Simple Optimization on Graphs

3-9-2006

Opening Discussion

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

Minimum Spanning Trees

- Last time I did the brief overview of a greedy algorithm. Now we want to look at the specifics.
- What I outlined last time is Kruskal's algorithm. We treat the graph as a forest and each iteration add the shortest edge that connects two separate trees. An efficient implementation uses disjoint sets and sorts the edges from smallest to largest. O(E log V)
- Prim's algorithm grows a single tree and picks the minimum edge that connects a new vertex to the tree. An efficient implementation keeps a heap of the vertices with the order determined by the shortest edge to the tree. O(E log V) or O(E+V log V)

Single Source Shortest Path

- Shortest path has optimal substructure so you know it can be solved DP and might be solvable greedy.
- No solution to this includes a cycle. If a graph has a cycle of negative length then the shortest path is not well defined. All positive cycles would be eliminated and zero length cycles can be cut out WLOG.
- A critical question is whether you allow negative weight edges in your graph. The answer to that determines what algorithm you have to use.

Initialization and Relaxation

- All the algorithms we will talk about keep distance estimate, d, and a parent, p, for each vertex.
- We initialize the distance estimate with infinity for all nodes except the source which gets a value of 0. All parents are set to nil.
- We can relax node v w.r.t u by checking if d(v)>d(u)+w(u,v). If it is, we alter the weight and make p(v)=u.

Bellman-Ford

- This algorithm works on general weighted, directed graphs. Negative edges are allowed.
- After initializing, this algorithm runs through all edges in the graph V-1 times relaxing the nodes for each edge in turn.
- A check is then done to see if there are any negative cycles. That will happen if any edge could be relaxed again.
- This algorithm is O(VE).

DAGs

- For a directed acyclic graph we can use a simpler algorithm.
- First to a topological sort of the DAG. (This takes O(V+E) time.)
- Then initialize and run through the sorted list of vertices. Relax every edge coming out of each vertex in turn.
- The full runtime for this algorithm is also O(V+E).

Dijkstra

- If all the weights are non-negative we can use a superior, greedy algorithm.
- Here we keep a set S (that is a subset of V) of vertices we know the minimum distance to. It begins as the empty set. We also keep a min queue of the vertices sorted by their current distance elements.
- While the queue isn't empty we pull of the minimum element and add it to S. Then we relax all the edges coming out of that new vertex. Note the queue must be updated to reflect this relaxation.
- If a Fibonacci heap is used this runs in O(V log V+E) time.

Reminders

- Enjoy your spring break. Assignment #5 is due when you get back so you might want to practice some coding while you off having fun. Then again, what could be more fun than coding?