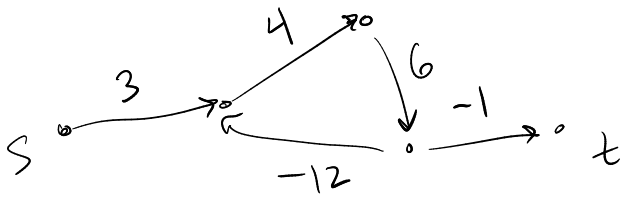
Floyd-Warshall algorithm

Recall: Dijkstra finds shortest paths in weighted graph from a single source w/ no negative edges.

Today:
- Negative edges ✓
- All-pairs shortest paths ✓

Potential problem w/ negative weights:



Negative cycles! Can go around as many times as you want!

Input: weighted, directed graph G

Output: want 2D array S where $S[u,v]$ = length of shortest path $u \to v$. (ideally + way to construct actual shortest path)

Idea: recurrence for S by adding a parameter.

assume vertices numbered $1 \ldots n$.

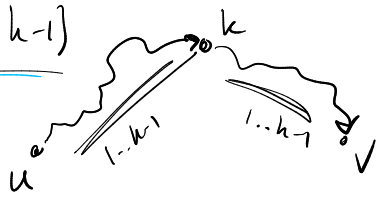
$S[u,v,k]$ = length of shortest path $u \to v$ that only uses vertices $\{1,\ldots,k\}$ as intermediate nodes on path.

Base case(s):

$$S[u,v,0] = \begin{cases} w_{uv} & \text{if there is an edge } u \to v \\ \infty & \text{if no edge } u \to v \\ 0 & \text{if } u = v \end{cases}$$

Recursive case:

$$S[u,v,k] = \min \begin{cases} S[u,v,k-1] & \leftarrow \text{don't use vertex } k \\ S[u,k,k-1] + S[k,v,k-1] \end{cases}$$



We need to memoize! We can fill in an $n \times n \times n$ array. w/ values of s.

As an optimization, we can just use an $n \times n$ array $s[u,v]$ and iteratively overwrite it for each value of k.

Pseudocode:

$$s[u,v] \leftarrow \begin{cases} 0 & \text{if } u = v \\ w_{uv} & \text{if } u \to v \\ \infty & \text{otherwise} \end{cases}$$

$next[u,u] \leftarrow u$ for $u \in V$. (?)

```
for k ← 1...n:
    for u ← 1...n:
        for v ← 1...n:
            if s[u,k] < ∞ and s[k,v] < ∞ and
                    s[u,k] ⊕ s[k,v] < s[u,v]:
                s[u,v] ← s[u,k] + s[k,v]
                next[u,v] ← next[u,k]
```

Keep 2D array $next[u,v]$ = next vertex along shortest path from u to v.

$\Theta(n^3)$.

ie $\Theta(V^3)$.

Check: $s[u,u] < 0$ iff u is on a negative cycle.