Lecture

Bellman Ford Algorithm

Wilson Rivera

Bellman-Ford Algorithm

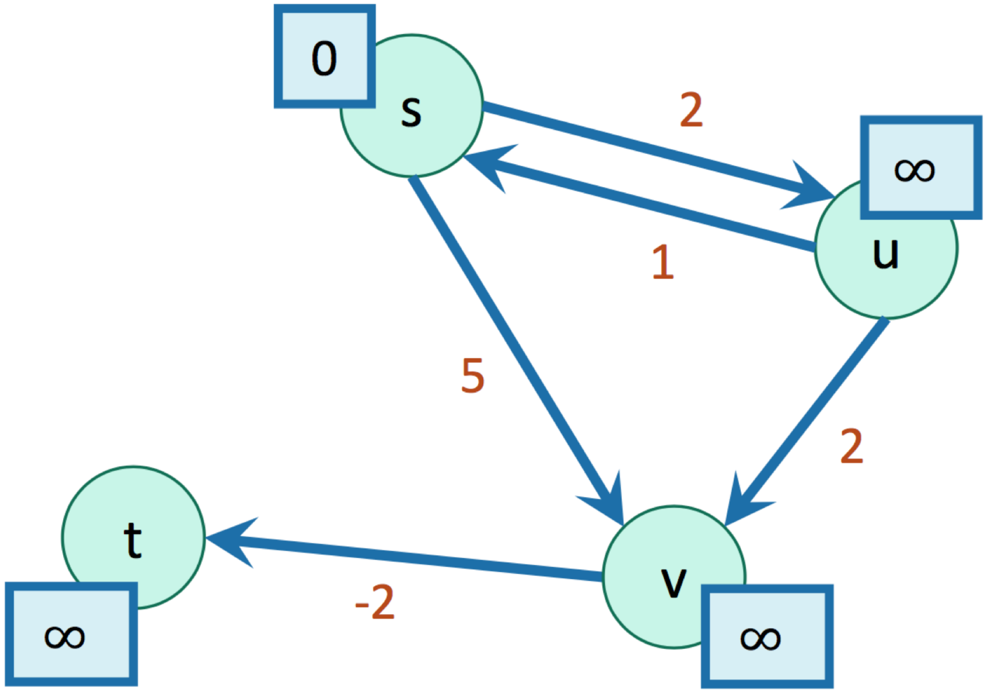
Algorithm 1: Bellman-Ford Algorithm (G, s)

$d^{(0)}[v] = \infty \forall v \in V$
$d^{(0)}[s] = 0$
$d^{(k)}[v] = \text{None} \forall v \in V \forall k > 0$
for $k = 1, \ldots, n-1$ **do**
    $d^{(k)}[v] \leftarrow d^{(k-1)}[v]$
    for $(u, v) \in E$ **do**
        $d^{(k)}[v] \leftarrow \min\{d^{(k)}[v], d^{(k-1)}[u] + w(u, v)\}$
    // here we can release the memory for $d^{(k-1)}$, we'll never need it again.
return $d^{(n)}[v], \forall\ v \in V$

Bellman-Ford Algorithm

$$d^{(k)}[b] \leftarrow \min\{\, d^{(k-1)}[b],\ \min_a \{d^{(k-1)}[a] + \mathrm{weight}(a,b)\} \,\}$$

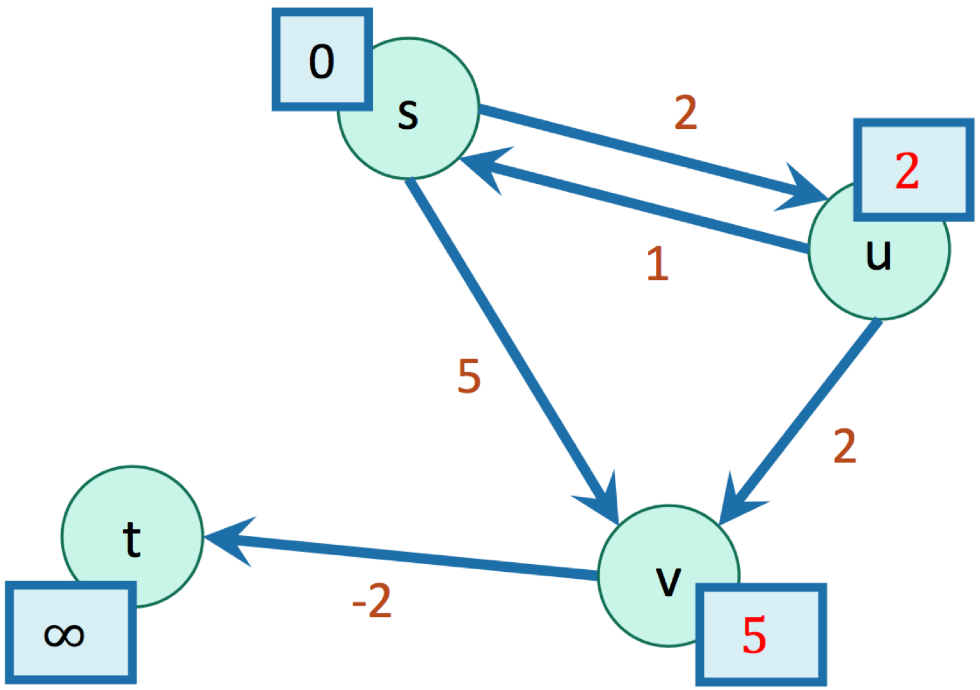
	s	u	v	t
$d^{(0)}$	0	∞	∞	∞
$d^{(1)}$				
$d^{(2)}$				
$d^{(3)}$				



Bellman-Ford Algorithm

$$d^{(k)}[b] \leftarrow \min\{\, d^{(k-1)}[b],\ \min_a \{d^{(k-1)}[a] + \mathrm{weight}(a,b)\} \,\}$$

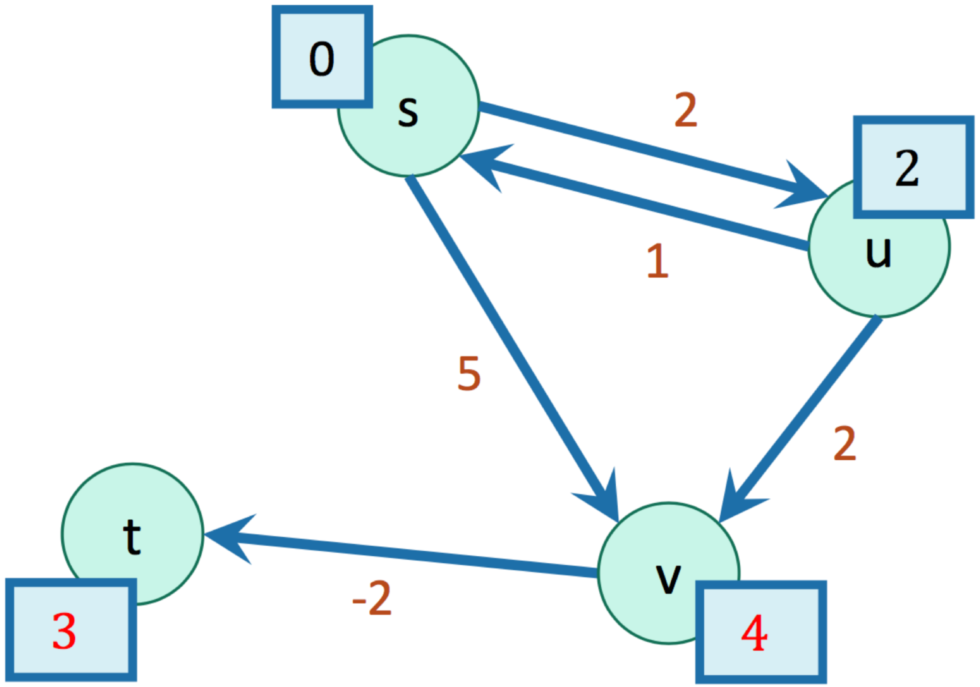
	s	u	v	t
$d^{(0)}$	0	∞	∞	∞
$d^{(1)}$	0	2	5	∞
$d^{(2)}$				
$d^{(3)}$				



Bellman-Ford Algorithm

$$d^{(k)}[b] \leftarrow \min\{\, d^{(k-1)}[b],\ \min_a \{d^{(k-1)}[a] + weight(a,b)\}\,\}$$

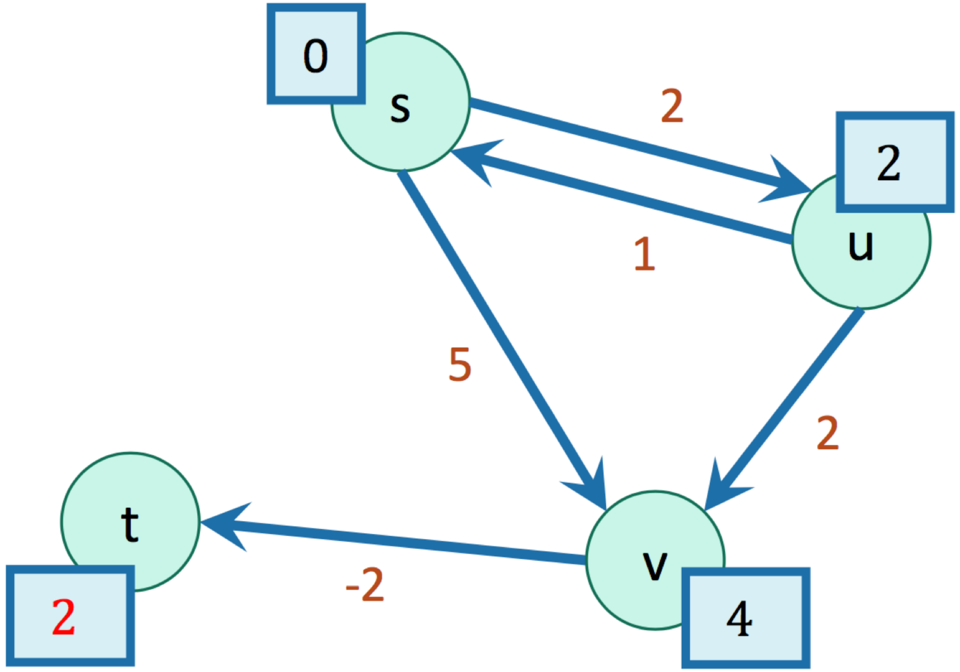
	s	u	v	t
$d^{(0)}$	0	∞	∞	∞
$d^{(1)}$	0	2	5	∞
$d^{(2)}$	0	2	4	3
$d^{(3)}$				



Bellman-Ford Algorithm

$$d^{(k)}[b] \leftarrow \min\{\, d^{(k-1)}[b],\ \min_a \{d^{(k-1)}[a] + weight(a,b)\}\,\}$$

	s	u	v	t
$d^{(0)}$	0	∞	∞	∞
$d^{(1)}$	0	2	5	∞
$d^{(2)}$	0	2	4	3
$d^{(3)}$	0	2	4	2



Bellman-Ford Algorithm

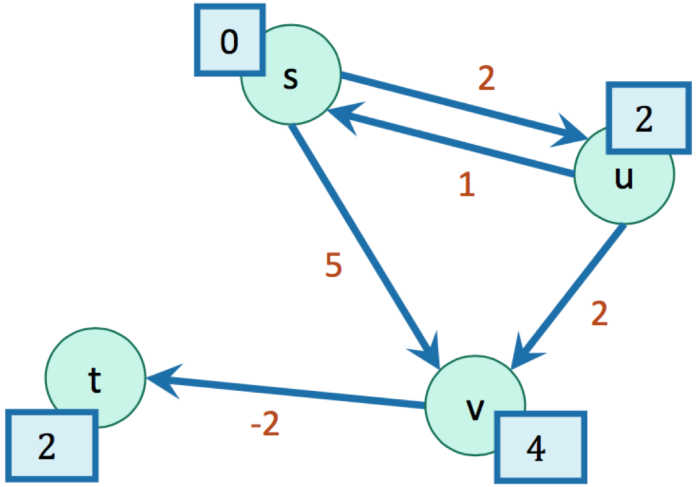
$d^{(k)}[b]$ is the cost of the shortest path from s to b with at most k edges in it.

SANITY CHECK:

- The shortest path with 1 edge from s to t has cost ∞. (there is no such path).
- The shortest path with 2 edges from s to t has cost 3. (s-v-t)
- The shortest path with 3 edges from s to t has cost 2. (s-u-v-t)

And this one is the shortest path!!!

	s	u	v	t
$d^{(0)}$	0	∞	∞	∞
$d^{(1)}$	0	2	5	∞
$d^{(2)}$	0	2	4	3
$d^{(3)}$	0	2	4	2



Bellman-Ford Algorithm

- Slower (but arguably simpler) than Dijkstra's algorithm
  - O(mn)
- Works with negative edge weights.
- Space complexity
- We need space to store the graph and two arrays of size n.
- The Bellman-Ford Algorithm is correct as long as G has no negative cycles.

Dynamic Programming

- **Dynamic Programming** is a basic paradigm in algorithm design used to solve problems by relying on intermediate solutions to smaller sub-problems.
  - **Optimal substructure**.
    - Optimal solutions to sub-problems are sub-solutions to the optimal solution of the original problem.
  - **Overlapping sub-problems**.
    - The sub-problems show up again and again

$$d^{(k)}[b] \leftarrow \min\{\, d^{(k-1)}[b],\ \min_a \{d^{(k-1)}[a] + \mathrm{weight}(a,b)\}\,\}$$

- Bellman-Ford algorithm
  - Another single-source shortest path algorithm
  - This is an example of **dynamic programming**