Shortest Paths

- Directed Graph $G = (V, E)$
- Length $\text{len}(x, y)$ on $(x, y)$
- $\text{len}(P) = \sum_{(x, y) \in P} \text{len}(x, y)$
- $d(x, y) = \min_{x \to y \text{ path } P} \text{len}(P)$

Today: Single source shortest path from some node $v$ to all other nodes

$d(u) = d(v, u)$

Representation:

![Diagram of a directed graph with shortest paths marked]
Alg 1: Bellman-Ford

Dynamic Programming

Subproblems: $OPT(v)$

$OPT(v, i) =$ shortest path to $v$ with $\leq i$ hops

Then:

$\text{len}(OPT(v, k)) = \begin{cases} 0 & \text{if } v = v_0, k = 0 \\ \infty & \text{if } v = v_k, k = 0 \\ \min \left( \text{len}(OPT(v, k-1)) + \text{len}(v, u) \right) & \text{otherwise} \end{cases}$

$\text{RT}, k = 0$: definition

$k > 0$: let $v$ be argmin \( u \in \text{EE} \) \( \text{len}(OPT(v, k-1)) + \text{len}(v, u) \)

$\Rightarrow$ $OPT(v, k-1)$ path with $k-1$ hops, length $\text{len}(OPT(v, k-1)) + \text{len}(v, u)$

$\Rightarrow$ $\text{len}(OPT(v, k)) \leq \min \left( \text{len}(OPT(v, k-1)) + \text{len}(v, u) \right)$
Now get obvious dynamic program!

\[ MC, 0 \] = \infty \quad \forall \ v \in V, \ n \notin V
\]

\[ MC, 0 \] = 0

\[
\text{for } (k = 1 \text{ to } n-1) \ \\
\quad \text{for } (v \in V) \ \\
\quad \ \\
\quad \text{MC}(v, k) = \min_{w : (v, w) \in E} \left( MC(w, k-1) + \text{len}(w, u) \right)
\]
Then: After BF completes,
\[ MC(u, k) = \text{len}(\text{OPT}(u, k)) \quad \forall k \leq u, \forall u \in V \]

PF: \( k < 0 \): definition

Inductive step:

\[ MC(u, k) = \min_w (MC(u, k-1) + \text{len}(w, u)) \]

\[ \text{def of alg} = \min_w (\text{len}(\text{OPT}(w, k-1)) + \text{len}(w, u)) \]

\[ \text{induction} = \text{len}(\text{OPT}(w, k)) \]

Running time:

\[ \# \text{table entries: } n^2 \]

\[ \text{time/table entry: } \frac{n}{n^3} \]

Then: time \( O(mn) \)
PE: For each k, consider every edge once.

Negative Weights + Cycles:

1) BF good with negative edges
2) BF can be used to detect neg. weight cycles.
   Run one more iteration, check if changed
Relaxation is common primitive in shortest path algorithms.

\( \hat{d}(u) \): upper bound on \( d(u) \)

Init: \( \hat{d}(u) = \infty \) \( \forall u \notin V \)
\( \hat{d}(v) = 0 \)

relax \( (x, y) \): can we improve \( \hat{d}(y) \) by going through \( x \)?

```plaintext
relax \( (x, y) \) {
    if \( (\hat{d}(y) > \hat{d}(x) + \text{len}(x, y)) \) {
        \( \hat{d}(y) = \hat{d}(x) + \text{len}(x, y) \)
        \( y.\text{parent} = x \)
    }
}
```
BF as relaxations:

\[ f_{-v}(i = 1 \text{ to } n) \{ \]
\[ \quad \text{parallel} \]
\[ \quad \text{foreach } (u \in V) \{ \]
\[ \quad \quad \text{foreach edge } (x, u), \relax(x, u) \]
\[ \} \]
\[ \} \]
Dijkstra's Algorithm

"greedy starting at u"

\[ T = \emptyset \]
\[ \hat{d}(v) = 0, \quad \hat{d}(u) = \infty \quad \forall u \in V \]

while (not all nodes in \( T \))
  let \( u \notin T \) be node with minimum \( \hat{d}(u) \)
  add \( u \) to \( T \)
  for each edge \((u, x)\) with \( x \in T\)
    relax \((u, x)\)
**Thm**: Throughout alg:

1) \( T \) is a shortest-path tree from \( u \) to vertex \( v \) in \( T \)
2) \( \hat{d}(u) = d(u) \) \( \forall u \in T \)

**Pf**: Induction on \# iterations of alg.

Base case: \( v \) added at time 1 with \( \hat{d}(v) = \hat{d}(v) = 0 \)

Inductive step: add \( u \) to \( T \), \( u.\text{parent} = u \)

\( \Rightarrow \hat{d}(u) = \hat{d}(v) + \text{len}(w,u) = \hat{d}(v) + \text{len}(w,u) \)

\[ \hat{d}(v) \leq \text{len}(v,y) + \text{len}(y,z) \leq \text{len}(v,w) + \text{len}(w,z) \leq \text{len}(x,v) + \text{len}(v,w) = \hat{d}(v) \]
Running Time:
Alg needs to:
- select node with \( \min \hat{d} \) \( n \) times
- decrease \( \hat{d} \) values \( \leq 1 \) time per relaxation
\( \Rightarrow \leq m \) times total

If store \( \hat{d}(v) \) with \( v \) in a list:
- select takes \( O(n) \) time, decrease \( O(1) \)
\( \Rightarrow \) total \( O(n^2 + mn) = O(n^2) \)

Use heap!
Need:
- Insert \( (n \) times)
- Extract-Min \( (n \) times)
- Decrease-Key \( (n \) times)
Binary Heap:

$O(\log n)$ each op. (amortized)

$\Rightarrow O(m \log n) = O(n \log n)$ total

if $G$ connected

Fib. Heap:

Insert, Decrease-key $O(1)$ amortized

Extract-min $O(\log n)$ amortized

$\Rightarrow O(m + n \log n)$