Weighted graphs

Your mother is so fat,
even I cannot find the shortest path around her.
Weighted graph

Edges in weighted graph are assigned a weight: $w(v_1, v_2)$, where $v_1, v_2$ in $V$

If path $p = <v_0, v_1, ..., v_k>$ then the weight is: $w(p) = \sum_{i=1}^{k} (v_{i-1}, v_i)$

Shortest Path:
$\delta(u,v): \min\{w(p) : v_0 = u, v_k = v\}$
Shortest paths

Today we will look at single-source shortest paths.

This finds the shortest path from some starting vertex, s, to any other vertex on the graph (if it exists).

This creates $G_{\pi}$, the shortest path tree.
Shortest paths

Optimal substructure: Let $\delta(v_0, v_k) = p$, then for all $0 \leq i \leq j \leq k$, $\delta(v_i, v_j) = p_{i,j} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$

Proof?

Where have we seen this before?
Shortest paths

Optimal substructure: Let $\delta(v_0, v_k) = p$, then for all $0 \leq i \leq j \leq k$, $\delta(v_i, v_j) = p_{i,j} = <v_i, v_{i+1}, \ldots, v_j>$

Proof? Contradiction!
Suppose $w(p'_{i,j}) < p_{i,j}$, then let $p'_{0,k} = p_{0,i} \cdot p'_{i,j} \cdot p_{j,k}$, then $w(p'_{0,k}) < w(p)$
Shortest path

We will do the same thing we have done before with BFS and DFS:

Makes a queue and put in/pull out

Two major differences:
(1) How to remove from queue (min)
(2) Update “grey” vertexes (‘‘relax’’)

Relaxation

We will only do relaxation on the values $v.d$ (min weight) for vertex $v$

$$\text{Relax}(u,v,w)$$

if($v.d > u.d + w(u,v)$)

\[v.d = u.d + w(u,v)\]

$v.\pi = u$
Relaxation

We will assume all vertices start with $v.d=\infty, v.\pi=\text{NIL}$ except $s$, $s.d=0$

This will take $O(|V|)$ time

This will not effect the asymptotic runtime as it will be at least $O(|V|)$ to find single-source shortest path
Relaxation

Relaxation properties:
1. \( \delta(s,v) \leq \delta(s,u) + \delta(u,v) \) (triangle inequality)
2. \( v.d \geq \delta(s,v) \), \( v.d \) is monotonically decreasing
3. if no path, \( v.d = \delta(s,v) = \infty \)
4. if \( \delta(s,v) \), when \( (v.\pi).d=\delta(s,v.\pi) \) then
   relax\((v.\pi,v,w)\) causes \( v.d=\delta(s,v) \)
5. if \( \delta(v_0,v_k) = p_{0,k} \), then when relaxed in
   order \((v_0, v_1), (v_1, v_2), ... (v_{k-1},v_k)\) then
   \( v_k.d=\delta(v_0,v_k) \) even if other relax happen
6. when \( v.d=\delta(s,v) \) for all \( v \) in \( V \), \( G_\pi \) is shortest
   path tree rooted at \( s \)
Directed Acyclic Graphs

DFS can do topological sort (DAG)

Run DFS, sort in decreasing finish time
Directed Acyclic Graphs

DAG-shortest-paths(G,w,s)
topologically sort G
initialize graph from s
for each u in V in topological order
  for each v in G.Adj[u]
    Relax(u,v,w)

Runtime: $O(|V| + |E|)$
Depth first search
Directed Acyclic Graphs

Correctness:

Prove it!
Correctness:
By definition of topological order, When relaxing vertex v, we have already relaxed any preceding vertices
So by relaxation property 5, we have found the shortest path to all v
BFS (unweighted graphs)

Create FIFO queue to explore unvisited nodes
Dijkstra's algorithm is the BFS equivalent for non-negative weight graphs.
Dijkstra

Dijkstra(G, w, s)
initialize G from s
Q = G.V, S = empty
while Q not empty
    u = Extract-min(Q)
    S = S U {u}
    for each v in G.Adj[u]
        relax(u, v, w)

S optional
Dijkstra

Runtime?
Dijkstra

Runtime:
Extract-min() run $|V|$ times
Relax runs Decrease-key() $|E|$ times
Both take $O(lg n)$ time

So $O((|V| + |E|) lg |V|)$ time
(can get to $O(|V|lg|V| + E)$ using Fibonacci heaps)
Dijkstra

Runtime note:
If $G$ is almost fully connected, 
$|E| \approx |V|^2$

Use a simple array to store $v.d$

$\text{Extract-min}() = O(|V|)$

$\text{Decrease-key}() = O(1)$

total: $O(|V|^2 + E)$
Correctness: (p.660)

Sufficient to prove when \( u \) added to \( S \), \( u.d = \delta(s,u) \)

Base: \( s \) added to \( S \) first, \( s.d=0=\delta(s,s) \)

Termination: Loop ends after \( Q \) is empty, so \( V=S \) and we done
Dijkstra

Step: Assume v in S has v.d = δ(s,v)
Let y be the first vertex outside S on path of δ(s,u)

We know by relaxation property 4, that δ(s,y)=y.d (optimal sub-structure)

y.d = δ(s,y) ≤ δ(s,u) = u.d, as w(p)≥0
Dijkstra

Step: Assume $v$ in $S$ has $v.d = \delta(s,v)$
But as $u$ was picked before $y$,
$u.d \leq y.d$, combined with $y.d \leq u.d$

$y.d = u.d$

Thus $y.d = \delta(s,y) = \delta(s,u) = u.d$