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

Recall: single-source shortest paths [6.006]
- given directed graph $G = (V, E)$, vertex $s \in V$
  & edge weights $w : E \rightarrow \mathbb{R}$
- find $S(s,v) =$ shortest-path weight $s \rightarrow v$, $\forall v \in V$
  (well-defined if no negative weight cycles)

<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 using Fibonacci heaps.
All-pairs shortest paths:
given edge-weighted graph $G = (V, E, w)$,
find $S(u,v)$ for all $u,v \in V$

situation  \hspace{1cm} algorithm  \hspace{1cm} time  \hspace{1cm} \text{E} = O(n^2)
unweighted  \hspace{1cm} |V| \times \text{BFS}  \hspace{1cm} O(VE)  \hspace{1cm} O(n^3)
nonneg. weights  \hspace{1cm} |V| \times \text{Dijkstra}  \hspace{1cm} O(VE + V^2 \log V)  \hspace{1cm} O(n^3)
general  \hspace{1cm} |V| \times \text{B-F}  \hspace{1cm} O(n^3)  \hspace{1cm} O(n^3)
general

TODAY

these results (except third) are also best known — don’t know how to beat $|V| \times \text{Dijkstra}$
{\textbf{Dynamic program (\#1)}:

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. \( d_{uv}^{(m)} = \min (d_{ux}^{(m-1)} + w(x,v) \text{ for } x \text{ in } V) \)

4. \( d_{uv}^{(0)} = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{else} \end{cases} \)

5. if no neg.-weight cycles then (by B-F analysis)
   shortest path is simple \( \Rightarrow S(u,v) = d_{uv}^{(n)} = d_{uv}^{(n+1)} = \ldots \) for \( u,v \in V \)
   (neg.-weight cycle \( \iff d_{uv}^{(n)} < 0 \text{ for some } v \in V \))

\textbf{Time:} \( V^3 \) subproblems \( \cdot O(V) \) guess + combination work/subproblem
\( = O(V^4) \) - no better than \( V \times \text{Bellman-Ford} \)

\textbf{Bottom-up via relaxation steps:} (like Dijkstra \\
& \text{& Bellman-Ford})

for \( m \) in range \((1,n)\):
  for \( u \) in \( V \):
    for \( v \) in \( V \):
      for \( x \) in \( V \):
        if \( d_{uv} > d_{ux} + d_{xv} \):
          \text{relaxation step}
          \( d_{uv} = d_{ux} + d_{xv} \)
          \text{omit superscripts because more relaxation never hurts}
Matrix multiplication: (recall)
given \( n \times n \) matrices \( A \& B \), 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:
- **Gedanken Experiment**: What if \( \oplus = \min \) \& \( \odot = + ? \)
- then \( C = A \oplus B \) is 
  \[
c_{ij} = \min_k (a_{ik} + b_{kj})
\]

- define \( D^{(m)} = (d_{ij}^{(m)}) \), \( W = (w(i,j)) \), \( V = \{1, 2, \ldots, n\} \)
  \[
  \Rightarrow D^{(m)} = D^{(m-1)} \odot W \quad \text{(by \( \odot \) above)}
  = W^{(m)}
\]
  where \( W^{(m)} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \)

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

Matrix multiplication algorithm (#2)
- \( n \) multiplications \( \Rightarrow O(n^4) \) time (still no better)
- repeated squaring: \( (W^2)^2 \cdots = W^{2^{\log_2 n}} = W^{n-1} \)
  \[
  = (S(i,j)) \text{ if no negative-weight cycles}
\]
- time: \( O(n^3 \log n) \)
- neg.-weight cycles \( \Leftrightarrow \) neg. diagonal entries in \( W^m \)
- can't use Strassen etc. \( \Rightarrow \) (no negation)
Transitive closure:  
$t_{ij} = \begin{cases} 
1 & \text{if there's a path } i \rightarrow j \\
0 & \text{else} 
\end{cases}$  
\[ = \begin{cases} 
is S(i, j) < \infty ? \Rightarrow \text{special case of APSP} 
\end{cases} \]
- \((\forall i, \forall j, \text{or-and})\) is a ring \(\Rightarrow\) can use Strassen etc.
\(\Rightarrow O(n^{2.376} \lg n)\) time

Floyd-Warshall algorithm: faster dynamic program

1. subproblem \(c^{(k)}_{uv} = \text{weight of shortest path } u \rightarrow v\) whose intermediate vertices \(\in \{1, 2, \ldots, k\}\) \((V = \{1, 2, \ldots, n\})\)
2. guessing = does shortest path use vertex \(k\)?
3. \(c^{(k)}_{uv} = \min(c^{(k-1)}_{uv}, c^{(k-1)}_{uk} + c^{(k-1)}_{kv})\) \(\Rightarrow\) no neg.-weight cycles
4. \(c^{(0)}_{uv} = w(u, v)\)
5. \(S(u, v) = c^{(n)}_{uv}\)

Time: \(O(n^3)\) subproblems \(\cdot\) 2 choices \(\cdot\) \(O(1)\) \(\Rightarrow\) use vertex \(k\) only once

Bottom up via relaxation:
\(C = (w(u, v))\)
for \(k = 1, 2, \ldots, n:\)
for \(u\) in \(V:\)
for \(v\) in \(V:\)
if \(c_{uv} > c_{uk} + c_{kv}\):  
\(\Rightarrow\) relaxation again
\(c_{uv} = c_{uk} + c_{kv}\)
\(\Rightarrow\) again OK to omit superscripts
Johnson's algorithm (#4)

1. Find a function $h: V \rightarrow \mathbb{R}$ such that $w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$ for all $u, v \in V$ or determine that a negative-weight cycle exists.

2. Run Dijkstra's algorithm on $(V, E, w_h)$ from every source vertex $s \in V$.

3. Claim $S(u, v) = S_h(u, v) - h(u) + h(v)$.

Proof of claim:
- Look at any $u \rightarrow v$ path $p$ in $G$.
- Say $p$ is $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$.

\[ w_h(p) = \sum_{i=1}^{k} w_h(v_{i-1}, v_i) \]

\[ = \sum_{i=1}^{k} \left[ w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i) \right] \]

\[ = \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \text{ (telescoping)} \]

\[ = w(p) + h(u) - h(v) \]

- So all $u \rightarrow v$ paths change in weight by the same offset $h(u) - h(v)$.

\[ \Rightarrow \text{shortest path is preserved (but offset)} \]
How to find $h$? \( (1) \)

$$w_h(u,v) = w(u,v) + h(u) - h(v) \geq 0$$

$\iff$ $h(v) - h(u) \leq w(u,v)$

**SYSTEM OF DIFFERENCE CONSTRAINTS**

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

**Proof:** say $v_0 \to v_1 \to \cdots \to v_k \to v_0$ is neg. weight if $h(v_1) - h(v_0) \leq w(v_0,v_1)$ & $h(v_2) - h(v_1) \leq w(v_1,v_2)$

$\vdots$

& $h(v_k) - h(v_{k-1}) \leq w(v_{k-1},v_k)$

& $h(v_0) - h(v_k) \leq w(v_k,v_0)$

then sum: $0 \leq w(\text{cycle}) < 0 \quad \square$

**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
- $s \to v$ path now exists

$\Rightarrow$ $S(s,v)$ is finite for all $v \in V$

- assign $h(v) = S(s,v)$

$\Rightarrow$ $h(v) - h(u) \leq w(u,v) \iff S(s,v) - S(s,u) \leq w(u,v)$

$\iff S(s,v) \leq S(s,u) + w(u,v)$

**TRIANGLE INEQUALITY** \( \square \)
Analysis:

1. Bellman-Ford from s
   + reweight all edges
   \( O(VE) \) \( \rightarrow O(VE + V^2 \lg V) \)

2. \( |V| \times \text{Dijkstra} \)
   \( O(E) \) \( \rightarrow O(V^2) \)

3. reweight all pairs

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

\[ \begin{align*}
\text{e.g.} & \quad \text{LB} \leq t_{\text{end}} - t_{\text{start}} \leq \text{UB} \\
& \quad 0 \leq t_{\text{start2}} - t_{\text{end1}} \leq 3 \\
& \quad |t_{\text{start1}} - t_{\text{start2}}| \leq 3 \text{ or } 0
\end{align*} \]