12 Feb 2024

The Bellman-Ford Algorithm (§6.8)

Input: A graph $G = (V, E)$ [directed]

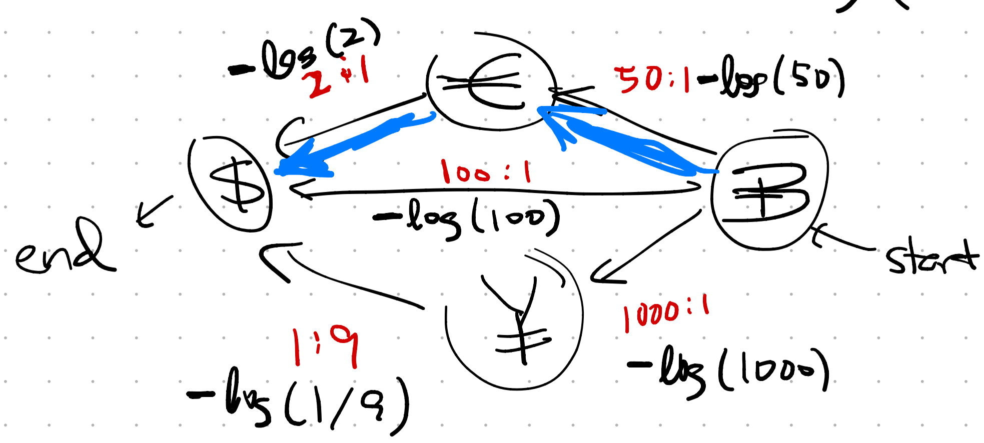
Edge costs c_{uv} for each $(u,v) \in E$.

Requirement: No negative cost cycles.

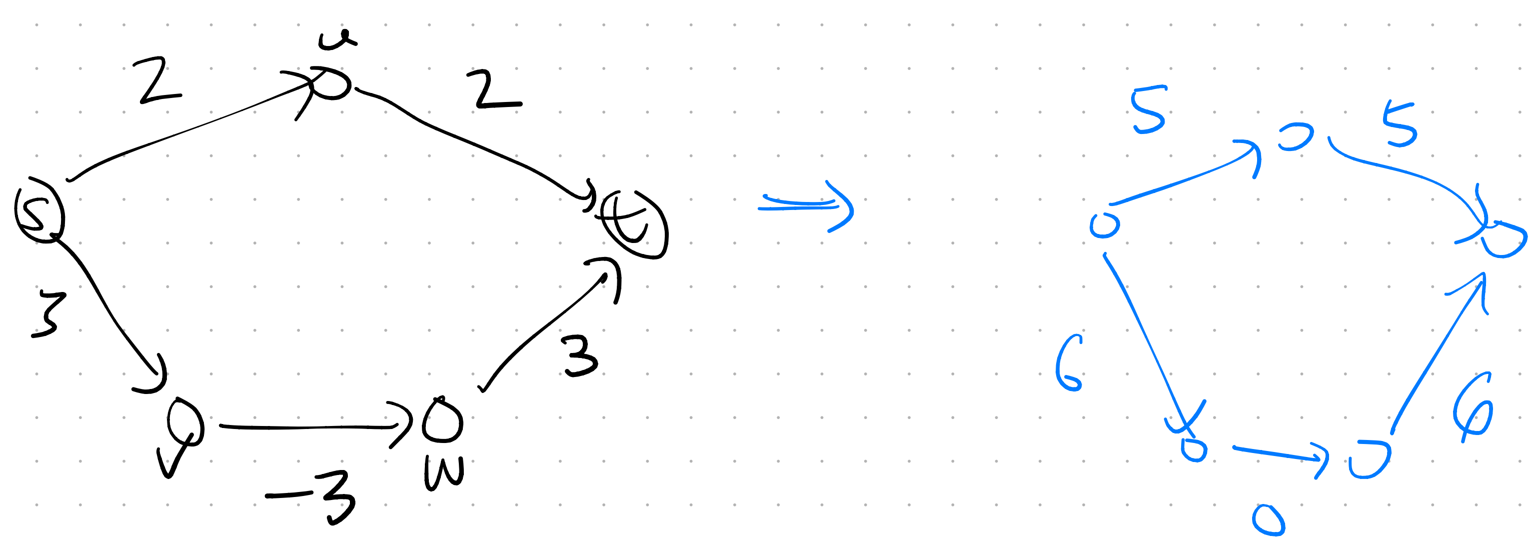
First part of lecture: No cycles at all.

E.g. vertices are currencies ($, €, ¥, ฿)

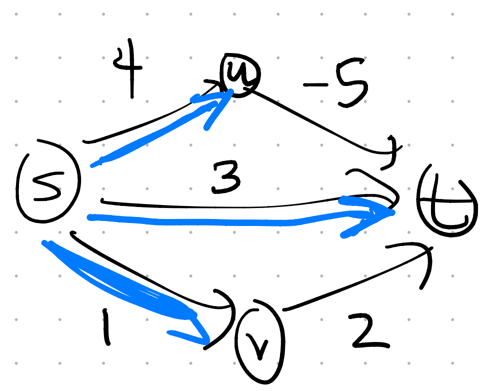
edge costs are log(exchange rate).



Just add a const. to each edge cost to make them ≥ 0?



Run Dijkstra?

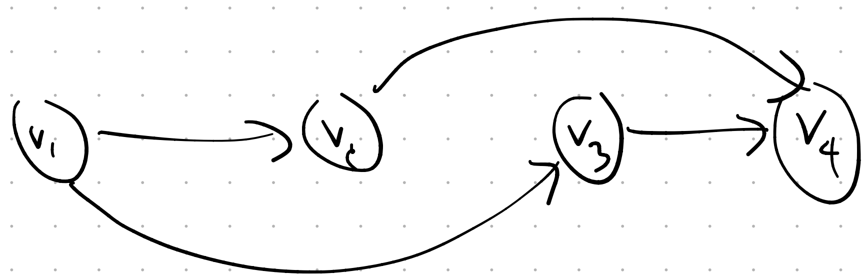


Bellman–Ford in DAGs (DAG-BF)

Assume vertex set V is in topological sort order.

$V = \{v_1, v_2, \ldots, v_n\}$

Every $(v_i, v_j) \in E$ satisfies $i < j$.



Takes $O(\underset{|E|}{m} + \underset{|V|}{n})$ time to find this ordering.

Assume $s = v_1$, $t = v_n$.

$$\text{PATHS}(s, v_j) = \begin{cases} \{\langle s \rangle\} & \text{if } j = 1 \\ \displaystyle\bigcup_{\text{edges } e = (v_i, v_j)} \text{APPEND}_e(\text{PATHS}(s, v_i)) & \text{if } j > 1 \end{cases}$$

Every path from s to v_j is formed by appending some edge $e = (v_i, v_j)$ to a path from s to some earlier vertex v_i.

(*) $$\text{MINCOST}(s, v_j) = \begin{cases} 0 & \text{if } j = 1 \\ \min\{c_{ij} + \text{MINCOST}(s, v_i) : (v_i, v_j) \in E\} & \text{if } j > 1. \end{cases}$$

DAG-BF(G, s, t):

Topologically sort G. Assume $V = \{v_1, \ldots, v_n\}$, $i < j$ $\forall (v_i, v_j) \in E$. $s = v_1$, $t = v_n$.

$M[1] = 0$
for $j = 2, \ldots, n$:

$$M[j] = \min\left\{ c_{ij} + M[i] \;\middle|\; (v_i, v_j) \in E \right\}$$

treating $\min(\emptyset)$ as ∞

endfor
output $M[n]$.

Time complexity of loop iteration j

$$= O(\#\text{ edges into } j) + O(1)$$

Total time complexity

$$\sum_j O(\#\text{ edges into } j) + \sum_j O(1)$$

$$O(m) + O(n)$$

$$O(m+n)$$

DAG-BF is $O(m+n)$.

If G contains cycles:

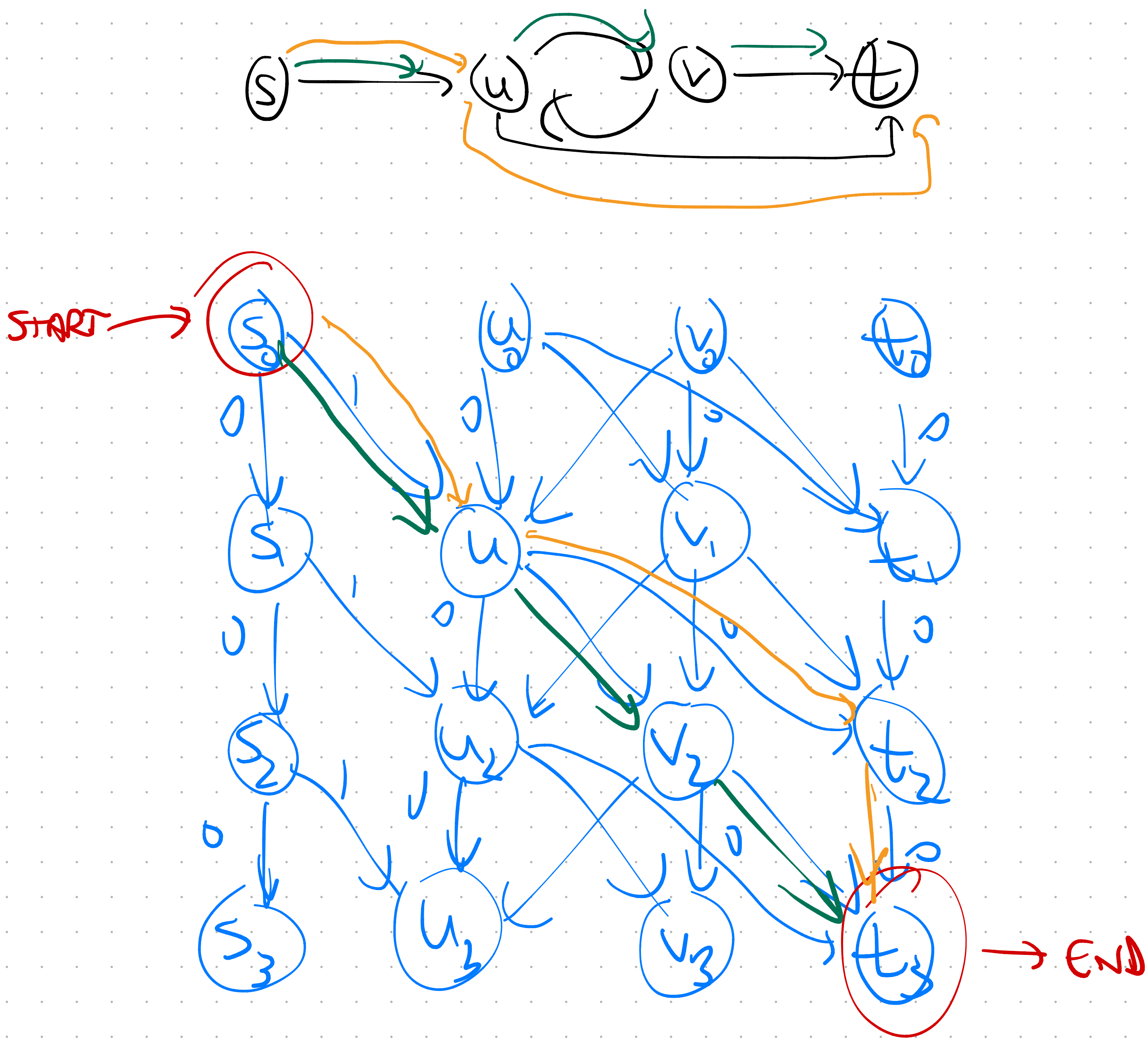
- if $\exists$ cycle of negative total cost reachable from s, and can reach t, then $\nexists$ min-cost path.

$$\begin{tikzcd} s \arrow[r, "1"] & u \arrow[r, bend left, "-1"] & v \arrow[l, bend left, "-1"] \arrow[r, "1"] & t \end{tikzcd}$$

- Assume no negative cost cycles.

THEN: The min-cost s-t path has $\leq n$ vertices in it.

Convert G with cycles into $G \times [n]$ acyclic.



This construction reduces general case to DAG case.

BF runs in $O(mn + n^2)$