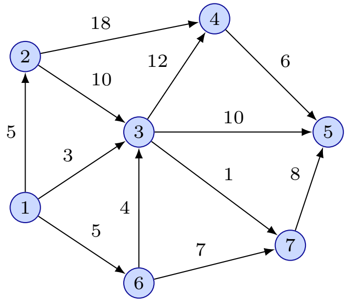
EE365: The Bellman-Ford Algorithm

Shortest path problems

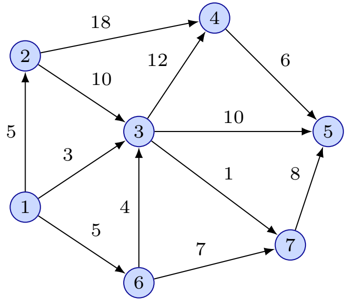
- given weighted graph and a destination vertex
- find lowest cost path from *every vertex* to destination



Dynamic programming principle

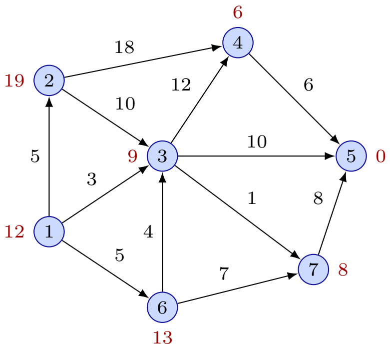
- let g_{ij} = cost of edge $i \to j$ $\qquad$ (∞ if no edge)
- let v_i = cost of shortest path from i to destination; it must satisfy

$$v_i = \min_j (g_{ij} + v_j)$$



Dynamic programming principle

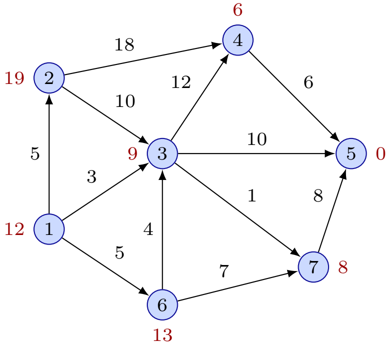
$$v_i = \min_j (g_{ij} + v_j)$$



- starting at vertex i
- g_{ij} is cost of next step
- shortest path minimizes sum of
  - cost for next step
  - shortest path from where you land

Dynamic programming principle

$$v_i = \min_j (g_{ij} + v_j)$$



- once we know v, we also know the optimal path from all initial vertices
- from vertex i, move to the minimizer j

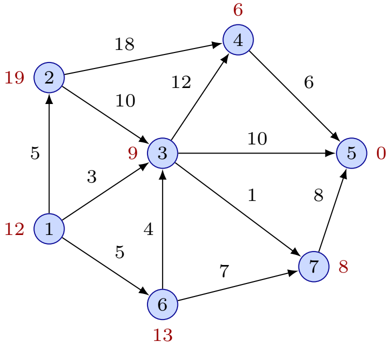
Bellman-Ford algorithm

- let $v_i^0 = \begin{cases} 0 & \text{if } i = \text{destination} \\ \infty & \text{otherwise} \end{cases}$
- for $k = 0, \ldots, n-1$
  - $v_i^{k+1} = \min\{v_i^k, \min_j (g_{ij} + v_j^k)\}$

- v_i^k is lowest cost path from i to destination in k steps or fewer
- if $v^n \neq v^{n-1}$ then graph has negative cycle, and cost may be made $-\infty$
- stop early if $v^{k+1} = v^k$
- n vertices, m edges, runs in $O(mn)$ time

Bellman-Ford algorithm

$$v_i^{k+1} = \min\{v_i^k, \min_j(g_{ij} + v_j^k)\}$$



$$v^0 = \begin{bmatrix} \infty \\ \infty \\ \infty \\ \infty \\ 0 \\ \infty \\ \infty \end{bmatrix} \quad v^1 = \begin{bmatrix} \infty \\ \infty \\ 10 \\ 6 \\ 0 \\ \infty \\ 8 \end{bmatrix} \quad v^2 = \begin{bmatrix} 13 \\ 20 \\ 9 \\ 6 \\ 0 \\ 14 \\ 8 \end{bmatrix} \quad v^3 = \begin{bmatrix} 12 \\ 19 \\ 9 \\ 6 \\ 0 \\ 13 \\ 8 \end{bmatrix}$$

Dynamic programming

- breaks up large problem into nested subproblems
- works backward in time (for deterministic problems, can also work forwards)
- stores the solution of subproblems in the value function, to allow reuse at many states

Shortest path problems

- Dijkstra's algorithm is similar but faster ($O(m + n \log n)$), and requires non-negative weights
- both BF and Dijkstra give shortest path from *every* vertex to destination
- other algorithms, such as $A^\star$, find shortest path between two vertices