Cosc-320
Single-Source Shortest Path in Directed Graphs
November 29, 2003

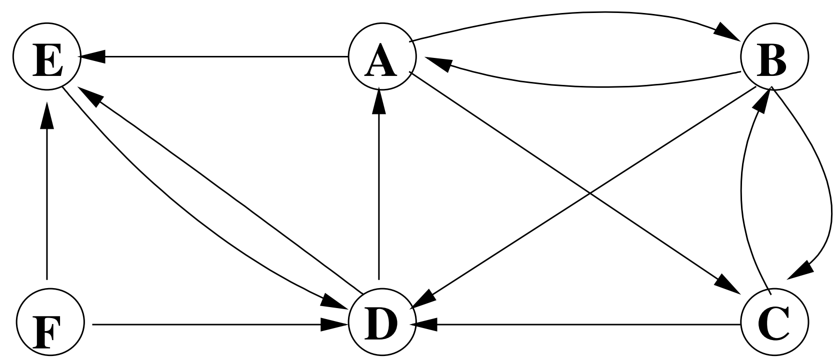
Introduction

These notes on a shortest path algorithm expound on Chapter 16 of "Data Structures with C++ using STL 2nd Edition," William Ford and William Topp, Prentice-Hall, 2002, Section 16.6.)

$\diamond$ **Definition**: The *single-source shortest path* problem is to find the shortest path (minimum number of edges) from a given starting vertex, V_s, to every other vertex in a directed graph $G(V, E)$.

• Note: The text calls this *shortest-path* leaving off the "single-source" part. The text also does not directly find the shortest-path to every vertex, it finds the shortest path from V_s to some other designated vertex.

• For an example, we use Figure 16-22 from the text, reproduced here.



Text Figure 16–22

The Algorithm

First, construct a two-dimensional table, T. The rows of T are indexed by the vertices in the graph. T has two columns, one called `DIST` and the other called `VERT`.

$T[V]$[DIST] holds the distance (number of edges) from V_s to V. $T[V]$[VERT] holds the vertex from which the algorithm arrived at V.

Initially, for each vertex, V, $T[V][\text{DIST}] = \infty$ (except for the starting vertex, $T[V_s][\text{DIST}] = 0$) and $T[V]$[VERT] is undefined (except $T[V_s][\text{VERT}] = V_s$).

After choosing $V_s = F$ in the example graph, the initialized table will be:

Vertex	DIST	VERT
A	∞	
B	∞	
C	∞	
D	∞	
E	∞	
F	0	F

Now, construct an empty queue of vertices, Q and push V_s onto Q.

Finally, run the following loop:

```
while (Q is not empty)
{
  V = Q.top();
  Q.pop();

  for (each neighbor, W, of V)
    if (T[W][DIST] == INFINITY)
      {
        T[W][DIST] = T[V][DIST] + 1;
        T[W][VERT] = V;
        Q.push(W);
      }
}
```

After running the algorithm on the graph in Figure 16-22 from the text, table T will be:

Vertex	DIST	VERT
A	2	D
B	3	A
C	3	A
D	1	F
E	1	F
F	0	F

The length of the shortest path from V_s (namely vertex F) to a vertex V is found in $T[V]$[DIST]. Thus, the shortest path from vertex F to vertex C has path length of 3. The shortest path from F to A has length of 2.

The table can also be used to determine the path in each case. To find the shortest path from V_s to V, start at $T[V][\mathrm{VERT}]$ and work toward V_s. To reconstruct the shortest path from F to C, observe that we got to C from A; to A from D; to D from F. Thus the path is F-D-A-C.

Performance

Setting up the table and queue is in $O(1)$. Accessing table and queue elements is also in $O(1)$ and is done a maximum of $|V|$ times. The loop is a breadth-first traversal of the graph which is in $O(|V| + |E|)$. Thus, the single-source shortest path algorithm is in $O(|V| + |E|)$.