Lecture in a minute

Dijkstra's Direct Inductive Proof.
Know distance to $S$.
Inv: $d(v) -$ length of path through $S$.
Smallest $d(v)$ is correct, add to $S$.
Updating neighbors of $v$ enforces Inv.

Bellman-Ford: Negative weights.
Dijkstra doesn't work.
Update edge $(u,v)$: $d(v) = \min(d(v), d(u) + l(u,v))$.
Update all edges $|V| - 1$ times.
Paths of length $k$ correct after iteration $k$.

DAG:
linearize and process vertices in order.
Updates of edges in order along path.

Update/Bellman-Ford.

Dijkstra's Algorithm.

foreach $v$:
$d(v) = \infty$.
$d(s) = 0$.
Q.Insert($s,0$)
While $u = Q.DeleteMin()$:
foreach edge $(u,v)$:
if $d(v) > d(u) + l(u,v)$:
$d(v) = d(u) + l(u,v)$
Q.InsertOrDecreaseKey($v,d(v)$)

Runtime:
$|V|$ DeleteMins.
$|V|$ Inserts.
$\leq |E| \text{ DecreaseKeys}.$
Binary heap: $O((|V| + |E|) \log |V|)$

Compared to Dijkstra:

Dijkstra:
"Know distance to processed nodes, $R$.
"Add node $v$ closest to $s$ outside of $R$.

Closest node $v$ has path...

$u$ in $R$. Since $v$ is closest node not in $R$.
$d(u)$ corresponds to length of a shortest path.
$d(v) \leq d(u) + l(u,v)$. Since $u$ was processed by Algorithm.
$d(v)$ corresponds to length of path.
Set by some $u$, which corresponds to path by induction plus an edge $(u',v)$.
Thus, when $v$ is added to $d(v)$ is correct.
Corresponds to the length of the shortest path.

Alt Proof.

Dijkstra:
"Know distance to processed nodes, $R$.
"Add node $v$ closest to $s$ outside of $R$.

Closest node $v$ has path...

$u$ in $R$. Since $v$ is closest node not in $R$.
$d(u)$ correct by induction.
$d(v) \leq d(u) + l(u,v)$. Since $u$ was processed by Algorithm.
$d(v)$ corresponds to length of path.
Set by some $u$, which corresponds to path by induction plus an edge $(u',v)$.
Thus, when $v$ is added to $d(v)$ is correct.
Corresponds to the length of the shortest path.

Negative edges.

Notice: argument for Dijkstra breaks for negative edges.

For example.

Dijkstra:
Process $s$: Set $d(a) = 1, d(b) = 2, d(c) = 3$.
Process $a$, $d(a) = 1$: No outgoing edges.
Process $b$, $d(b) = 2$: $d(a)$ still set to 1.
Process $c$, $d(c) = 3$: Set $d(b) = 0$.
But, can't process $b$, again!!!
$d(a)$ still 1. Should be 0.
Problem: $d(b)$ was incorrect when processed due to negative edge.

Update/Bellman-Ford.

def update ((u,v)):
dist(v) = min (dist(v), dist(u) + l(u,v)).

In Dijkstra: Process closest unprocessed node, update neighbors.
Properties:
1) $d(v)$ is correct if $u$ is second to last node on shortest path, and $d(u)$ correct.
2) Never makes $d(v)$ too small. Harmless..

If update $(s, u_1), \ldots (u_{k-1}, f)$, then $(u_k, u_{k+1}) \ldots$ and finally $(u_0, f)$. It's all good!
How??? Update! Here, there, everywhere...

Bellman-Ford:
do $n - 1$ times,
update all edges.

Correctness: After $i$th loop, $d(v)$ is correct for $v$ with $i$ edge shortest paths.
Time: $O(|V||E|)$
**Example:**

Careful: Negative Cycles.

Bellman-Ford: $d(v) \leq \text{length of \ shortest \ path.}$

Assumes length of shortest path is at most $n - 1.$

Why? No cycles!

Why? Cycle only adds length $\text{????.}$

Not necessarily with negative edges.

When won’t it?

If there is a negative cycle.

After $n$ iterations, some distance changes, there must be negative cycle!

---

Example: negative cycle

- $S$ to $A$
- $D$ to $B$
- $C$

---

Property on cycle.

For negative cycle, $C$ where $l(C) < 0,$ where $v \in C,$ $d(v) < \infty,$ there exist edge $(u,v) \in C,$ where $d(v) > d(u) + l(u,v).$

If not, $d(v) \leq d(u) + l(u,v)$ or $d(v) - d(u) \leq l(u,v)$ for all edges in the cycle.

Thus, $\sum_{(u,v) \in C} d(v) - d(u) \leq \sum_{e \in C} l(u,v) = l(C) < 0.$

But, $\sum_{(u,v) \in C} d(v) - d(u) = 0$ since each vertex appears once positively and once negatively in the sum.

Contradiction.

Negative cycle $\implies$ update after $n$ iterations of Bellman-Ford.

---

**DAG**

Dijkstra: Directed graph with positive edge lengths.

$O((m + n) \log n)$ time or a bit better.

Bellman-Ford: Directed graph with arbitrary edge lengths.

$O(nm)$ time.

Also $O(nm)$ time to detect negative cycle.

Directed acyclic graphs?

Negative Cycle? Possible? Not possible?

Not possible. No cycles at all!

DAG:

Remember ...the Alamo!

Remember ...linearization. Inverse post-ordering!

Remember ...Goliad!

Remember ...updating along path makes it all good.

Shortest path for DAG:

linearize/topological sort

process nodes (and update neighbors in order.)

---

DAG

Every path looks like this in a topological order.

The vertices (and edges) along path are processed in order.
Process $s$, $d(s) = 0$: Updates $d(c) = 3$, $d(b) = 2$, $d(a) = 1$.
Process $c$, $d(c) = 3$: Update $d(b) = 0$.
Process $b$, $d(b) = 0$: $d(a) = 0$.
Process $a$, $d(a) = 0$.
Done.

Dijkstra’s Direct Inductive Proof.

Know distance to $S$.

Inv: $d(v)$ - length of path through $S$.

Smallest $d(v)$ is correct, add to $S$.

Updating neighbors of $v$ enforces Inv.

Bellman-Ford: Negative weights.

Dijkstra doesn’t work.

Update edge $(u, v)$: $d(v) = \min(d(v), d(u) + l(u, v))$.

Update all edges $|V| - 1$ times.

Paths of length $k$ correct after iteration $k$.

DAG:

Linearize and process vertices in order.

Updates of edges in order along path.

Good Weekend!