Graph Theory II

Senior Computer Team

November 19, 2004

1 Bellman-Ford

The Bellman-Ford algorithm is another shortest path algorithm. Given a graph with V nodes and E edges, it will find shortest paths from one node (source) to all other nodes. It is an $O(V * E)$ algorithm that works on graphs with negative edges but not negative cycles. In addition, it is significantly easier to code than Dijkstra's algorithm.

There are two key ideas you should note. One is that a shortest path will not contain more than V-1 edges in the graph, assuming there is no negative cycle. If there are more than V-1 edges in the shortest path, some node will have been visited twice. This is not optimal. The other key idea is DP. In each iteration, you consider whether using a specific edge gives a better distance to the endpoints or not. Think of the shortest path as being necessarily composed of at most V-1 edges, which are iteratively put into place.

```
Bellman-Ford(a graph with V vertices and E edges)
{
   bestdist = 0 for source, INF for all others
   for 1 to V-1
      for edges i=1 to E
         if bestdist[startpt[i]] + weight[i] < bestdist[endpt[i]]
            bestdist[endpt[i]]= bestdist[startpt[i]] + weight[i]

         //do this step only if the graph is undirected
         if bestdist[endpt[i]] + weight[i] < bestdist[startpt[i]]
            bestdist[startpt[i]]= bestdist[endpt[i]] + weight[i]
}
```

2 Minimum Spanning Tree

There are two main Minimum Spanning Tree (MST) algorithms, namely Prim's algorithm and Kruskal's algorithm; both are based on the greedy algorithm. As Prim's algorithm is

easier to code, that is preferred. In sparse graphs with a large number of nodes but relatively few edges, Kruskal's algorithm may provide a better alternative.

2.1 Kruskal's algorithm

Kruskal's idea is that we first sort the connection edges in increasing costs. Then we go through them in that order and see if using an edge is good or not. So the Greedy here is that if an edge does not create a cycle and the edge costs less than other edges, then use it. And it works in $O(ElogE)$ time. However, coding this algorithm is somewhat difficult.

```
Kruskal(a graph with V vertics and E edges)
{
   sort edges in increasing cost

   for each edge (1->V)
      if this edge does not create cycle
         then use this edge
}
```

3 Problems

Problem 1: Amazing Barn (abridged) [USACO Competition Round 1996]
Consider a very strange barn that consists of N stalls (N $\leq$ 2500). Each stall has an ID number. From each stall you can reach 4 other stalls, but you can't necessarily come back the way you came.

Given the number of stalls and a formula for adjacent stalls, find any of the 'most central' stalls. A stall is 'most central' if it is among the stalls that yields the lowest average distance to other stalls using best paths.

Problem 2: Railroad Routing (abridged) [USACO Training Camp 1997, Contest 1]
Farmer John has decided to connect his dairy cows directly to the town pasteurizing plant by constructing his own personal railroad. Farmer John's land is laid out as a grid of one kilometer squares specified as row and column.

The normal cost for laying a kilometer of track is \$100. Track that must gain or lose elevation between squares is charged a per-kilometer cost of \$100 + \$3 x (meters of change in elevation). If the track's direction changes 45 degrees within a square, costs rise an extra \$25; a 90 degree turn costs \$40. All other turns are not allowed.

Given the topographic map, and the location of both John's farm and the plan, calculate the cost of the cheapest track layout.