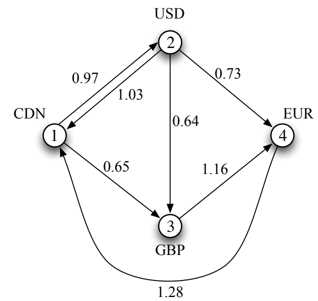
# COMPSCI 311: Introduction to Algorithms

## Lecture 17: Dynamic Programming – Shortest Paths

Dan Sheldon

University of Massachusetts Amherst

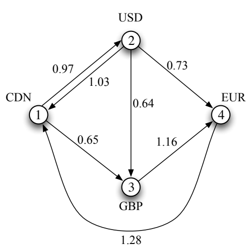
## Currency Trading



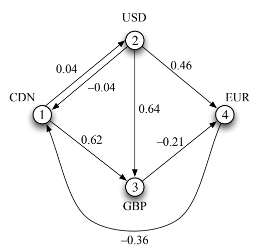

- **Problem**: given directed graph with exchange rate $r_e$ on edge $e$, find $s \to t$ path $P$ to maximize overall exchange rate $\prod_{e \in P} r_e$
- **Assumption** (no arbitrage): no cycles $C$ such that $\prod_{e \in C} r_e > 1$.

## From Rates to Costs

- Similar, but not the same as finding a shortest path.
- Let's change from rates to costs by transforming the problem.
- Let $c_e = -\log r_e$ be the *cost* of edge $e$



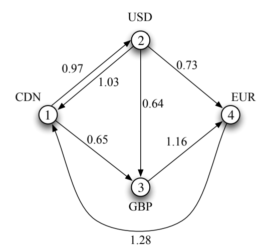

**Rates**

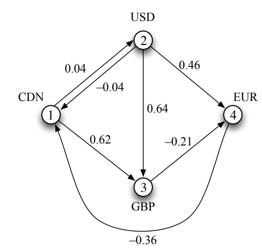



**Costs**

## From Rates to Costs

- The cost (length) of a path becomes the negative log of its rate

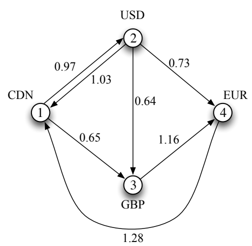


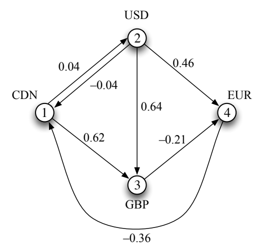




$$c_{12} + c_{23} + c_{34} = -\log r_{12} + -\log r_{23} + -\log r_{34} = -\log(r_{12} r_{23} r_{34})$$

## From Rates to Costs

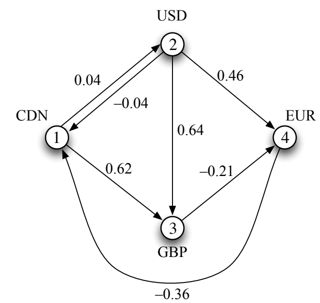
- Because $\log$ is monotone we have: lower cost $\iff$ higher rate




- **New problem**: find the $s \to t$ path of minimum cost

## Currency Trading as Shortest Path Problem




- Negative edge weights!
- **Problem**: given a graph with edge weights that may be negative, find shortest $s \to t$ path
- **Assumption**: no cycle $C$ such that $\sum_{e \in C} c_e < 0$. Why?

## Dynamic Programming Approach (False Start)

- Let $\mathrm{OPT}(v)$ be the cost of the shortest $v \to t$ path
- What goes wrong with this?
- The recurrence is not well-defined, e.g., there are nodes $i$ and $j$ where $\mathrm{OPT}(i)$ depends on $\mathrm{OPT}(j)$ and vice versa.
- **Idea**: We can fix this by "adding a variable" to the recurrence that is always decreasing.

## Bellman-Ford Algorithm

Let $\mathrm{OPT}(i, v)$ be cost of shortest $v \rightsquigarrow t$ path $P$ **with at most** $i$ **edges**

- If $P$ uses at most $i-1$ edges then $\mathrm{OPT}(i, v) = \mathrm{OPT}(i-1, v)$
- Else $P = v \to w \rightsquigarrow t$ where $w \rightsquigarrow t$ path uses $i-1$ edges, so

$$\mathrm{OPT}(i, v) = c_{v,w} + \mathrm{OPT}(i-1, w)$$

This gives the recurrence

$$\mathrm{OPT}(i, v) = \min\left\{\mathrm{OPT}(i-1, v), \ \min_{w \in V}\{c_{v,w} + \mathrm{OPT}(i-1, w)\}\right\}$$
$$\mathrm{OPT}(0, t) = 0$$
$$\mathrm{OPT}(0, v) = \infty \text{ if } v \neq t$$

## Clicker

With negative edge lengths, paths can get *shorter* as we include more edges.

Assuming all cycles have positive cost and $m > n$, what is the largest possible number of edges in a shortest-length path from $v$ to $t$?

A. $n$

B. $m$

C. $n-1$

D. $m-1$

## Bellman-Ford

$$\mathrm{OPT}(i,v) = \min\left\{\mathrm{OPT}(i-1,v),\ \min_{w\in V}\{c_{v,w} + \mathrm{OPT}(i-1,w)\}\right\}$$

Subproblems? $\mathrm{OPT}(i,v)$ for $i = 1$ to $n-1$, $v \in V$
(**Fact**: shortest path has at most $n-1$ edges)

Shortest-Path($G$, $s$, $t$)

- $n$ = number of nodes in $G$
- Create array $M$ of size $n \times n$
- Set $M[0,t] = 0$ and $M[0,v] = \infty$ for all other $v$
- **for** $i = 1$ to $n-1$ **do**
  - **for** all nodes $v$ in any order **do**
    - Compute $M[i,v]$ using the recurrence above

Running time? $O(n^3)$. Better analysis $O(mn)$. Example

## Clicker

Suppose there is some iteration $i$ for which $M[i,v] = M[i-1,v]$ for all $v$. Then

A. There is a negative cycle in the graph.

B. We can terminate the algorithm after the $i$th iteration, because no future values will change.

C. There are no negative edge costs in the graph.

D. The graph is undirected.

## Bellman-Ford-Moore: Efficient Implementation

- Store only one column: $M$ array $\rightarrow$ $d$ vector
- Only consider neighbors $w$ whose value changed
- Keep track of shortest path using successor array

Shortest-Path($G$, $t$)

- set $d[t] = 0$ and $d[v] = \infty$ for all $v \neq t$
- set succ$[v] =$ `null` for all $v$
- **for** $i = 1$ to $n-1$ **do**
  - **for all** nodes $w \neq t$ **do**
    - **if** $w$ updated in last iteration **then**
      - **for all** $(v,w) \in E$ **do**
        - **if** $c_{v,w} + d[w] < d[v]$ **then**
          - $d[v] = c_{v,w} + d[w]$
          - succ$[v] = w$

- Space? $O(m+n)$, time $O(mn)$

## Clicker

Suppose we remove the assumption that there are no negative cycles, and find that $\text{OPT}(n, v) < \text{OPT}(n-1, v)$ for some node $v$. Then

A. There is a negative cycle on some $v \rightsquigarrow t$ path in the graph.

B. There are no negative edge costs in the graph.

C. There is a negative cycle on some $t \rightsquigarrow v$ path in the graph.

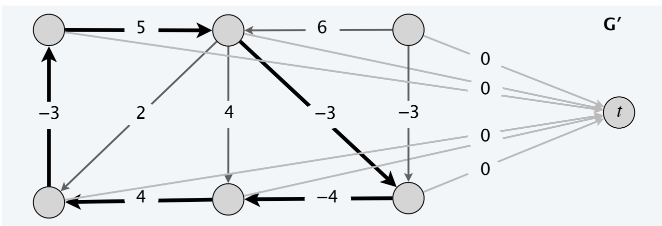
D. There are no negative cycles in the graph.

## Negative Cycles

- How to detect negative-weight cycles?
  - Suppose $\text{OPT}(n, v) < \text{OPT}(n-1, v)$. Then there is a negative cycle on some $v \rightsquigarrow t$ path, since shortest paths have at most $n-1$ edges in the absence of negative cycles.
  - Suppose $\text{OPT}(n, v) = \text{OPT}(n-1, v)$ for all $v$. Then the algorithm will not update after the $n$th iteration $\implies$ no negative cycles on any $v \rightsquigarrow t$ path.
- **Fact**: there is a negative cycle on some $v \rightsquigarrow t$ path iff $\text{OPT}(n, v) < \text{OPT}(n-1, v)$ for some $v$.
- Detect negative cycles by running for one more iteration to see if some value decreases!

## Detecting Negative-Weight Cycles

But this only detects cycles on paths to a fixed target node $t$. How to find a negative-weight cycle anywhere in the graph?




Add a dummy target node.