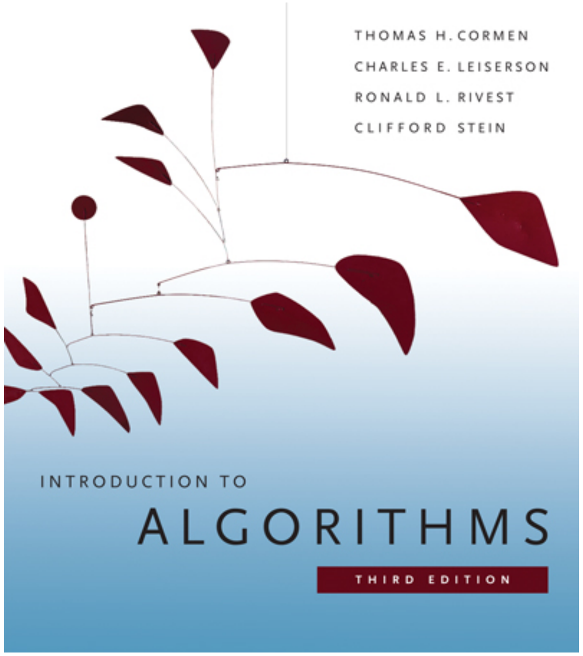
6.006- *Introduction to Algorithms*

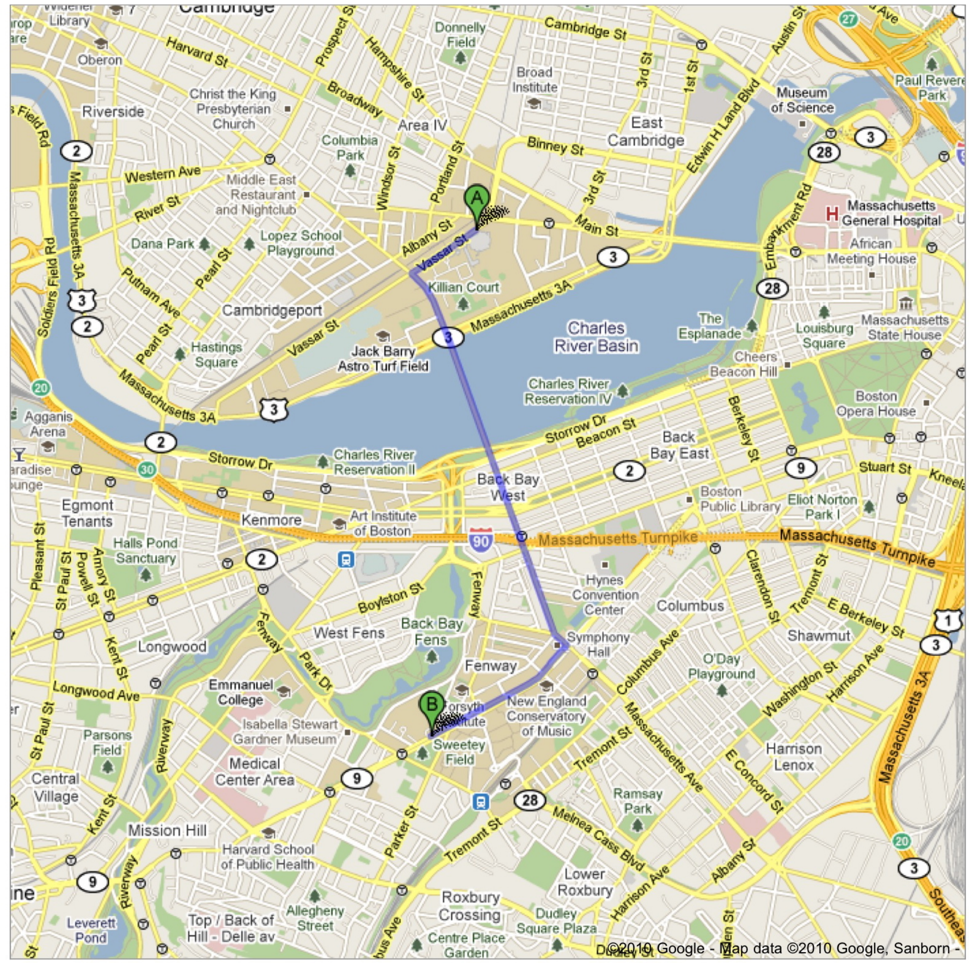


Lecture 14

Prof. Patrick Jaillet

Lecture overview

Shortest paths
- Definition
- Generic algorithm
- Some properties

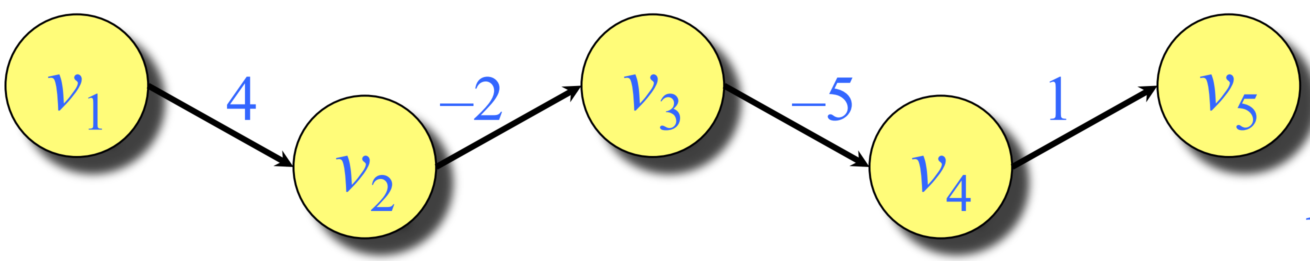


Readings: CLRS 24 (intro)

Paths in graphs

Consider a directed graph $G = (V, E)$ with edge-weight function $w : E \to \mathbb{R}$.
The ***weight*** of path $p = v_1 \to v_2 \to \cdots \to v_k$ is defined to be the sum of all weights on the path, i.e., $w(p) = w(v_1, v_2) + \ldots + w(v_{k-1}, v_k)$

Example:



$w(p) = –2$

Shortest paths - definition

- A ***shortest path*** from u to v is a path of minimum weight from u to v
- The ***shortest-path weight*** $\delta(u, v)$ from u to v is defined as the weight of any shortest path from u to v

Special cases:

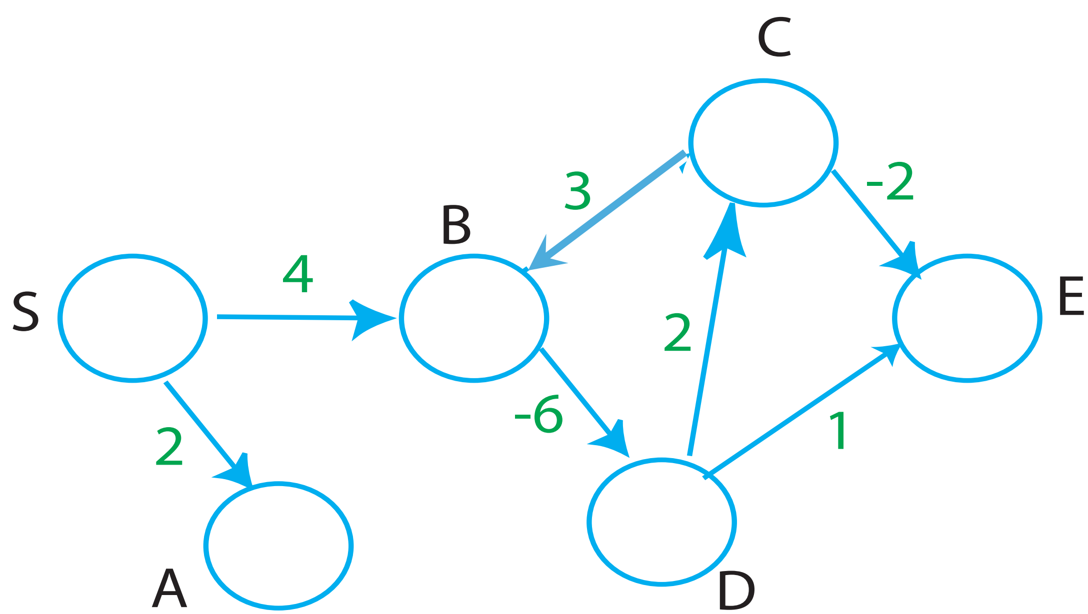
1. no path from u to v exists: $\delta(u, v) = \infty$

   "you cannot get there from here"

2. negative weight cycles…=> undefined

Well-definedness of shortest paths

If a graph G contains a negative-weight cycle, then some shortest paths may not exist.



Negative weight cycles: $\delta(s, c)$ undefined (algorithm should detect such situations)

Single source shortest path problem

Problem: Given a directed graph $G = (V, E)$ with edge-weight function w, and a node s, find $\delta(s, v)$ (and a corresponding path) for all v in V

Today:

• Generic algorithm and some structural properties

Next three lectures:

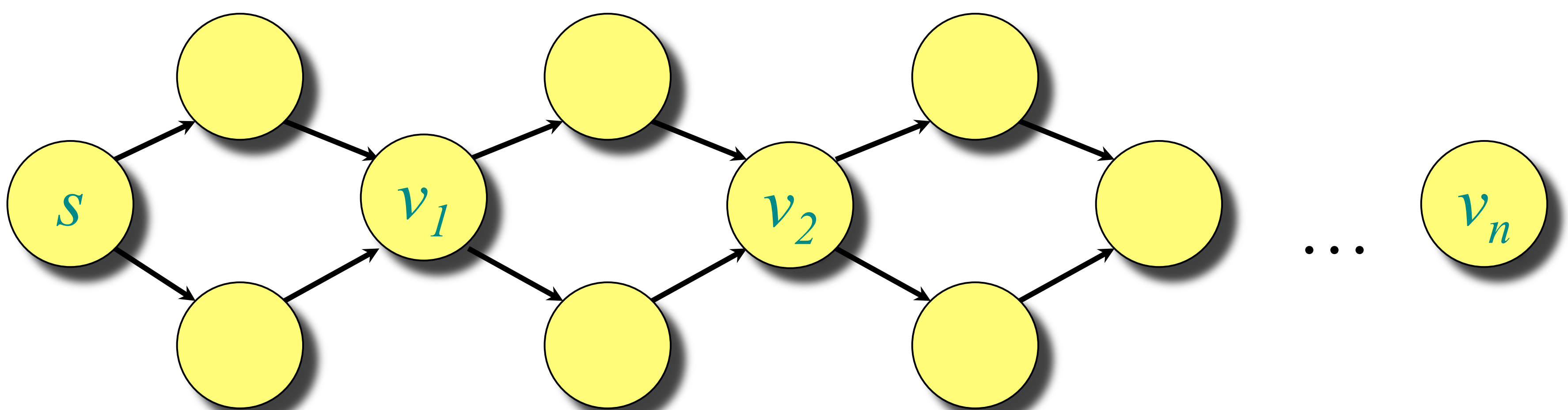
• Bellman-Ford: deals with negative weights

• Dijkstra algorithm: fast and faster, but assumes non-negative weights

Digression

Question: why can't we just enumerate all paths to find the shortest one ?

Answer: there can be exponentially many of them!



2^n different paths from s to v_n , $3n+1$ vertices

Useful data structures

- $d[v]$ = length of best path from s to v so far
- initialization $d[s] = 0$; $d[v] = \infty$ otherwise
- at any step update $d[v]$ so that $d[v] \geq \delta(s, v)$

- $\pi[v]$ = predecessor of v on a best path so far
- initialization $\pi[s] = \mathrm{s}$; $\pi[v] = \mathrm{nil}$ otherwise

A generic algorithm

$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ s
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ nil

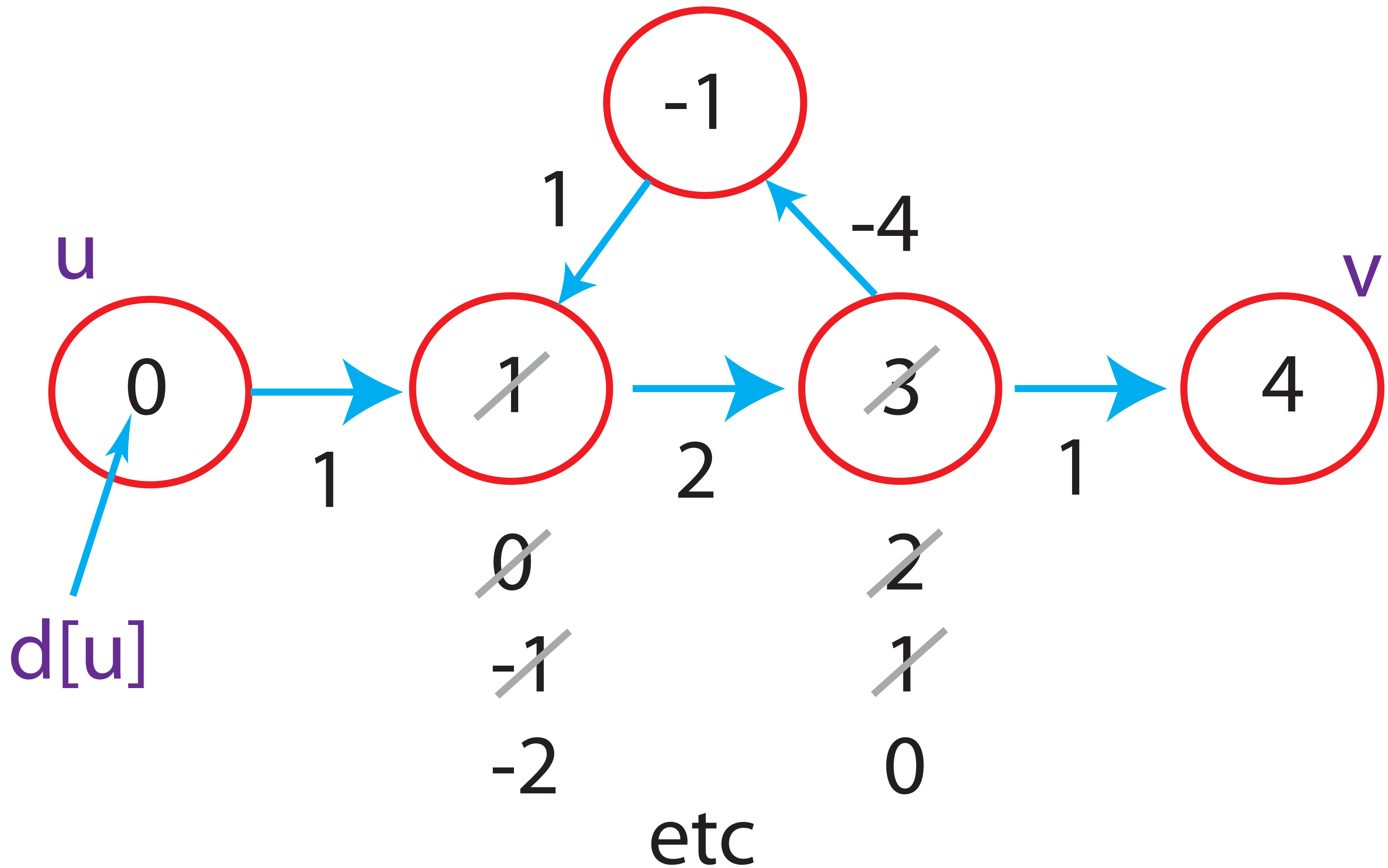
} ***initialization***

while there is an edge $(u, v) \in E$ s. t.
  $d[v] > d[u] + w(u, v)$ **do**
 select one such edge "somehow"
 set $d[v] \leftarrow d[u] + w(u, v)$
   $\pi[v] \leftarrow$ u
endwhile

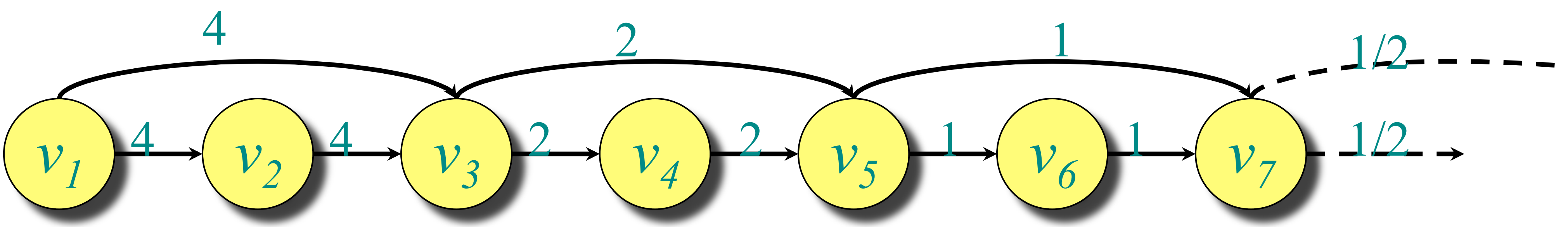
} ***relaxation step***

(the trick is in the "somehow" step…)

Will not stop when negative cycles



What if no negative cycle ....



	4	8	10	12	13	14
						13
				10	11	12
						11
		4	6	8	9	10
						9
				6	7	8
						7

Analysis for previous example ...

Let:

- $n+1$ be the number of **vertices**
- $T(n)$ number of relaxations on $v_1,\ldots v_{n+1}$

Relax (v_1,v_2) and (v_2,v_3)

Relax (v_1,v_3)

We have:

$$T(n)= 2+T(n-2)+1+T(n-2) = 2\,T(n-2) +3$$

Recursion on $v_3,\ldots v_{n+1}$

$$T(n)=\Theta(2^{n/2})$$

Conclusion: need to be careful how we relax

Another digression ....

Exponential Bad

$$T(n) = C_1 + C_2 T(n - C_3)$$

if $C_2 > 1$, trouble!
Divide & Explode

Polynomial Good

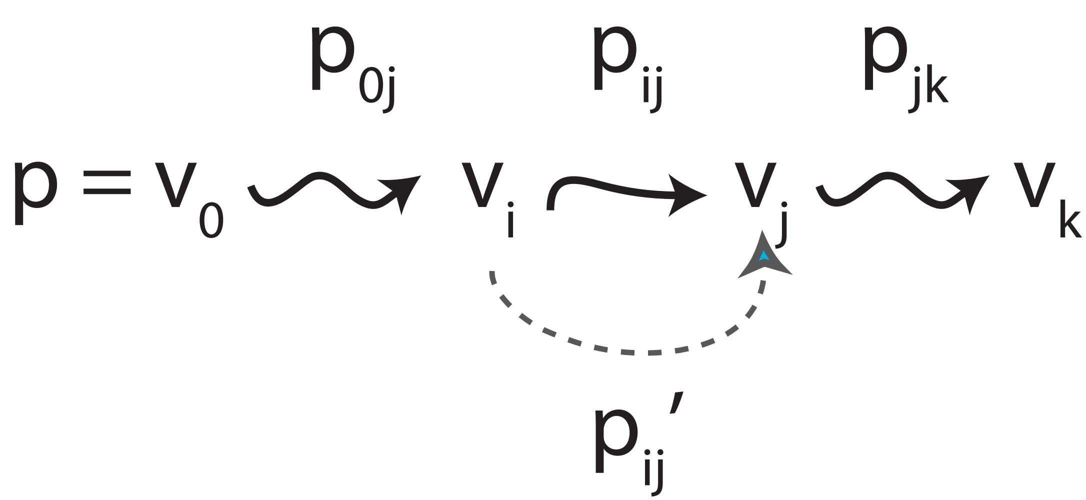
$$T(n) = C_1 + C_2 T(n / C_3)$$

$C_2 > 1$ okay provided $C_3 > 1$
if $C_3 > 1$
Divide & Conquer

Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.

Proof. By contradiction ...

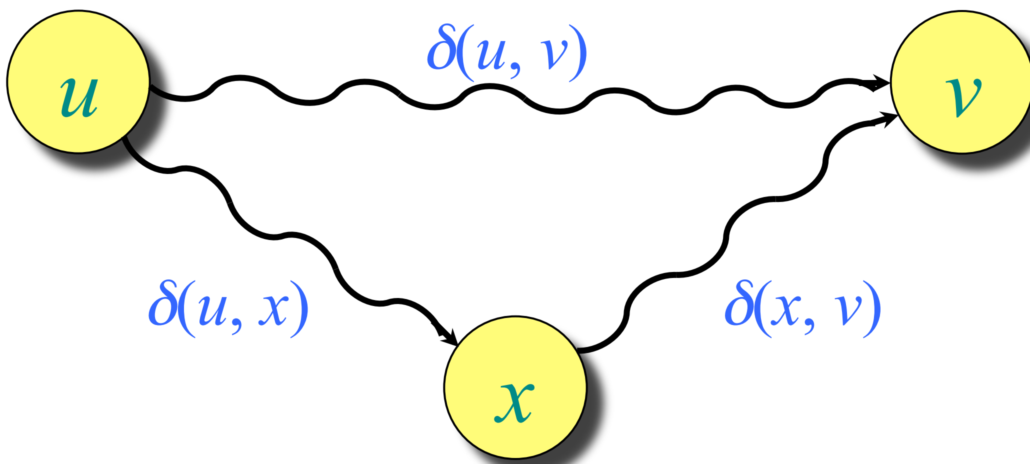


Triangle inequality

Theorem. For all $u, v, x \in V$, we have

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Proof.



□