

PRISM: Summer Discovery Experience, May 2010

Topic 3: Building a tree

Christopher King

Department of Mathematics

April 26, 2010

Many applications of graphs involve trying to find the shortest walk between two nodes on a connected graph G where every edge has a weight (or metric) assigned to it. Denote by $d(v, w)$ the weight of the edge (v, w) . The *length* of a walk is the sum of the weights of edges along the walk. See Figure 15 for an example.

Such an exhaustive listing is useless for large graphs, hence the need for an efficient algorithm. Dijkstra invented an algorithm that finds the shortest walks from a given node v to all other nodes. The idea is to sample all possible walks starting from v , and to recursively compute the shortest ones. Each node w is labeled with an estimate $L(w)$ of the shortest walk length to v . Initially, this estimate is just the weight of the edge $d(v, w)$, or else $+\infty$ if there is no such edge. When the estimate becomes certain, we regard the node as being *permanently labeled* and keep track of this with a set P of permanently labeled nodes. The node added to P at each step will be the closest to node v out of those that are not yet in P .

Here is the algorithm. Initially, $P = \{v\}$, $L(v) = 0$ and $L(w) = d(v, w)$ for all $w \neq v$.

Step 1 (Find the next closest node) Find $w \in P$ such that

$$L(w) = \min\{L(u) : u \notin P\}$$

Set $P = P \cup \{w\}$. If P contains all the nodes, then stop: the algorithm is complete.

Step 2 (Updating of labels) For all $u \notin P$ set

$$L(u) = \min\{L(u), d(w, u) + L(w)\}$$

Go to Step 1.

Problem 1: apply Dijkstra to the graph in Figure 15.

You should end with a tree subgraph of G , and it should give you the shortest walks from node a to all other nodes.

There is a related famous problem called the *traveling salesman* problem, namely to find a shortest possible walk that visits each node exactly once.

Problem 2 (open): solve the traveling salesman problem for Figure 15.
