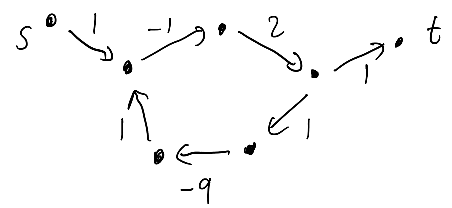
Lec10-Bellman-Ford

KT 6.8

Sunday, September 15, 2024 11:39 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 negative, find the shortest $s \to t$ path.

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 paths, since # edges dominates

$$\text{cost}(P) = M \times \text{hops}(P) + \text{cost}(P)$$

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 neg cycle

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

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

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

Relaxation step (Ford step):

Find 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 for $s \to u$ for all u.

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

proof. **Lemma 1:** After 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 $i = 0$, $d_s[s] = 0 < \infty$, + $\exists\, s \to s$ path of length 0.

Induction hypo: Assume true steps $< i$

Induction step: In step i, we update distance using 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 earlier step.

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

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

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

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

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

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.

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

$\square$

Together, lemmas imply the Thm $\square$

together, lemmas imply the Thm

Implementation

$G = (V, E)$

ShortestPath (G, s, t):

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

$d[s] = 0.$

$Q = [s]$ ← a queue

while $Q \neq \emptyset$:

~~while relaxation possible:~~

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

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

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

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

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

parent $[v] = u$

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

$Q_0 = [s]$

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

while $Q_i \neq \emptyset$:

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

⋮

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

When is relaxation possible? Need $\underbrace{d_s[u]}_{\text{this must have decreased}} + d(u,v) < d_s[v]$ ← only ever decrease $d_s[u]$

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

Running time:

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

Q_i will capture all updates from paths with i edges

because things are added to Q_i when 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 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 finding neg cycle.
$\Rightarrow$ can directly count that.

ShortestPath(G, s, t):

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

```
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 still

Early example of dynamic programming.

DP solves problems by breaking into subproblems.

Bellman-Ford solves first best $s \to u$ path of 0 hops
then best $s \to u$ path of 1 hops
then best $s \to u$ path of 2 hops
⋮
finally best $s \to u$ path of $|V|-1$ hops.

each stage uses answers from previous stage.