Today: All-pairs shortest paths
- dynamic programming
- matrix multiplication
- Floyd-Warshall algorithm
- Johnson's algorithm
- difference constraints

Recall: single-source shortest paths [6.006] Chapter 24
- given directed graph $G = (V,E)$, vertex set $V$, & edge weights $w: E \to \mathbb{R}$
- find $s(s,v) =$ shortest-path weight $s \to v \ \forall v \in V$

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted $(w=1)$</td>
<td>BFS</td>
<td>$O(V+E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>$O(E+V \log V)$</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>$O(VE)$</td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td>topological sort</td>
<td>$O(V+E)$</td>
</tr>
<tr>
<td></td>
<td>+1 pass Bellman-Ford</td>
<td></td>
</tr>
</tbody>
</table>

all of these results are the best known
All-pairs shortest paths:
given edge-weighted graph \( G=(V,E,w) \),
find \( S(uv) \) for all \( u,v \in V \)

\[
\begin{array}{cccc}
\text{situation} & \text{algorithm} & \text{time (obvious)} & \text{dense} \\
\text{unweighted} & |V| \times \text{BFS} & O(VE) & \frac{E=O(V^2)}{O(V^3)} \\
\text{nonneg. weights} & |V| \times \text{Dijkstra} & O(VE+V^2 \log V) & O(V^3) \\
\text{general} & |V| \times \text{B-F} & \left\{ \begin{array}{c}
O(V^2 E) \\
O(VE+V^2 \log V)
\end{array} \right. & O(V^3) \\
\text{general} & \text{TODAY} & O(VE+V^2 \log V) & O(V^3)
\end{array}
\]

these results (except third) are also best known — don’t know how to beat \( |V| \times \text{Dijkstra} \)

Assume \( V=\{1,2,\ldots,n^3\} \) so \(|V|=n^3\) today.
**Method I**

**Dynamic program (ℓ1):** \( O(\nu^4) \)

1. subproblem \( d_{uv}^{(m)} = \text{weight of shortest path } u \to v \) using \( \leq m \) edges
2. guessing = what's the last edge \((x,v)\)?
3. \( d_{uv}^{(m)} = \min(d_{ux}^{(m-1)} + w(x,v) \text{ for } x \in V) \)
4. \( d_{uv}^{(0)} = \infty \text{ if } u = v \)
   \( = 0 \text{ else} \)
5. if no neg.-weight cycles then (by B-F analysis) shortest path is simple \( \Rightarrow S(u,v) = d_{uv}^{(n-1)} = d_{uv}^{(n)} = \ldots \) (neg.-weight cycle \( \iff d_{vv}^{(n-1)} < 0 \text{ for some } v \in V \))

**Time:** \( V^3 \) subproblems \( \times \) \( V \) choices \( \times \) \( O(1) \) time/ch.

\( = O(V^4) \) - no better than \( \nu \times \text{Bellman-Ford} \) \( O(\nu^2 E) \)

& worse if \( E \ll \nu^2 \)

**Bottom-up via relaxation steps:** (like Dijkstra & Bellman-Ford)

for \( m \) in range \( (1, n) \):
  for \( u \) in \( V \):
    for \( v \) in \( V \):
      for \( x \) in \( V \):
        if \( d_{uv} > d_{ux} + w_{xv} \):
          \( d_{uv} = d_{ux} + w_{xv} \) \( 3 \) relaxation step

\( \hat{\nu} \) subpath of a shortest path is a shortest path

\( \hat{\nu} \) \( \nu \)

\( \infty \)
Matrix multiplication: (recall)
given $n \times n$ matrices $A$ and $B$, compute $C = A \cdot B$:
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$
- $O(n^3)$ via standard algorithm
- $O(n^{2.807})$ via Strassen's algorithm
- $O(n^{2.376})$ via Coppersmith-Winograd algorithm

Connection to shortest paths:
- define $\oplus = \min$ and $\circ = +$
- then $C = A \oplus B$ is $c_{ij} = \min_k (a_{ik} + b_{kj})$
- define $D^{(m)} = (d_{ij}^{(m)})$, $W = (w(i,j))$, $V = \{1, 2, \ldots, n^3\}$
  \Rightarrow $D^{(m)} = D^{(m-1)} \circ W$ (by $\oplus$ above)
  $$= W^m$$
  where $W^0 = \begin{pmatrix} 0 & 1 & \cdots & 0 \\
1 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 \end{pmatrix}$

$[W^m$ makes sense because $\oplus$ is associative, which follows from $(R, \min, +)$ being closed semiring$]$}

Matrix multiplication algorithm:
- $n-2$ multiplications $\Rightarrow O(n^4)$ time (still no better)
- repeated squaring: $(W^2)^2 \cdots = W^{2^{\log_2 n}} = W^{n-1}
  $$= (S(i,j))$ if no negative-weight cycles
- time: $O(n^3 \log n) = \Theta(n^3 \log \log n)$
- neg.-weight cycles $\Rightarrow$ neg. diagonal entries
- can't use Strassen etc. $\circ$ (no negation)
Special case where we can use Strassen:

**Transitive closure:**
\[
t_{ij} = \begin{cases} 
1 & \text{if there's a path } i \rightarrow j \\
0 & \text{else} 
\end{cases}
\]
\[
= \begin{cases} 
\text{is } S(i,j) < \infty ? & \Rightarrow \text{special case of APSP} \\
\end{cases}
\]
\[
= O(n^{2.376} \log n) \text{ time}
\]

Can also compute mod \( p \)

**Floyd-Warshall algorithm:** faster dynamic program

1. **Subproblem** \( c_{uv}^{(k)} = \text{weight of shortest path } u \rightarrow v \) whose intermediate vertices \( \in \{1, 2, \ldots, k\} \)

\[
\overset{\text{i-->s}}{\overset{\text{sk-->ek}}{\cdots}} \overset{\text{sk-->v}}{\text{sk-->e}} \overset{\text{v-->s}}{\text{v-->e}} \overset{\text{v-->1}}{\text{v-->2}} \overset{\text{v-->n}}{\text{v-->n}}
\]

2. **Guessing** = does shortest path use vertex \( k \)?

3. \( c_{uv}^{(k)} = \min (c_{uv}^{(k-1)}, c_{uk}^{(k-1)} + c_{kv}^{(k-1)}) \)

4. \( c_{uv}^{(0)} = w(u, v) \)

5. \( S(u, v) = c_{uv}^{(n)} \)

**Time:** \( O(V^3) \) subproblems \( \times 2 \) choices \( \times O(1) = O(V^3) \)

Bottom up via relaxation:

\[
C = (w(u, v))
\]

for \( k = 1, 2, \ldots, n \):

for \( u \) in \( V \):

for \( v \) in \( V \):

if \( c_{uv} > c_{uk} + c_{kv} \):

\( c_{uv} = c_{uk} + c_{kv} \) 3 "relaxation" again (with a twist)

\( \Rightarrow \) again OK to omit superscripts

Simple & efficient in practice
Johnsons algorithm:
1. find function \( h: V \rightarrow \mathbb{R} \) such that \( \omega_h(u,v) = \omega(u,v) + h(u) - h(v) \geq 0 \) for all \( u,v \in V \) or determine that a negative-weight cycle exists
2. run Dijkstras algorithm on \( (V,E,\omega_h) \) from every source vertex \( S \in V \)
   \( \Rightarrow \) get \( \delta_h(u,v) \) for all \( u,v \in V \)
3. claim \( \delta(u,v) = \delta_h(u,v) - h(u) + h(v) \)

Proof of claim:
- look at any \( u \to v \) path \( p \) in \( G \)
- say \( p \) is \( V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_k \)

\[ \Rightarrow \omega_h(p) = \sum_{i=1}^{k} \omega_h(V_{i-1} \rightarrow V_i) \]

\[ = \sum_{i=1}^{k} \left[ \omega(V_{i-1} \rightarrow V_i) + h(v_{i-1}) - h(v_i) \right] \]

\[ = \sum_{i=1}^{k} \omega(V_{i-1} \rightarrow V_i) + h(v_0) - h(v_k) \text{ telescoping} \]

\[ = \omega(p) + h(u) - h(v) \]
- so all \( u \to v \) paths change in weight by the same offset \( +h(u) - h(v) \)

\( \Rightarrow \) shortest path is preserved (but offset) \( \square \)
How to find $h$? \( \text{(1)} \)
\[
W_h(u,v) = w(u,v) + h(u) - h(v) \geq 0
\]
\[
\iff h(v) - h(u) \leq w(u,v)
\]
\[
\Rightarrow \text{SYSTEM OF DIFFERENCE CONSTRAINTS} \ h(x) = \text{vars}
\]

**Theorem:** if \( (V,E,w) \) has a negative-weight cycle then no solution to difference constraints

**Proof:** say \( v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k \rightarrow v_0 \) is neg. weight
\[
\begin{align*}
& h(v_1) - h(v_0) \leq w(v_0, v_1) \\
& h(v_2) - h(v_1) \leq w(v_1, v_2) \\
& \quad \vdots \\
& h(v_k) - h(v_{k-1}) \leq w(v_{k-1}, v_k) \\
& h(v_0) - h(v_k) \leq w(v_k, v_0)
\end{align*}
\]
then sum: \( 0 \leq w(\text{cycle}) < 0 \) \( \blacksquare \)

**Theorem:** if \( (V,E,w) \) has no negative-weight cycle then can solve difference constraints

**Proof:** add to \( G \) a new vertex \( s \)
\& add weight-0 edges \((s,v)\) for all \( v \in V \)
- introduce no negative-weight cycles
- \( s \rightarrow v \) path now exists
\( \Rightarrow S(s,v) \) is finite for all \( v \in V \)
- assign \( h(v) = S(s,v) \)
\[
\begin{align*}
h(v) - h(u) & \leq w(u,v) \iff S(s,v) - S(s,u) \leq w(u,v) \\
\Rightarrow S(s,v) & \leq S(s,u) + w(u,v) \text{ TRIANGLE INEQUALITY} \quad \blacksquare
\end{align*}
\]
Alternate reduction: for every \((u,v) \in E\),
add \((u,v)\) with weight \(M' = M \cdot \text{largest } w_l\).

⇒ Strongly connected, still no neg.-weight cycles

Analysis:
1. = Bellman-Ford from \(s\)
   
   \[ O(VE) \]

2. + reweight all edges
   
   \[ O(E) \]

3. = \( |V| \times \text{Dijkstra} \)
   
   \[ O(VE + V^2 \log V) \]

Also: Bellman-Ford can solve any system of difference constraints \(\{x-y \leq c_3\}\)
(or report unsolvable)
in \(O(VE)\) where \(V=\text{variables}, E=\text{constraints}\)

Exercise: Bellman-Ford minimizes \(\max_i x_i - \min_i x_i\)

Applications to real-time programming
multimedia scheduling
temporal reasoning

Bounds on:
- duration
- gap
- synchrony

E.g. \(LB \leq t_{\text{end}} - t_{\text{start}} \leq UB\)
\(0 \leq t_{\text{start2}} - t_{\text{end1}} \leq 3\)
\(|t_{\text{start1}} - t_{\text{start2}}| \leq 3\) or 0