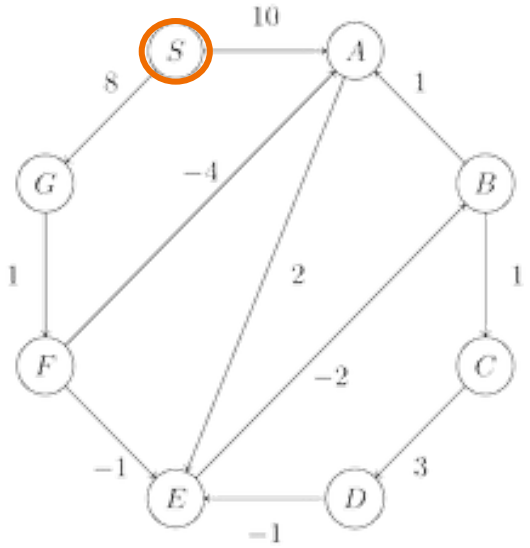
Bellman-Ford Algorithm

Clare Allsopp-Shiner
Ziting Shen

Bellman-Ford algorithm?

- Pathfinding algorithm.
- Can process graphs with negative edge weights.
- Finds shortest distance for all vertices.
- Calculates from all outgoing vertices, replacing values when a shorter path is found, with number of vertices, **n - 1** iterations.

Example;



Number of nodes is 8,
Number of iterations is 7

Starting at node **S**

	Iteration							
Node	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5
C	∞	∞	∞	∞	11	7	6	6
D	∞	∞	∞	∞	∞	14	10	9
E	∞	∞	12	8	7	7	7	7
F	∞	∞	9	9	9	9	9	9
G	∞	8	8	8	8	8	8	8

- If node not reachable, value listed as infinite
- Value at **S** always zero.
- Values of shorter paths replace values of longer paths when found

Pseudocode

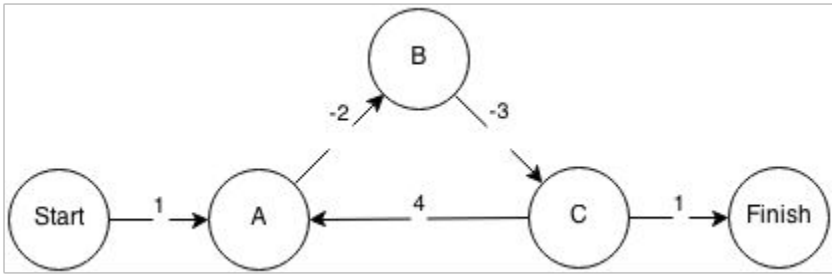
```
function BellmanFord(list vertices, list edges, vertex source)
  // Step 1: initialize data structures
    for each vertex v in vertices:
      distance[v] := infinite
      predecessor[v] := null

   distance[source] := 0

  // Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
    for each edge (u, v) with weight w in edges:
      if distance[u] + w < distance[v]:
        distance[v] := distance[u] + w
        predecessor[v] := u

  // Step 3: check negative-weight cycles
  for each edge (u, v) with weight w in edges:
    if distance[u] + w < distance[v]:
      error "negative-weight cycle"
  return distance[], predecessor[]
```

Negative Cycles



the sum of edge weights around the cycle is negative →

∄a shortest path from the source to some vertices

Proof of Correctness

Lemma. After i repetitions of relaxation:

- Distance(u) ≠ infinity → Distance(u) = the length of some path from s to u
- ∃ path from s to u with at most i edges → Distance(u) ≤ the length of the shortest path from s to u with at most i edges

Proof by induction:

Base case: i = 0

Induction: i = n → i = n+1

Negative Weights? *that's when you want to use Bellman-Ford rather than Dijkstra...

1. Chemistry: heat produced in chemical reactions
2. Transactions: get and lose money
3. Longest path from a single source vertex in acyclic graphs:

- Turn all edge weights to be negative
- Run Bellman-Ford algorithm

4. Longest path w/o repeated edges from a single source vertex in cyclic graphs??

Brute Force!!

Time Complexity

Bellman-Ford: $O(VE) \rightarrow O(V^2 \log V)$

Dijkstra: $O(V^2)$ (w/ list) $\rightarrow O((E + V) \log V)$ (w/ binary heap)

$\rightarrow O(E + V \log V)$ (w/ Fibonacci heap)

$\rightarrow O(E \log \log L)$ (w/ Fibonacci heap)