Lecture 15: Shortest Paths I: Intro

Lecture Overview

- Weighted Graphs
- General Approach
- Negative Edges
- Optimal Substructure

Readings

CLRS, Sections 24 (Intro)

Motivation:

Shortest way to drive from A to B Google maps “get directions”

Formulation: Problem on a weighted graph $G(V, E)$ $W : E \rightarrow \mathbb{R}$

Two algorithms: Dijkstra $O(V \lg V + E)$ assumes non-negative edge weights
Bellman Ford $O(VE)$ is a general algorithm

Application

- Find shortest path from CalTech to MIT
  - See “CalTech Cannon Hack” photos web.mit.edu
  - See Google Maps from CalTech to MIT
- Model as a weighted graph $G(V, E), W : E \rightarrow \mathbb{R}$
  - $V =$ vertices (street intersections)
  - $E =$ edges (street, roads); directed edges (one way roads)
  - $W(U, V) =$ weight of edge from $u$ to $v$ (distance, toll)

$$\text{path } p = <v_0, v_1, \ldots, v_k>$$
$$(v_i, v_{i+1}) \in E \text{ for } 0 \leq i < k$$

$$w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$$
Weighted Graphs:

Notation:

\[ v_0 \xrightarrow{p} v_k \] means \( p \) is a path from \( v_0 \) to \( v_k \). \((v_0)\) is a path from \( v_0 \) to \( v_0 \) of weight 0.

Definition:

Shortest path weight from \( u \) to \( v \) as

\[
\delta(u, v) = \begin{cases} 
\min \left\{ w(p) : u \xrightarrow{p} v \right\} & \text{if } \exists \text{ any such path} \\
\infty & \text{otherwise (v unreachable from } u) 
\end{cases}
\]

Single Source Shortest Paths:

Given \( G = (V, E) \), \( w \) and a source vertex \( S \), find \( \delta(S, V) \) [and the best path] from \( S \) to each \( v \in V \).

Data structures:

\[
d[v] = \begin{cases} 
0 & \text{if } v = s \\
\infty & \text{otherwise} 
\end{cases} \leftrightarrow \text{initially} \\
= \delta(s, v) \leftrightarrow \text{at end} \\
d[v] \geq \delta(s, v) \text{ at all times}
\]

\( d[v] \) decreases as we find better paths to \( v \), see Figure 1.

\( \Pi[v] = \) predecessor on best path to \( v \), \( \Pi[s] = \text{NIL} \)
Example:

![Shortest Path Example](image.png)

Figure 1: Shortest Path Example: Bold edges give predecessor Π relationships

**Negative-Weight Edges:**

- Natural in some applications (e.g., logarithms used for weights)
- Some algorithms disallow negative weight edges (e.g., Dijkstra)
- If you have negative weight edges, you might also have negative weight cycles
  \[\implies\] may make certain shortest paths undefined!

**Example:**

See Figure 2

\[ B \to D \to C \to B \text{ (origin)} \text{ has weight } -6 + 2 + 3 = -1 < 0! \]

Shortest path \( S \to C \text{ (or } B, D, E) \) is undefined. Can go around \( B \to D \to C \) as
Shortest path \( S \rightarrow A \) is defined and has weight 2

If negative weight edges are present, s.p. algorithm should find negative weight cycles (e.g., Bellman Ford)

**General structure of S.P. Algorithms (no negative cycles)**

\[
\begin{align*}
\text{Initialize:} & \quad \text{for } v \in V: & d[v] & \leftarrow \infty \\
& & \Pi[v] & \leftarrow \text{NIL} \\
& & d[S] & \leftarrow 0
\end{align*}
\]

\[
\begin{align*}
\text{Main:} & \quad \text{repeat} & \\
& & \text{select edge } (u,v) \quad \text{[somehow]} \\
& & \text{“Relax” edge } (u,v) \\
& & \begin{cases} 
& \text{if } d[v] > d[u] + w(u,v) : \\
& \quad d[v] \leftarrow d[u] + w(u,v) \\
& \quad \pi[v] \leftarrow u
\end{cases}
\end{align*}
\]

\[
\text{until all edges have } d[v] \leq d[u] + w(u,v)
\]
Complexity:

Termination?  (needs to be shown even without negative cycles)
Could be exponential time with poor choice of edges.

![Diagram showing the running of the generic algorithm with nodes and edge weights.]

Figure 3: Running Generic Algorithm. The outgoing edges from $v_0$ and $v_1$ have weight 4, the outgoing edges from $v_2$ and $v_3$ have weight 2, the outgoing edges from $v_4$ and $v_5$ have weight 1.

In a generalized example based on Figure 3, we have $n$ nodes, and the weights of edges in the first 3-tuple of nodes are $2^n$. The weights on the second set are $2^n - 1$, and so on. A pathological selection of edges will result in the initial value of $d(v_{n-1})$ to be $2 \times (2^n + 2^{n-1} + \cdots + 4 + 2 + 1)$. In this ordering, we may then relax the edge of weight 1 that connects $v_{n-3}$ to $v_{n-1}$. This will reduce $d(v_{n-1})$ by 1. After we relax the edge between $v_{n-5}$ and $v_{n-3}$ of weight 2, $d(v_{n-2})$ reduces by 2. We then might relax the edges $(v_{n-3}, v_{n-2})$ and $(v_{n-2}, v_{n-1})$ to reduce $d(v_{n-1})$ by 1. Then, we relax the edge from $v_{n-3}$ to $v_{n-1}$ again. In this manner, we might reduce $d(v_{n-1})$ by 1 at each relaxation all the way down to $2^n + 2^{n-1} + \cdots + 4 + 2 + 1$. This will take $O(2^n)$ time.

Optimal Substructure:

**Theorem:** Subpaths of shortest paths are shortest paths

Let $p = <v_0, v_1, \ldots, v_k>$ be a shortest path

Let $p_{ij} = <v_i, v_{i+1}, \ldots, v_j>$  \quad 0 \leq i \leq j \leq k
Then $p_{ij}$ is a shortest path.

Proof: $p = v_0 \rightarrow v_i \rightarrow v_j \rightarrow v_k$

If $p'_{ij}$ is shorter than $p_{ij}$, cut out $p_{ij}$ and replace with $p'_{ij}$; result is shorter than $p$. Contradiction.

Triangle Inequality:

**Theorem:** For all $u, v, x \in X$, we have

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v)$$

**Proof:**

![Figure 4: Triangle inequality](image-url)