Motivation

- PS5: find shortest path from Caltech to MIT.
- Map is a graph of road segments
- Last segment is 0.7 mile on Mass Ave, but some interstate segments are many miles long; we clearly have to represent length of road segment in the graph.

- Arbitrage

I have $1 and can convert it to yen at the local bank; I get ¥2. For each ¥, I can get $0.49 (the bank needs to make money).
If I traverse the cycle, I end back with $\$, but only $0.98: 1 \times 2 \times 0.49 = 0.98$. Not good. But suppose that a bank in Japan has slightly different exchange rates:

\[
\begin{array}{c}
\$ \\
\downarrow 0.49 \\
\uparrow 0.51 \\
\¥
\end{array}
\]

Now there is a path from $\$ to $\$ (a cycle) that makes me $0.02 on the dollar. People exploit that and this is what causes exchange rates to converge. Clearly, weighted graphs are useful...

**Definitions**

A directed graph $G(V,E)$

Edges have weight $w(u,v)$ $(u,v \in V$, $(u,v) \in E)$

Weight is a function $w: E \rightarrow \mathbb{R}$
Path $p = <v_0, v_1, v_2, ..., v_k>$ where
$(v_i, v_{i+1}) \in E$ for $0 \leq i \leq k$; $v_0 \rightarrow v_k$

Weight of a path $w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$

Shortest paths:

$$\delta(u,v) = \begin{cases} 
\min \{w(p): u \xrightarrow{p} v\} & \\
\infty & \text{if } v \text{ is unreachable from } u
\end{cases}$$

Back to Modelling

This clearly works for geographic routes:
we want to minimize the sum of the
segment lengths on the path.
But not for arbitrage; there we multiplied
the rates.

Solution: use $w(u,v) = \log(\text{exchange rate})$

Now an additive cycle with negative
weight means we can make money.
Single-Source Shortest Paths

Input: $G=(V,E)$, $w$, a source vertex $s$
Goal: Find $\delta(s,v)$ for every $v \in V$
and the best path from $s$ to $v$

Data structures:

$d[v] = \text{length of best path to } v$

so far

initialization: $d[v] = \begin{cases} 0 & v = s \\ \infty & \text{otherwise} \end{cases}$

during the algorithm, $d[v] \geq \delta(s,v)$
when the algorithm terminates,
$d[v]$ will be exactly $\delta(s,v)$

$\Pi[v] = \text{predecessor of } v \text{ on best path}$
so far, initialize $\Pi[v] = \text{None}$.

Example

![Graph Diagram]

$d[v]$ at end of alg.

predecessor relationship
General structure of SSSP algorithms

Based on the observation that each $(u,v) \in E$ is a constraint

$$\delta(s,v) \leq \delta(s,u) + w(u,v)$$

We start with some $d[v]$ that satisfies constraints (and $\delta(s,s) = 0$) but is not minimal, and relax the constraints until we are (hopefully) done.

for $v$ in $V$:

$$d[v] = \infty$$

$$\Pi[v] = \text{None}$$

$$d[s] = 0$$

while $d[v] > d[u] + w(u,v)$ for some $u$:

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

$$\Pi[v] = u$$

Does not terminate if $\exists$ negative cycle (need negative cycle detection)

Complexity may be exponential!
Complexity (bad case):

\[ s \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \]

Use \( \rightarrow \) to set \( d[u] \) and recurse from \( u \), then
use \( \rightarrow \rightarrow \) to lower \( d[u] \) and recurse again.

\[ T(n) = 3 + 2T(n-2) = \Theta(2^{n/2}) \]

We'll see two algorithms that do much better:
Bellman-Ford \( O(VE) \), detects negative cycles
Dijkstra \( O(V\log V + E) \) but assumes that
there are no negative cycles.
Two Structural Properties

Theorem: subpaths of shortest paths are also shortest paths.

Proof: Let $p = \langle v_0, v_1, \ldots v_i, v_i, v_{i+1}, \ldots v_j, \ldots v_k \rangle$

$p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$

$p = v_0 \overset{p_{i,i}}{\longrightarrow} v_i \overset{p_{i,i+1}}{\longrightarrow} v_{i+1} \overset{p_{i,j}}{\longrightarrow} v_j \overset{p_{j,k}}{\longrightarrow} v_k$

If $w(p_{ij}) < w(p_{ij})$ then $p$ is not shortest; we can replace $p_{ij}$ with $p'_{ij}$ and get $p: v_0 \overset{p'}{\longrightarrow} v_k$ with $w(p') < w(p)$.

Theorem: triangle inequality, for all $u, v, x \in V$

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

Proof: