Announcements

prelab 10 is due on Monday. There are parts of it you won't be able to do until after Monday. Hand those in next Friday. I will specify which parts are such on the lab itself.

Friday, November 30, 12

Ex: unweightedSP( A ) will determine the s.p. from A to B,C, and D.

A

B

C

D

A.dist

0

B.dist

1

C.dist

2

D.dist

2

A.prev

null

B.prev

A

C.prev

B

D.prev

C

A.prevB.prevC.prevD.prev

nullABB

Friday, November 30, 12

Algorithm Details

unweightedSP( Vertex s )

• Let S be the set of “explored” nodes, initially S = { s }
• for each node v (including s) set v.prev = null, v.dist = infty
• initialize queue Q to contain s, set s.dist = 0
• while Q is non-empty
• let v be the front element of Q
• add v to S (mark it as “explored”)
• for each of v’s unexplored neighbors w
• if w is not already in S
• set w.prev = v
• set w.dist = v.dist + 1
• add w to Q

printPath( Vertex w )

• if w.dist == infty, print “unreachable”
• else if (w.prev != null)
• printPath( w.prev )
• print “ ---> w ”

RT: (vague) for loop is O(n)
while loop is O(n)
therefore O(n^2)

RT: (precise) for loop is O(deg(v))
and so O(m+n)

RT: just O(n)

S = ∅
Input: a graph $G=(V,E)$ with positive edge costs and a start vertex $s$.
Goal: determine the cost of the least-cost path from $s$ to every vertex in $V$ (and while you're at it, compute the path itself).

Q: Will unweightedSP (aka bfs) still work?
A: Sadly, no.

- Need to adjust the order in which we remove vertices from the queue.
- Rather than removing first-in, want to remove closest-to-$s$.
- We'll need to adjust `w.dist` many times, as we find shorter distances.

**Idea:** weightedSP($s$)

- Explore vertices out of $s$ at a uniform rate.
- Think of edge costs as lengths of pipe, and follow water flowing out of $s$.
- We'll explore vertices in the order the water hits them.
- Once we explore a vertex, we've found its shortest path.
- As we explore each vertex, we can see new paths to its neighbors that may or may not be better than the paths we already know about.

**Dijkstra's Algorithm**

1. Let $S$ be the set of "explored" nodes, initially empty.
2. For each node $v$ (including $s$), set $v.pred = null$ and $v.dist = \infty$.
3. Initialize a priority queue $Q$ to contain $s$, and set $s.dist = 0$.
4. While $Q$ is non-empty:
   1. Let $v$ be the vertex with minimum $dist$ in $Q$.
   2. Add $v$ to $S$ (mark it as "explored").
   3. For each neighbor $w$ of $v$ that is unexplored:
      1. If $v.dist + \text{cost}(v, w) < w.dist$:
         1. Set $w.prev = v$.
         2. Set $w.dist = v.dist + \text{cost}(v, w)$.
         3. If $w$ is not in $Q$, add it to $Q$.
         4. Else, decrease the key of $w$ in $Q$.

**Example**

Let's initialize the set of "explored" nodes, initially empty $S = \emptyset$. For each node $v$ (including $s$), set $v.pred = null$ and $v.dist = \infty$.

```
S = \emptyset
s.dist = 0
```

Let $Q$ be the priority queue of vertices. A vertex's distance is its priority.

We'll explore vertices out of $s$ at a uniform rate. Think of edge costs as lengths of pipe, and follow water flowing out of $s$. Once we explore a vertex, we've found its shortest path.

As we explore each vertex, we can see new paths to its neighbors that may or may not be better than the paths we already know about.
Proof of Correctness

A set of explored nodes $S$ is either

- empty, or
- $S' \cup \{v\}$, where $S'$ is a set of explored nodes and $v$ is the last vertex added.

For a set $S$ of explored nodes, let $P(S)$ denote the property that

- for each $u \in S$, the cost of the shortest $s$-$u$ path is $u$.dist, and the preceding node along this path is $u$.prev.

We will show $P(S)$ is true for all sets of explored nodes, by induction on $S$.

**Step 0:** try it on some examples (see the previous slide to check.)

**Step 1:** base case. Let $S = \emptyset$. Then $P(S)$ is vacuously true. That was easy.

**Step 2:** inductive step. Let $S' \cup \{v\}$, where $v$ is the last vertex added at this point. For the induction hypothesis, suppose $P(S')$ is true, that is, for all $u \in S'$, the cost of the shortest $s$-$u$ path is $u$.dist, and the preceding node along this path is $u$.prev. We want to show that the cost of the shortest path to $v$ is $v$.dist, and the preceding node is $v$.prev.

(Why do we care? Well, if $P(S)$ is true for all sets of explored nodes, then $P(V)$ is true for the set of all the vertices $V$.)

1. Connect shortest $s$-$v$ path to shortest $s$-$u$ path for some $u \in S'$. The shortest $s$-$v$ path $P$ starts at $s$ in $S$, and ends at $v$. (Note: it's possible that $\notin S'$ or $x \notin S'$, that's ok.)

2. Connect $v$.dist to $u$.dist for some $u \in S'$.

Let $x$ be the first vertex along $P$. Let $y$ be the vertex on $P$ that precedes $x$ in $S$. Then $x = x$.dist + cost(x,y) + cost(y,v) for all $y \in S$, where $y \neq v$.

By our algorithm, $v$.dist $\leq$ $y$.dist for all $y \in S$, where $y \neq v$.

So there exists some vertex $x \in S$ along $P$ such that $x = x$.dist + cost(x,y) + cost(y,v) for all $y \in S$, where $y \neq v$.

The shortest $s$-$v$ path $P$ starts at $s$ in $S$, and ends at $v$. (Note: it's possible that $\notin S'$ or $x \notin S'$, that's ok.)

By our algorithm, $v$.dist $\leq$ $y$.dist for all $y \in S$, where $y \neq v$.

By def of $v$.dist, $v$.dist $\leq y$.prev.dist + cost(y.prev,y) for all $y \in S'$, where $y \neq v$.

So the property $P$ holds for $S' \cup \{v\}$.

By induction, $P(S)$ is true for all sets of explored nodes $S$. By induction on $S$.