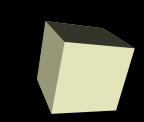
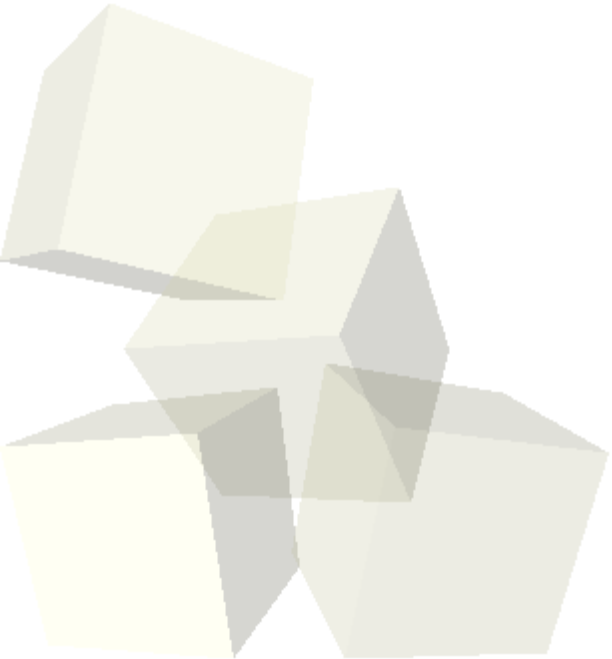


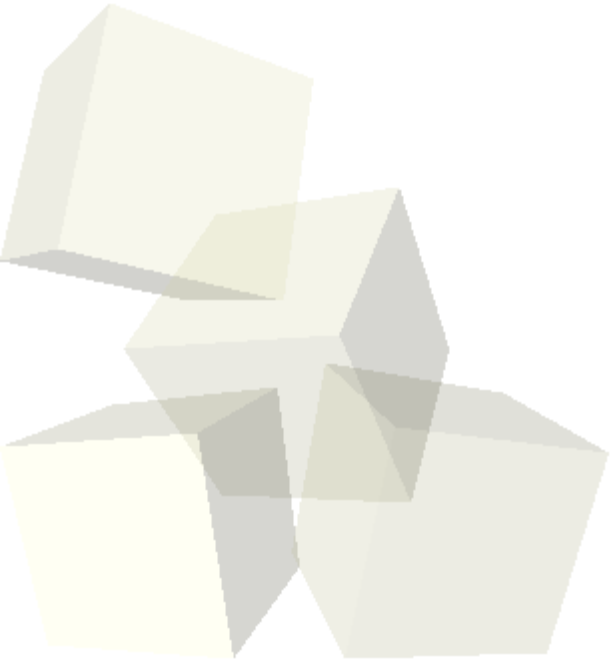
All-Pairs Shortest Path

3-21-2006



Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- Grades have been posted for assignment #1. Hopefully I should be able to get through one or two more on Wednesday.



All-Pairs Shortest Path

- For some problem you want to know the shortest path from all vertices to all other vertices. You could solve this doing a single source shortest path from each vertex, but that duplicates a lot of work because shortest paths have optimal substructure.
- These algorithms compute not only the lengths of the paths, but the paths themselves which are stored as a predecessor matrix. p_{ij} is the parent of j on the shortest path from i.

Matrix Multiplication

- A simple way to calculate shortest paths is with a DP algorithm that mirrors matrix multiplication. The idea is that each step you are finding the shortest path between all vertices that is one "jump" longer than what you had before. The initial adjacency matrix gives the shortest paths if you can only go over one edge.
- We take two matrices and instead of summing the products, we take the minimum of the sum. This gives us a matrix representing one extra jump.
- We can improve performance by doing repeated "squaring" instead of doing "multiply" N times. This gives $O(n^3 \log n)$ performance.

Floyd-Warshall

- This is a different DP solution for the all-pairs shortest path problem. In this representation we grow the number of intermediate vertices that we consider. So we start with the adjacency matrix then find a new matrix that would allow paths that have vertex 1 as an intermediate. Then allow 2 as well and so on.
- So we loop through adding new vertices, k. The recursion looks like this.

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & if\ k = 0 \\ min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & if\ k \geq 1 \end{cases}$$

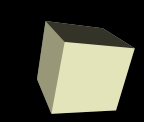
- This gives $O(n^3)$ code calculating from bottom up.

Paths with Floyd-Warshall

- A recursive function can also be defined for getting the parents on the paths using the Floyd-Warshall method. The idea is that the parent matrix starts as being nil if there is no path from i to j or i if there is an edge from i to j.
- At each step we see if the intermediate point is used. If not, the parent node stays the same as it had been for the earlier k. If so, then the parent is the parent on the path of k to j.

Johnson's Algorithm

- This algorithm doesn't use an adjacency matrix and is better for sparse matrices. It uses the single source shortest path algorithms that we discussed previously. The trick is that if there are negative edges it will change the weight on the edges so they are positive and it is safe to use Dijkstra's algorithm.



Reminders

- Remember that assignment #5 is due today.