Bellman-ford Algorithm

The Bellman-ford algorithm solves the single-source shortest paths problem in the general case in which edge weights may be negative.

Given a weighted directed graph $G = (V, E)$ with source S and weight function $w : E \rightarrow R$, the Bellman-ford Algorithm returns a boolean value indicating whether or not there is a negative-weight cycle that is reachable from the source.

If there is no such cycle, the algorithm produces the shortest paths and their weights.

* The algorithm relaxes edges, progressively decreasing an estimate $v.d$ on the weight of a shortest path from the source s to each vertex $v \in V$ until it achieves the actual shortest-path weight $\delta(s, v)$.

* The algorithm returns TRUE if and only if the graph contains no negative-weight cycles that are reachable from the source.

```
BELLMAN-FORD (G, w, S).
1. Initialize-Single-Source (G, S).
2. for i = 1 to |G.V| - 1.
3.     for each edge (u, v) ∈ G.E
4.         RELAX (u, v, w)
5. for each edge (u, v) ∈ G.E
6.     if v.d > u.d + w(u, v).
7.         return FALSE
8. return TRUE.
```

Intialize-Single-Source (G,S).

1. for each vertex $V \in G.V$
2. $V.d = \infty$
3. $V.\pi = NIL.$
4. $S.d = 0.$

RELAX (u,v,w)

1. if $v.d > u.d + w(u,v)$
2. $v.d = u.d + w(u,v)$
3. $v.\pi = u.$

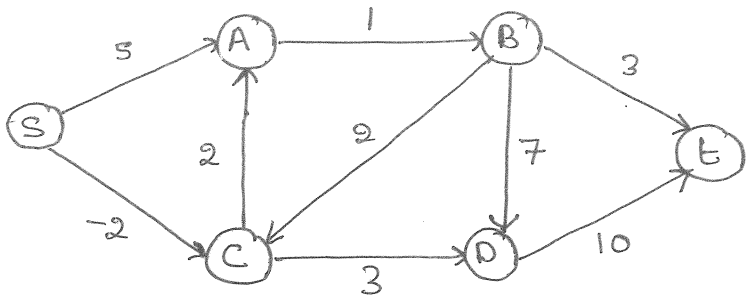
The Bellman-ford algorithm runs in time O(VE), since the initialization in Line 1 take $\Theta(V)$ time, each of the $|V|-1$ passes over the edges in lines 2-4 takes $\Theta(E)$ times, and the for loop of lines 5-7 takes $O(E)$ time.

Shortest-path routing - Application.

- Each link has a cost that reflects.
  - → The length of the link
  - → Delay on the link
  - → congestion
  - → $ cost.
- cost may change with time.
- The length of the route is the sum of the cost along the route.

The shortest path is often used in SNA (social network analysis) to calculate betweenness centrality among others. usually people with the strongests bonds tend to communicate through the shortest path.

Bellman-ford Algorithm



	S	A	B	C	D	t
d[v]	0	∞	∞	∞	∞	∞
$P_i[v]$						

finially answer after All five iteration is

	S	A	B	C	D	t
d[v]	0	~~∞~~ ~~5~~ 0	~~∞~~ ~~5~~ 1	~~∞~~ -2	~~∞~~ ~~8~~ 1	~~∞~~ ~~8~~ 4
$P_i[v]$	0	~~S~~ C	~~A~~ A	S	~~B~~ C	~~D~~ B.