CSE 202 NOTES FOR NOVEMBER 12, 2002

Data Structures in Algorithms

We will revisit Dijkstra's Single-Source Shortest Path algorithm. I won't relist the algorithm here.

In the most naive implementation of the algorithm we execute $O(m)$ operations per iteration, and $O(mn)$ iterations, if we are given only the adjacency matrix, so the algorithm is roughly $O(m^2n)$. Instead, maybe we can use the adjacency list, but this begins to get at a set of questions that we haven't really tackled before. To make a decision on what kind of data structure we should use, we need to determine how we're going to use the data structure. This can be determined by asking two basic questions: how do we query the data structures?, and how will the data structures be updated during the algorithm?

How do we query the data structures? In our algorithm, we need S, the set of nodes with computed distance to each, and Σ, the set of edges with a numerical field (the "cost" of the edge). We need to query the S data structure: is $u \in S$? We will also need to compute $d(u)$ easily. For Σ, we need to determine what the minimum cost element is quickly.

How will the data structures be updated during the algorithm? For S, we will need to add a single node. For Σ we will need to add edges outbound from v, and we will also need to remove edges inbound to v (when we incorporate a node v into S, we have to adjust the edges in Σ accordingly.

Clearly we want data structures that will give us the best performance; it seems that two candidate data structures would be an array of boolean values for S and a heap (augmented with the appropriate data) for Σ. In this case, since finding mins iz $O(1)$ and deleting/inserting is $O(\log m)$ we get that the total time of Dijkstra's algorithm is $O(m \log n) = O(|E| \log |V|)$. This seems a little confusing at first, since Σ holds edges, and you'd expect that this algorithm would run in $O(m \log m)$ time, but I think that Σ never holds $|E|$ edges, but only $|V|$ edges as it builds the spanning tree. Thus, even though Insert is called m times, it will only hold at most n elements, giving $m \log n$ as an upper bound on the time. Note that you can't just assume that since Insert is called m times that Σ will have m elements in it, since Delete is also called m times, and Σ doesn't have 0 elements in it.

Table 1. The number of times each operation is performed during Dijkstra's algorithm. (n is $|V|$ and m is $|E|$)

Operation	Number of Times
In S	$2m \in O(m)$
Insert S	n
FindMin Σ	n
Delete Σ	m
Insert Σ	m