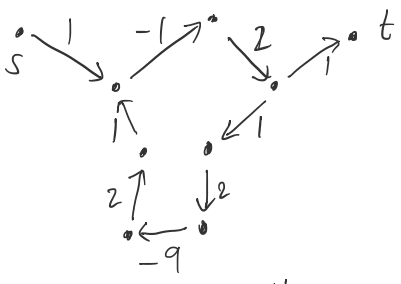
Lec11-Bellman-Ford

KT 6.8

Tuesday, September 19, 2023 9:10 PM

What about negative edges in shortest paths?



Problem: Given a directed graph with weighted edges $d(u,v) \in \mathbb{R}$ that may be pos. or neg., find the shortest path from s to t.

Complication: Negative cycles allow for arbitrarily low-cost paths

Solution idea: Add a big number to all edges to make them positive.

Problem: just turns the problem into shortest unweighted path, since number of edges dominates.

$$cost(P) = M \times hops(P) + cost(P)$$

Modified problem: Given a directed graph with weighted edges $d(u,v) \in \mathbb{R}$ that may be pos. or neg., either

(1) determine there is a negative cycle

or (2) find the shortest path from s to t.

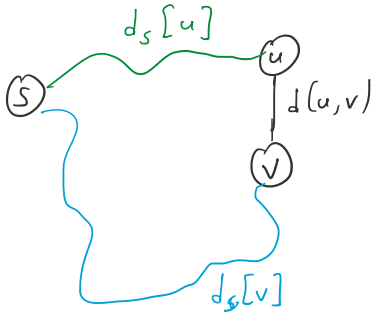
Bellman-Ford: Let $d_s[v]$ be the current estimated distance from s to v.

At start, $d_s[s] = 0$ and $d_s[v] = \infty \quad \forall v \neq s$.

Relaxation step (Ford step):

Find an edge (u,v) s.t. $d_s[u] + d(u,v) < d_s[v]$.

Set $d_s[v] = d_s[u] + d(u,v)$.



Thm If you cannot relax (via Ford step), then $d_s[u]$ = shortest path distance from s to u for all u.

After ... step, either $d_s[v] = \infty$ or $\exists\ s \to v$ path of length $d_s[v]$

distance from s to u for all u.

proof. **Lemma 1:** After any step i, either $d_s[v] = \infty$ or $\exists\ s \to v$ path of length $d_s[v]$.

proof. If $d_s[v] = \infty$, can't say anything at step i.

Let v be a vertex s.t. $d_s[v] < \infty$.

Proof by induction on i.

Base case: When $i = 0$, $d_s[s] = 0 < \infty$ and $\exists$ path of length 0 to s.

Induction hypo: Assume true steps $< i$.

Ind. step: In step i, we update distance using some edge (u,v) where $d_s[v] > d_s[u] + d(u,v)$. (setting $d_s[v] = d_s[u] + d(u,v)$)

$\Rightarrow d_s[u] < \infty$, so $d_s[u]$ was updated at an earlier step.

$\Rightarrow \exists$ path P_{su} of length $d_s[u]$ from $s \to u$.

$\Rightarrow P_{su} + (u,v)$ gives a path $s \to v$ with length new $d_s[v]$.

□

Lemma 2: When no more Ford steps possible, $\forall$ path P_{sv} from $s \to v$, length $(P_{sv}) \geq d_s[v]$. (we get the shortest path)

proof. By induction on #edges in P_{sv}.

Base case: When $|P_{sv}| = 1$, only single edge (s,v), and since can't do Ford step, $d(s,v) \geq d_s[v]$

Induction hypo: Assume true for all P_{sv} of k or fewer edges.

Induction step: Let P_{sv} be a $s \to v$ path of $k+1$ edges.

$P_{sv} = P_{su} + (u,v)$ for some u.

$\text{length}(P_{sv}) = \text{length}(P_{su}) + d(u,v) \geq d_s[u] + d(u,v) \geq d_s[v]$

otherwise could relax.

□

Together, these lemmas imply the Thm.

□

Implementation: $G = (V, E)$

Shortest Path (G, s, t):

Initialize $d[u] = \infty$ $\forall u$.

Shortest Path (G, s, t):

Initialize $d[u] = \infty \quad \forall u$.

$d[s] = 0$.

$Q = [s]$ (a queue)

while queue not empty:

~~while relaxation possible:~~

~~for $u \in V$:~~

$u = Q.\text{dequeue}()$ (remove front item)

for $v \in \text{neighbors}(u)$: (relax edges)

if $d[v] > d[u] + d(u,v)$:

$d[v] = d[u] + d(u,v)$

$\text{parent}[v] = u$

if $v \notin \text{queue}$, $Q.\text{enqueue}(v)$

$Q_0 = [s]$

for $i = 0, 1, 2, \ldots$

while Q_i not empty:

$u = Q_i.\text{dequeue}()$

→ if $v \notin Q_{i+1}$, $Q_{i+1}.\text{enqueue}(v)$

When is relaxation possible? Need $d_s[u] + d(u,v) < d_s[v]$ ← only ever decrease $d_s[u]$

└ this must have decreased

We can keep track of all nodes that decrease and just check them.

Running time:

Let's consider the queue in stages. $Q = [Q_0, Q_1, Q_2, \ldots Q_{|V|-1}]$

Q_i will capture all updates from paths with i edges,

because things are added to Q_i because nodes in Q_{i-1} were updated.

⇒ If we ever reach $Q_{|V|}$, then we have a cycle.

Each subqueue takes $O(|E|)$ time to process because we might have to check the Ford rule on every edge.

Total running time = $O(|V||E|)$. (slower than Dijkstra $O(|E| \log |V|)$)

Alt implementation:

Notice: each edge can only be relaxed $|V|-1$ times without forming a neg cycle.

⇒ can directly count that.

Shortest Path (G, s, t):

Initialize $d[u] = \infty \quad \forall u. \quad d[s] = 0$

```
ShortestPath(G, s, t):
    Initialize  d[u] = ∞  ∀u.   d[s] = 0
    for i = 1, ..., |V|-1:
        for (u,v) ∈ E:
            if d[v] > d[u] + d(u,v):
                d[v] = d[u] + d(u,v)
                parent[v] = u
    for (u,v) in E:
        if d[u] + d(u,v) < d[v]:
            report negative-weight cycle
```

$O(|V||E|)$ time.

(slower than Dijkstra)

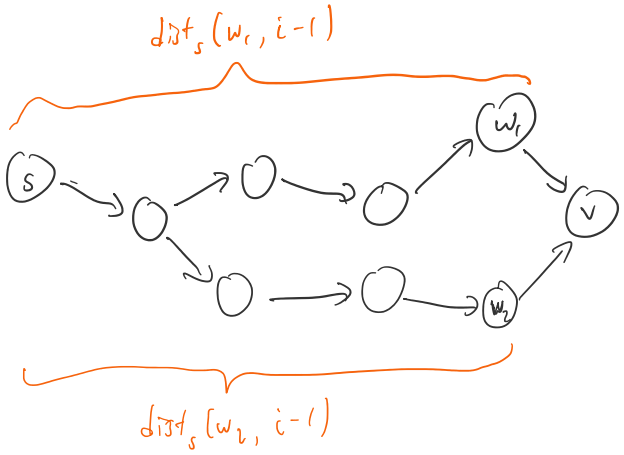
Dynamic Programming view: (pp solves problems by solving subproblems)

Def. Let $dist_s(v, i)$ be min-cost of $s \to v$ path using $\le i$ edges

Can recursively define $dist_s(v, i)$:

1. If best $s \to v$ path uses at most $i-1$ edges, then
   $dist_s(v, i) = dist_s(v, i-1)$

2. If best $s \to v$ path uses i edges, and the last edge is (w, v),
   then $dist_s(v, i) = d(w, v) + dist_s(w, i-1)$. ← similar to Ford update rule



Let $N(w)$ be neighbors of w.

Recurrence:

$$dist_s(v,i) = \min \begin{cases} dist_s(v, i-1) \\ \min_{w \in N(v)} dist_s(w, i-1) + d(w,v) \end{cases}$$

} $dist_s(v, x)$ only depends on $dist_s(w, y)$ for $y < x$.

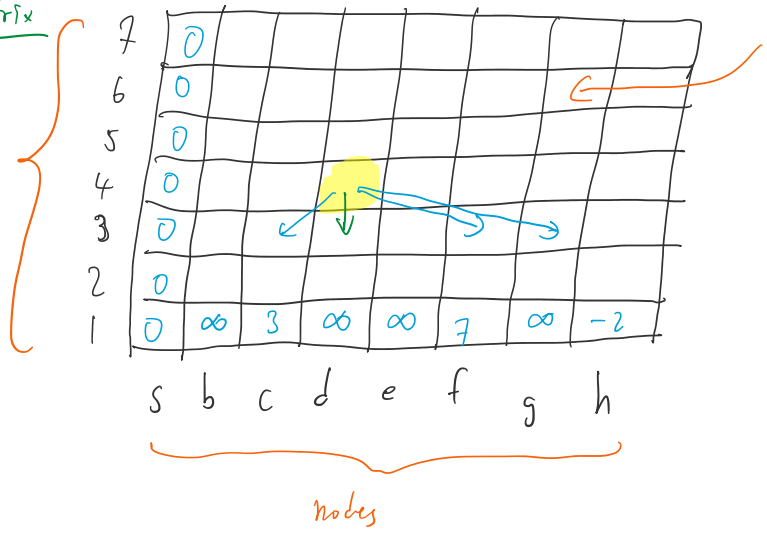
Base case: $dist_s(v, 1) = d(s, v)$ or ∞ if (s, v) does not exist.

Goal: compute $dist_s(t, n-1)$.

$|V|$ nodes, $|V|-1$ allowed hops

Only $|V| \times (|V|-1)$ possible arguments for $dist_s(\cdot, \cdot)$

Encode as matrix



Can fill from bottom up.

Bellman-Ford Pseudocode

```
Bellman-Ford (G, s, t)
    Initialize dist_s[x, 1] to d(s, x) ∀x
    For i = 1, ..., |V|-1:
        For v in V:
            best_w = None
            for w in N(v):
                best_w = min(best_w, dist_s[w, i-1] + d[w, v])
            dist_s[v, i] = min(best_w, dist_s[v, i-1])
    Return dist_s[t, n-1]
```

Running time:

Simple: $O(|V|^2)$ subproblems

$O(|V|)$ time to compute each entry in table (have to search all possible neighbors w)

$\Rightarrow O(|V|^3)$ time

Better analysis:

Let n_v be number of in-edges of v.

Actually only $O(n_v)$ time to compute the entry for v.

Total time $= O(|V| \underbrace{\sum_{v \in V} n_v}) = O(|V||E|)$.

in each row only $O(|E|)$ computations since just need to check in-edges once each.