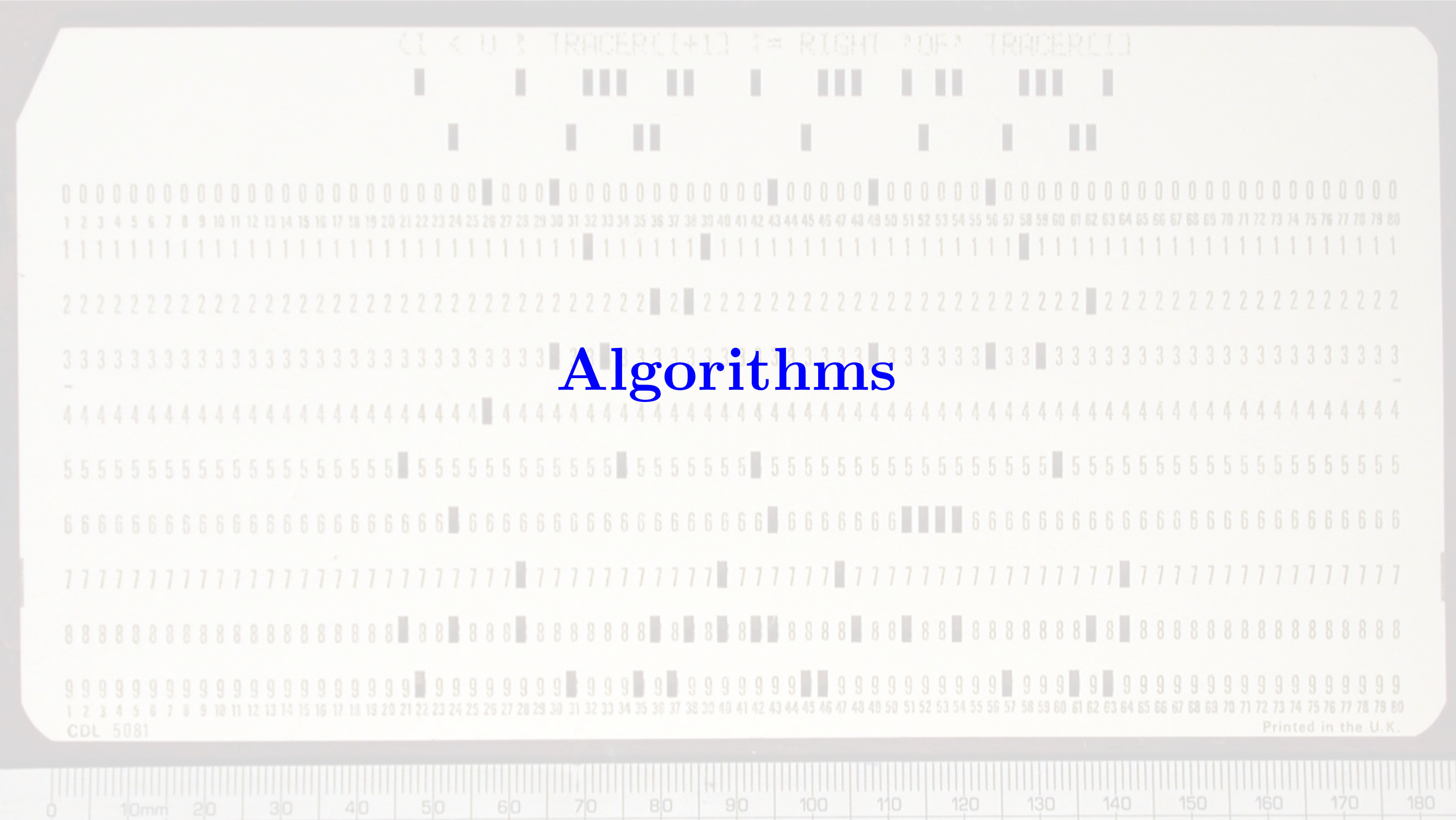
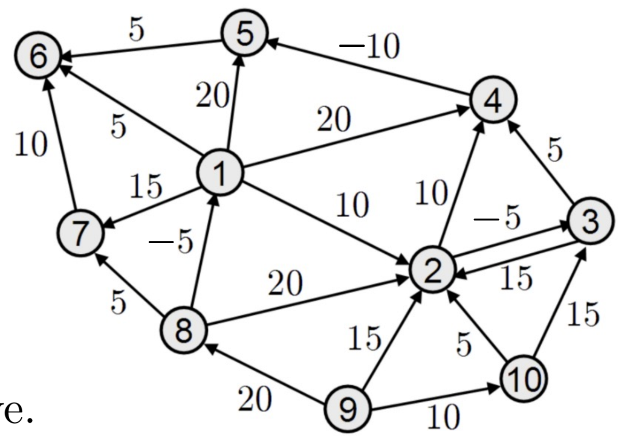
Algorithms



Single Source Shortest Paths (SSSP)

Given directed graph with weights on the edges, the goal is to compute all shortest paths from a specific node s to all other nodes.



The weights on the edges can be positive or negative.

Single Source Shortest Paths (SSSP)

Recall Dijkstra:

Input: Directed graph $G = (V, E)$, edge lengths c_e for each $e \in E$, source vertex $s \in V$.

Goal: For every destination $v \in V$, compute the length (sum of edge costs) of a shortest s-v path.

DIJKSTRA ALGORITHM

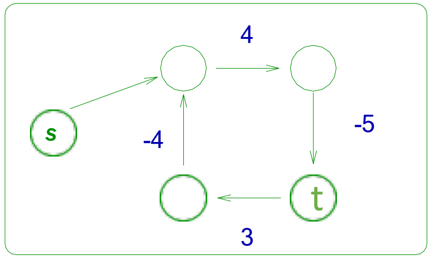
Facts:

- $O(m+n)logn$ (n = number of vertices, m = number of edges)
- $O(m \log n)$ running time using heaps

- Not always correct with negative edge lengths
  - Application: if edges $\rightarrowtail$ financial transactions

Solution: The Bellman-Ford algorithm

NEGATIVE CYCLES



Question: How to define shortest path when G has a negative cycle?

Solution #1: Compute the shortest s-v path, in the presence of cycles.

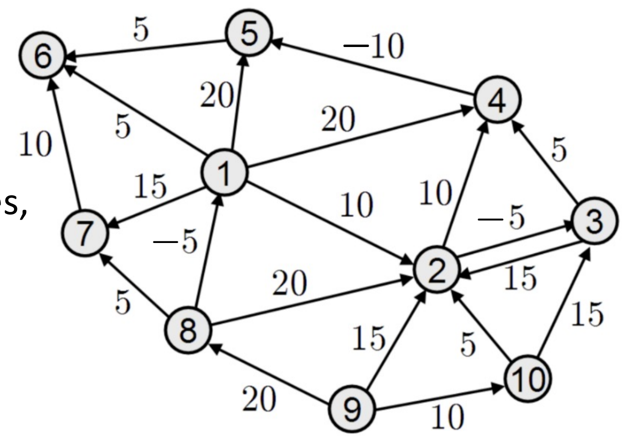
Problem: Undefined or $-\infty$. [will keep traversing negative cycle]

Solution #2: Compute shortest cycle-free s-v path [in the presence of –ve cycles]

Problem: NP-hard (no polynomial algorithm) [to be discussed later]

Solution #3: Assume input graph has no negative cycles [overall cycle cost].

Single Source Shortest Paths (SSSP)



Given a graph G with n nodes and m edges and no negative cycles, which of the following statement is the strongest:

1. For every v, there is a shortest path s-v with $\leq$n-1 edges
2. For every v, there is a shortest path s-v with $\leq$n edges
3. For every v, there is a shortest path s-v with $\leq$ m edges
4. No limit can be applied on the shortest path

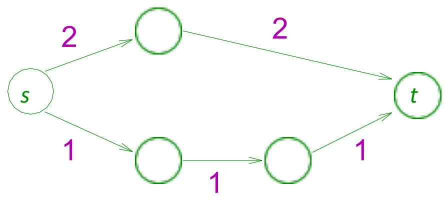
Single Source Shortest Paths (SSSP)

Input: Directed graph $G = (V, E)$, edge lengths c_e [possibly negative], source vertex $s \in V$.

Goal: For all destinations $v \in V$, compute the length of a shortest s-v path

DYNAMIC PROG SOLUTION

- Intuition: Requires some kind of ordering to divide the problem into smaller subproblems
- Issue: Not clear how to define “smaller” & “larger” subproblems.
- Solution: Exploit sequential nature of paths. Subpath of a shortest path should itself be shortest.
- Key idea: Artificially restrict the number of edges in a path.
- Subproblem size $\Longleftrightarrow$ Number of permitted edges
- Example:
  - Shortest path with 2 or less edges from s to t
  - 3 or less



OPTIMAL SUBSTRUCTURE

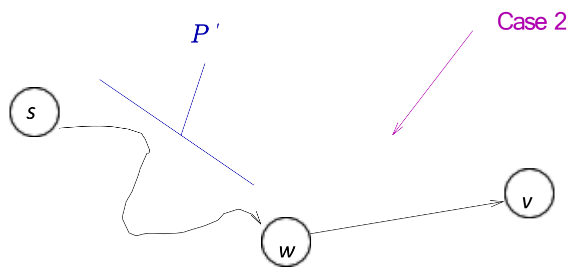
Lemma: Let $G = (V, E)$ be a directed graph with edge lengths c_e and source vertex s.

For every $v \in V$, $i \in \{1, 2, \ldots\}$, let

P = shortest s-v path <u>with at most i edges</u>. (Cycles are permitted.)

Case 1: If P has $\leq (i-1)$ edges, it is a shortest s-v path with $\leq (i-1)$ edges.

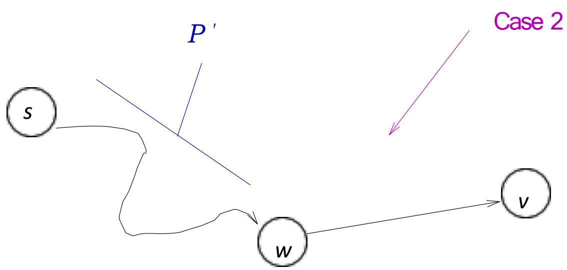
Case 2: If P has i edges with last hop (w, v), then P' is a shortest s-w path with $\leq (i-1)$ edges.



OPTIMAL SUBSTRUCTURE

Case 1: By (obvious) contradiction [assume there is a shorter i-1 path, H, then H<P, so H should be the shortest i path-Contradiction].

Case 2: If Q (from s to w, $\leq (i-1)$ edges) is shorter than P^J then $Q + (w, v)$ (from s to v, $\leq i$ edges) is shorter than $P^J + (w, v)$ (= P) which contradicts the optimality of P. QED!



Single Source Shortest Paths (SSSP)

Given all shortest paths of length $\leq$ i-1 from S.
How many candidate paths of legth $\leq$ i do we have to a node v:

1. 5
2. 1+ in-degree(v)
3. n
4. m
5. n-1

RECURRENCE

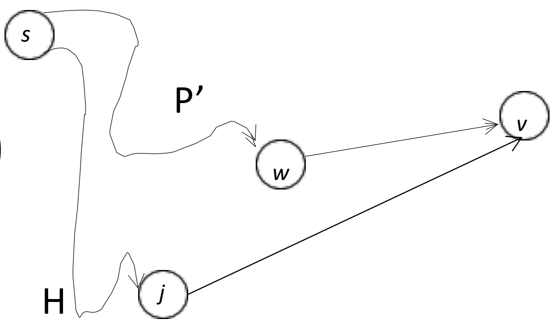
Notation: Let $L_{i,v}$ = minimum length of a s-v path with $\leq i$ edges.

- Defined as $+\infty$ if no s-v paths with $\leq i$ edges

Recurrence: For every $v \in V$, $i \in \{1,2,\ldots\}$

$$L_{i,v} = \min \begin{cases} L_{(i-1),v} & \text{Case 1} \\ \min_{(w,v)\in E}\{L_{(i-1),w} + c_{wv}\} & \text{Case 2} \end{cases}$$

Correctness: Brute-force search from the only (1+in-deg(v)) candidates (by the optimal substructure lemma).



RECURRENCE

Now: Recall that graph G has no negative cycles.

$\Rightarrow$ Shortest paths do not have cycles

[removing a cycle only decreases length]

$\Rightarrow$ Have $\leq (n-1)$ edges

Point: If G has no negative cycle, only need to solve subproblems up to $i = n - 1$.

Subproblems: Compute $L_{i,v}$ for all $i \in \{0, 1, \ldots, n-1\}$ and all $v \in V$.

The Bellman-Ford Algorithm

Let A = 2-D array (indexed by i and v)

	1	…	…	n-1
v_1				
. . .				
v_n				

Base case: $A[s, 0] = 0$; $A[v, 0] = +\infty$ for all $v \neq s$.

For $i = 1, 2, \ldots, n-1$

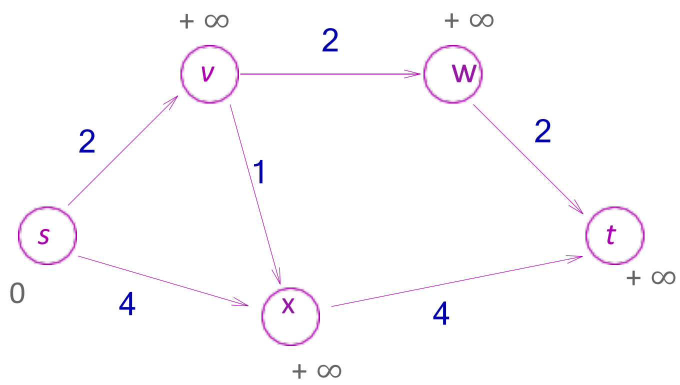
For each $v \in V$

$$A[v, i] = \min \begin{cases} A[v, i-1] \\ \min_{(w,v) \in E} \{A[w, i-1] + c_{wv}\} \end{cases}$$

As discussed: If G has no negative cycle, then algorithm is correct [with final answers = $A[v, n-1]$'s]

Example

$$A[v, i] = \min \begin{cases} A[v, i-1] \\ \min_{(w,v)\in E}\{A[w, i-1] + c_{wv}\} \end{cases}$$



$i = 0$
$i = 1$
$i = 2$
$i = 3$
$i = 4$