered elements of X. This can be done in linear time, but CLRS leaves the exact method as an exercise for the reader.
- This method is a (ln |X| +1)-approximation. It is better for small sets but doesn't get all that bad even for large sets.

Reminders

- I will be handing out your last test on Tuesday. I'm also accepting assignments though you can't get points for speed if you turn it in this late. Make sure to e-mail me telling me you have turned it in so I don't miss it.