Lecture17-ShortestPaths1

Sunday, October 18, 2020   5:51 PM

Applications:

- Map
- Network browsing
- ...

Paths in graph

Paths in graphs

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \to \mathbb{R}$. The *weight* of path $p = v_1 \to v_2 \to \cdots \to v_k$ is defined to be

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}) .$$

Minimize the path length among all possible length

Source	Destination
Single	Single
Single	All
All	Single
All	All

- Complexity wise, there is no difference between the first three
- The output size of the fourth one is n^2

Shortest Path

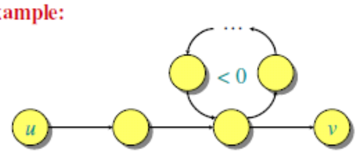
A shortest path from u to v is a path of minimum weight from u to v. The shortest-path weight from u to v is defined as δ(u, v) = min{w(p) : p is a path from u to v}.
Note: δ(u, v) = ∞ if **no path** from u to v exists.

Well-refinedness of shortest paths

Well-definedness of shortest paths

If a graph G contains a negative-weight cycle, then some shortest paths do not exist.

Example:



- Keep taking that negative cycle, then the path get shorter and shorter.

➢ We assume negative weight doesn't exist

Optimal substructure

Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.
Proof: cut and phase

? If the optimal substructure exist in longest single path problem (allow visit each vertex only once)

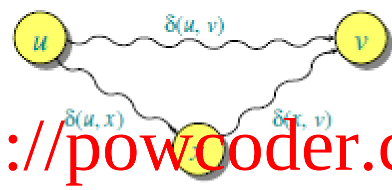
Triangle inequality

Triangle inequality

Shortest path satisfies.

Theorem. For all u, v, x ∈ V, we have δ(u, v) <= δ(u, x) + δ(x, v).

Proof.



Single -source shortest pahts

Single-source shortest paths

(nonnegative edge weights)

Problem. Assume that w(u, v) >= 0 for all (u, v) ∈ E. (Hence, all shortest-path weights must exist.) From a given source vertex s ∈ V, find the shortest-path weights δ(s, v) for all v ∈ V.

Idea: Greedy.

1. Maintain a set S of vertices whose shortestpath distances from s are known.
2. At each step, add to S the vertex v ∈ V – S whose distance estimate from s is minimum.
3. Update the distance estimates of vertices adjacent to v.

Dijkstra

Dijkstra's algorithm

$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
$S \leftarrow \emptyset$
$Q \leftarrow V$ ▷ Q is a priority queue maintaining $V - S$, keyed on $d[v]$
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    $S \leftarrow S \cup \{u\}$
    for each $v \in Adj[u]$
      do if $d[v] > d[u] + w(u, v)$
        then $d[v] \leftarrow d[u] + w(u, v)$

relaxation step

Implicit DECREASE-KEY

- We can also maintain an arrow based on who changed the value.

Lemma 1

Correctness - part I

Lemma. Initializing d[s] ← 0 and d[v] ← ∞ for all v ∈ V – {s} establishes d[v] >= δ(s, v) for all v ∈ V, and this invariant is maintained over any sequence of relaxation steps.

- d[v] >= δ(s, v) the inequality is maintain through the algorithm, meaning the estimate that we are making always an upper bound of the actural shortest path. And they can never less than the sortest path.

Proof. Suppose not. Let v be the first vertex for which $d[v] < \delta(s, v)$, and let u be the vertex that caused $d[v]$ to change: $d[v] = d[u] + w(u, v)$. Then,

$$\begin{aligned} d[v] &< \delta(s, v) && \text{supposition} \\ &\le \delta(s, u) + \delta(u, v) && \text{triangle inequality} \\ &\le \delta(s,u) + w(u, v) && \text{sh. path} \le \text{specific path} \\ &\le d[u] + w(u, v) && v \text{ is first violation} \end{aligned}$$

Contradiction. □

- Focus on the first violation

Lemma 2

Correctness - part II

Lemma. Let u be v's predecessor on a shortest path from s to v. Then, if d[u] = δ(s, u) and edge (u, v) is relaxed, we have d[v] = δ(s, v) after the relaxation.

Proof. Observe that $\delta(s, v) = \delta(s, u) + w(u, v)$. Suppose that $d[v] > \delta(s, v)$ before the relaxation. (Otherwise, we're done.) Then, the test $d[v] > d[u] + w(u, v)$ succeeds, because $d[v] > \delta(s, v) = \delta(s, u) + w(u, v) = d[u] + w(u, v)$, and the algorithm sets $d[v] = d[u] + w(u, v) = \delta(s, v)$. □

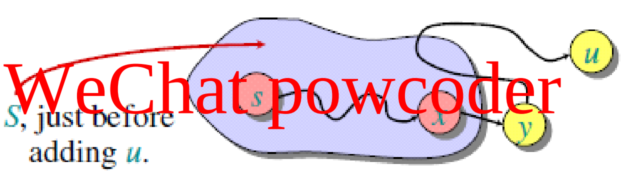
Theorem: correctness

Correctness - part III

Theorem. Dijkstra's algorithm terminates with d[v] = δ(s, v) for all v ∈ V.

Proof. It suffices to show that $d[v] = \delta(s, v)$ for every $v \in V$ when v is added to S. Suppose u is the first vertex added to S for which $d[u] > \delta(s, u)$. Let y be the first vertex in $V - S$ along a shortest path from s to u, and let x be its predecessor:

S, just before adding u.



Since u is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$. When x was added to S, the edge (x, y) was relaxed, which implies that $d[y] = \delta(s, y) \le \delta(s, u) < d[u]$. But, $d[u] \le d[y]$ by our choice of u. Contradiction. □

Running time analysis

Running time analysis:

|V| times: **while** $Q \ne \emptyset$
do $u \leftarrow$ EXTRACT-MIN(Q)
$S \leftarrow S \cup \{u\}$
degree(u) times: **for each** $v \in Adj[u]$
do if $d[v] > d[u] + w(u, v)$
then $d[v] \leftarrow d[u] + w(u, v)$

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.

Time = $\Theta(V \cdot T_{\text{EXTRACT-MIN}} + E \cdot T_{\text{DECREASE-KEY}})$

Note: Same formula as in the analysis of Prim's minimum spanning tree algorithm.

Note: Same formula as in the analysis of Prim's minimum spanning tree algorithm.

- Same formula with Prim

Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(E + V \lg V)$ worst case

What if the edge weight are all 1?
Instead of using priority queue, we can use a queue (simplify the problem, the cost of PQ is complex here.).