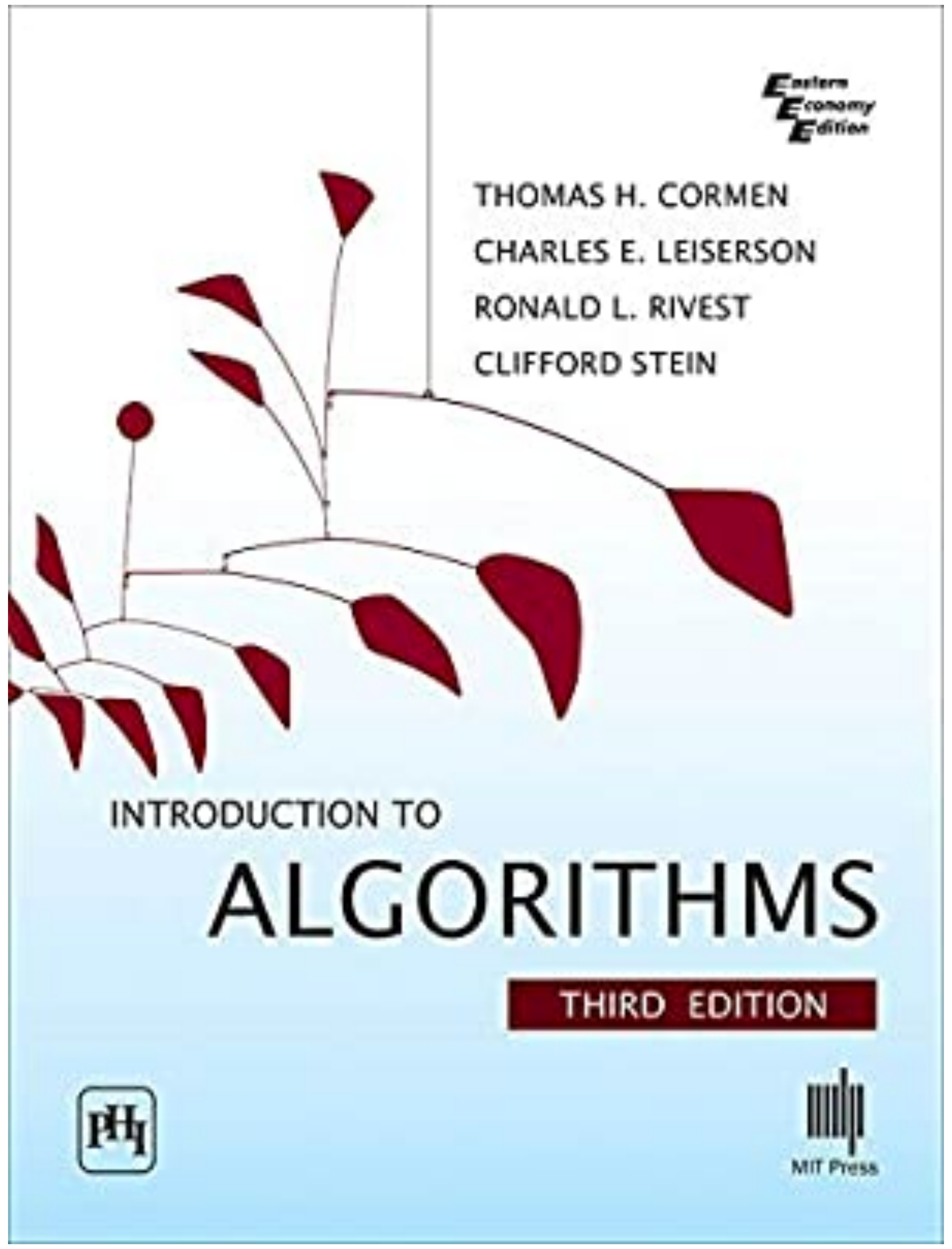
Lecture 3
Shortest Paths

Overview

- Weighted Graphs
- General Approach
- Negative Edges
- Optimal Substructure

Readings



CLRS Chapter 24 (Intro)

Motivation

The most common problem

Shortest way to drive from A to B Google maps “get directions”

Formulation:

Problem on a weighted graph G(V, E), $W: E \to R$

Two algorithms:

Dijkstra O($V \lg V + E$) assumes non-negative edge weights

Bellman Ford O($V * E$) is a general algorithm

Motivation Definitions

Model as a weighted graph G(V, E), W : E → R

– V = vertices (street intersections)

– E = edges (street, roads); directed edges (one way roads)

– W(U, V) = weight of edge from u to v (distance, toll)

$$\text{path } p \; = < v_0, v_1, \ldots v_k >$$

$$(v_i, v_{i+1}) \; \in \; E \quad \text{for} \quad 0 \leq i < k$$

$$w(p) = \sum_{i=0}^{k-1} \; w(v_i, v_{i+1})$$

Shortest path weight

Notation:

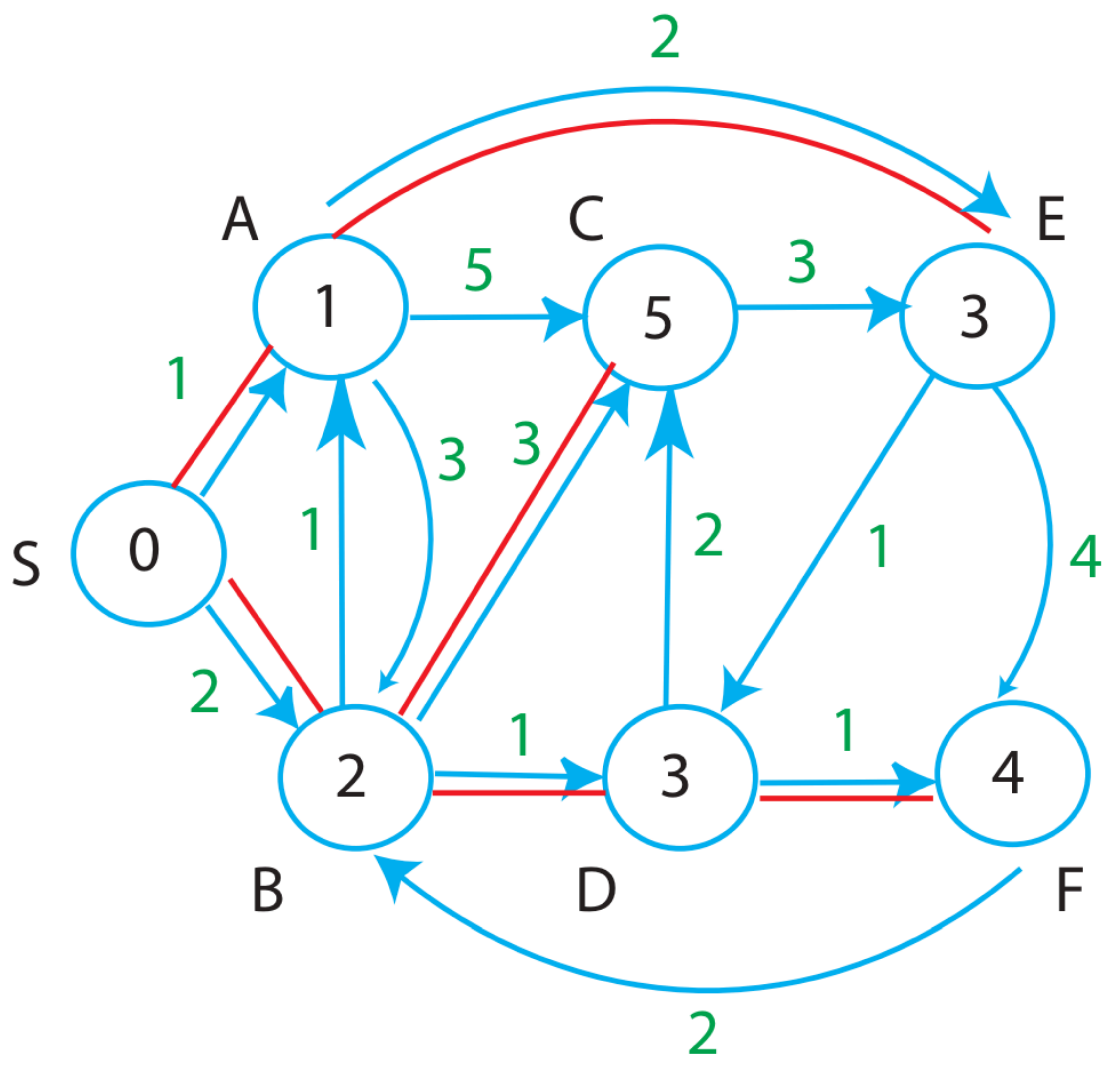
$v_0 \xrightarrow{p} v_k$ means p is a path from v_0 to v_k. (v_0) is a path from v_0 to v_0 of weight 0.

Definition:

Shortest path weight from u to v as

$$\delta(u,v) = \begin{cases} \min \left\{ w(p) : u \xrightarrow{p} v \right\} & \text{if } \exists \text{ any such path} \\ \infty & \text{otherwise} \quad (v \text{ unreachable from } u) \end{cases}$$

Example



Single Source Shortest Paths:

Given $G = (V, E), w$ and a source vertex S, find $\delta(S, V)$ [and the best path] from S to each $v \in V$.

Data structures:

$$\begin{aligned} d[v] &= \text{value inside circle} \\ &= \left\{ \begin{array}{ll} 0 & \text{if } v = s \\ \infty & \text{otherwise} \end{array} \right\} \Longleftarrow \quad \text{initially} \\ &= \delta(s, v) \Longleftarrow \quad \text{at end} \\ d[v] &\geq \delta(s, v) \quad \text{at all times} \end{aligned}$$

$d[v]$ decreases as we find better paths to v, see Figure 1.
$\Pi[v]$ = predecessor on best path to v, $\Pi[s] = \text{NIL}$

Negative-Weight Edges

- Natural in some applications. Really?
- Some algorithms disallow negative weight edges (e.g., Dijkstra)
- If you have negative weight edges, you might also have negative weight cycles $\Longrightarrow$ may make certain shortest paths undefined!

Example

B → D → C → B (origin) has weight
$-6 + 2 + 3 = -1 < 0$!

Shortest path S → C (or B, D, E) is undefined! Why?

If negative weight edges are present, s.p. algorithm should find negative weight cycles (e.g., Bellman Ford).

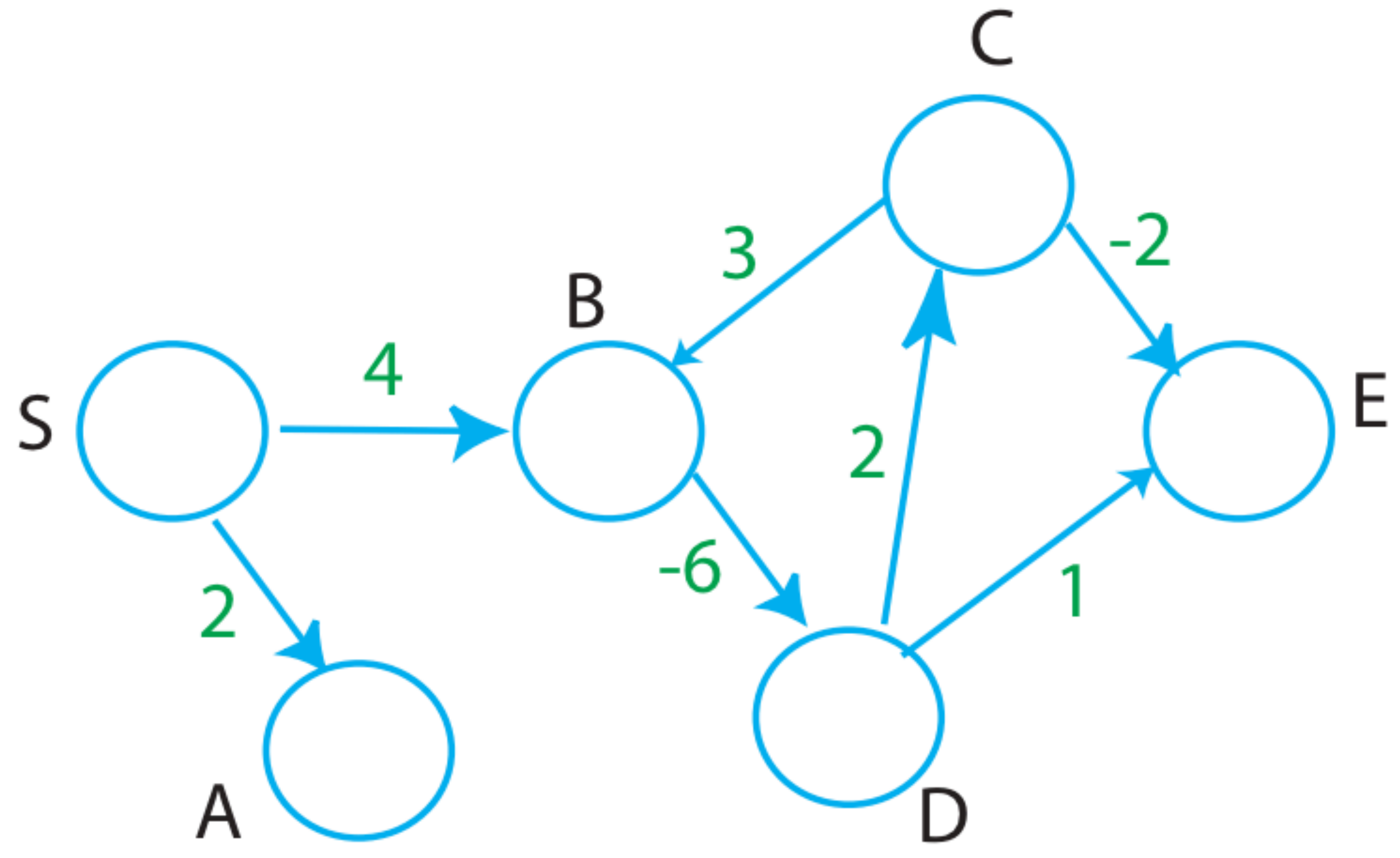


Figure 2: Negative-weight Edges.

General structure of S.P. Algorithms (no negative cycles)

Initialize: for $v \in V$: $d[v] \leftarrow \infty$

$\Pi[v] \leftarrow$ NIL

$d[S] \leftarrow 0$

Main: repeat

select edge (u, v) [somehow]

"Relax" edge (u, v)

$$\left[\begin{array}{l} \text{if } d[v] > d[u] + w(u,v) : \\ \quad d[v] \leftarrow d[u] + w(u,v) \\ \quad \pi[v] \leftarrow u \end{array}\right.$$

until all edges have $d[v] \leq d[u] + w(u, v)$