InitSSSP(s):
\[\text{dist}(s) \leftarrow 0\]
\[\text{pred}(s) \leftarrow \text{Null}\]
\[
\text{for all vertices } v \neq s
\]
\[\text{dist}(v) \leftarrow \infty\]
\[\text{pred}(v) \leftarrow \text{Null}\]

Edge \( u \rightarrow v \) is tense if
\[\text{dist}(u) + w(u \rightarrow v) = \text{dist}(v)\]

Relax(u→v):
\[\text{dist}(v) \leftarrow \text{dist}(u) + w(u \rightarrow v)\]
\[\text{pred}(v) \leftarrow u\]

FordSSSP(s):
InitSSSP(s)
while there is at least one tense edge
Relax any tense edge
**Bellman-Ford**

**InitSSSP(s)**

repeat $V - 1$ times
  for every edge $u \rightarrow v$
    if $u \rightarrow v$ is tense
      RELAX($u \rightarrow v$)
  for every edge $u \rightarrow v$
    if $u \rightarrow v$ is tense
      return “Negative cycle!”

Finds shortest paths if no cycle has negative weight.

$O(VE)$ time
No Negative Weight Edges: Dijkstra's Observations

1) $u \rightarrow v$ can only become tense if $\text{dist}(u)$ decreases

2) If you relax $u \rightarrow v$, you'll $\text{dist}(v) = \text{dist}(u)$ (if $\omega(u \rightarrow v) = 0$)

Keep a priority queue of fail vertices with key = $\text{dist}(u)$
This is Ford SSSP with Observation 1, so it's correct!
Analysis (assuming no neg. weights)

$\mathbf{u}_i \cdot \mathbf{w} \cdot \mathbf{v}$: vertex returned by $i$th call to $\text{ExtractMin}$

(s0 N, $s$)

$D_{i, i} = \text{dist} (\mathbf{u}_i)$ at the moment we do the $i$th $\text{ExtractMin}$

(s0 $D_{i, i} = 0$)

For all we know so far, $\mathbf{u}_i = \mathbf{u}_j$ for some $i \neq j$. 
Lemma: For all $u \not= j$, we have $d_j = d_i$.

Proof:
Fix $i$. We'll show $d_{i+1} \geq d_i$.

Suppose we relax $u_i \rightarrow u_{i+1}$ during $i$th round.

Immediately after,
$$\text{dist}(u_{i+1}) = \text{dist}(u_i) + w(u_i \rightarrow u_{i+1}) \geq \text{dist}(u_i).$$

Otherwise, $u_{i+1}$ was already in queue. But we didn't Extract
it, so \( \text{dist}(u_i) \leq \text{dist}(u_{\text{init}}) \).

Lemma: Each vertex is extracted at most once.

Proof: Suppose \( v = u_i = u_j \) for some \( j > i \).

We pulled it out, but put it back, so \( d_j < d_i \).

But we just argued that never happens!
Lemma: When Dijkstra ends, for all \( v \), \( \text{dist}(v) \) is the distance to \( v \).

Let \( s = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v = v \) be the shortest path to \( v \).

Let \( L_j \) be the length of \( v_0 \rightarrow \ldots \rightarrow v_j \). We'll prove by induction on \( j \) that \( \text{dist}(v_j) = L_j \).

\[
\text{dist}(v_0) = \text{dist}(s) = 0 = L_0.
\]
Suppose \( j > 0 \). By induction, we can assume we extract \( V_{j-1} \) with \( \text{dist}(V_{j-1}) \leq L_{j-1} \) at that time.

At that moment, either
\[
\text{dist}(V_j) = \text{dist}(V_{j-1}) + w(v_{j-1} \rightarrow v_j)
\]
or we set
\[
\text{dist}(V_j) = \text{dist}(V_{j-1}) + w(v_{j-1} \rightarrow v_j)
\]
when we look at \( v_{j-1} \rightarrow v_j \).

Either way, \( \text{dist}(V_j) \leq \text{dist}(V_{j-1}) + w(v_{j-1} \rightarrow v_j) \leq L_{j-1} + w(v_{j-1} \rightarrow v_j) = L_j \).
In particular, \( \text{dist}(v) = \text{dist}(v_e) = d_e = \text{distance to } v \).

So, we do at most one Insert + Extract Min per vertex.

at most one Decrease Key per edge.

With a binary heap, \( O(\log V) \) per op.
$O(E \log V)$ time total

Still correct with negative weight edges,

Still fast with very few negative weight edges.

But with many negative edges it may take exponential time.
$O(V)$ Insert + Extract Min

$O(E)$ Decrease Keys

**Binary Heap:** $O(\log V)$ time per op

**Fibonacci Heap:**
- $O(1)$ time (on average)
- Insert + Decrease Key: $O(\log V)$ time (on average)
- Extract Min: $O(E + V \log V)$

Dijkstra with Fibonacci Heaps; total
$O(E \log V)$ with binary heaps

$O(E + V \log V)$ with Fibonacci heaps

(Don't do this in practice. Stick to binary heaps.)
Edge Weights = 1 (want to minimize # edges on a path)

Use breadth-first search.

\[ O(V + E) \text{ time} \]

BFS(s):
  INITSSSP(s)
  PUSH(s)
  while the queue is not empty
    \( u \leftarrow \text{PULL()} \)
    for all edges \( u \to v \)
      if \( \text{dist}(v) > \text{dist}(u) + 1 \) \(\langle \text{if } u \to v \text{ is tense} \rangle\)
        \( \text{dist}(v) \leftarrow \text{dist}(u) + 1 \) \(\langle \text{relax } u \to v \rangle\)
        \( \text{pred}(v) \leftarrow u \)
        PUSH(v)
Directed Acyclic Graphs

\[ \text{dist}(v) = \begin{cases} 
0 & \text{if } v = s \\
\min_{u \to v} (\text{dist}(u) + w(u \to v)) & \text{otherwise}
\end{cases} \]

No (negative) cycles.

**DAGSSSP(s):**
for all vertices \( v \) in topological order
  - if \( v = s \)
    \[ \text{dist}(v) \leftarrow 0 \]
  - else
    \[ \text{dist}(v) \leftarrow \infty \]
  - for all edges \( u \to v \)
    - if \( \text{dist}(v) > \text{dist}(u) + w(u \to v) \) \( \langle \text{if } u \to v \text{ is tense} \rangle \)
      \[ \text{dist}(v) \leftarrow \text{dist}(u) + w(u \to v) \] \( \langle \text{relax } u \to v \rangle \)

**PushDAGSSSP(s):**
**InitSSSP(s)**
for all vertices \( u \) in topological order
  - for all outgoing edges \( u \to v \)
    - if \( u \to v \) is tense
      \[ \text{RELAX}(u \to v) \]

\( O(V + E) \) time
All-Pairs Shortest Paths

Want to compute \( \text{dist}(u,v) \)
for all \( u, v \in V \); distance from \( u \) to \( v \).

\( \Theta(V^2) \) values to compute

**ObviousAPSP**\((V, E, w)\): 
for every vertex \( s \)
\[ \text{dist}[s, \cdot] \leftarrow \text{SSSP}(V, E, w, s) \]

*Unweighted or DAG:*
\[ V \cdot O(E) = O(VE) = O(V^3) \] time

*Non-Negative Weights:*
\[ V \cdot O(E + V \log V) = O(VE + V^2 \log V) \]
Otherwise:

\[ V \cdot O(VE) = O(V^2 E) = O(V^4) \]

Can we get \( O(V^3) \) even with negative length edges?