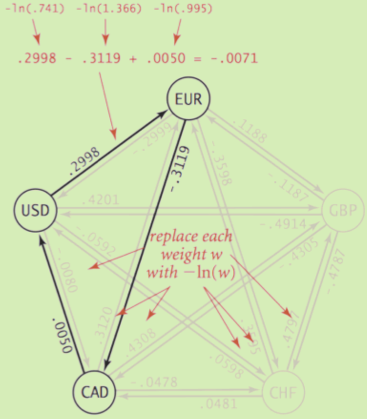
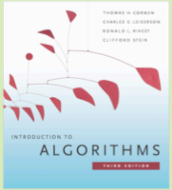


A negative cycle that represents an arbitrage opportunity

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

```
BELLMAN-FORD-FIND-NEGATIVE-CYCLE(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i ← 1 to |V[G]| − 1
    do for each edge (u, v) ∈ E[G]
            do RELAX(u, v, w)
for each edge (u, v) ∈ E[G]
    do if d[v] > d[u] + w(u, v)
          then mark v
               x ← v
               while π[x] is not marked
                   do mark π[x]
                      x ← π[x]
               return marked nodes
return NIL
```



- **w(u, v)** - Weight of the edge (u, v)
- **δ(s, v)** - Weight of the shortest path from s to v
- **v.d** - Estimate of the weight of the shortest path from s to v. Goal is $v.d = \delta(s, v)$
- **v.π** - Pointer to the parent vertex of v in the shortest path from s to v
- **Relaxing** an edge (u, v) updates $v.d$ if $u.d + w(u, v)$ is less than $v.d$.

The Bellman-Ford algorithm can be described in three steps:

1. **Initialize:** For all v, set $v.d = \infty$, $v.\pi$ = NIL. Set $s.d = 0$
2. **Relax:** Relax every edge in G. Repeat for a total of $|V| - 1$ times
3. **Detect Negative Cycles:** Relax every edge in G one more time. If no vertices were updated with a smaller $v.d$ value, then we are done and $v.d = \delta(s, v)$. If at least one vertex was updated, then a negative weight cycle must exist and the $v.d$ values are not necessarily correct. (Optional: Find the negative weight cycle and mark all the vertices on it and reachable from it to have $v.d = -\infty$)

Solution: Run BELLMAN-FORD on the graph and determine whether or not a negative-weight cycle exists. If one exists, then we discover it by finding an edge (u, v) that is part of the cycle. Once we have this, we can use a simple dynamic programming algorithm to find a negative-weight cycle containing u by computing the shortest path from u to every other node containing at most k edges. That is, compute the $n \times n$ matrix S, where $S[k, v]$ is the length of the shortest path from u to v containing at most k edges, and also remember the path along the way (using a separate predecessor matrix). Note that $n = |V|$. This can be done in $O(n^3)$ time. Notice that this is almost exactly what BELLMAN-FORD does, and so we can simply use the predecessor field that it computes. Specifically the following procedure should return a negative-weight cycle if one exists, and return NIL if none does.