Bellman-Ford Algorithm

Learning Goals

- Design pseudocode for a dynamic programming alg. [DP2]
- PA2

Shortest Path Problem

Input: $G = (V, E)$, $w : E \to \mathbb{R}$, $s, t \in V$, $|V| = n$, $|E| = m$, directed, no negative cycles

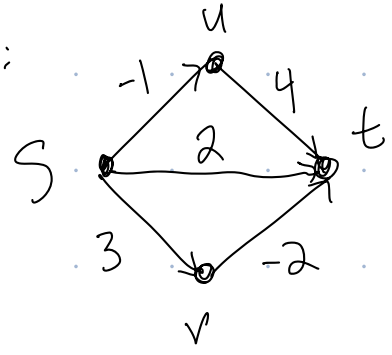
Output: Path P from s to t in G

e.g. $P = ((s,u), (u,v), \cdots (r,t))$ ← sequence of connected edges

such that $L(P) = \sum_{e \in P} w(e)$ is minimized

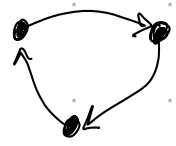
Example:

G:



Shortest Path:

Negative Cycle:



Various Approaches

- Brute Force
- Greedy
- Dynamic

Applications:

-

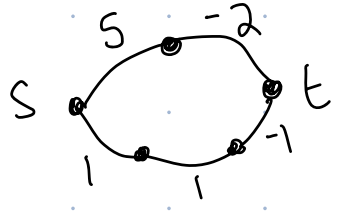
-

-

Defining Subproblems

Clever idea: $P_{u,i}$ = shortest path from s to u that uses at most i edges

What is $L(P_{t,1})$, $L(P_{t,2})$, $L(P_{t,3})$ for the following graph



A) $\infty, 3, 1$ B) $\infty, 2, 3$ C) $0, 2, 3$ D) $0, 3, 1$

Last choice a strategy makes?

Designing a Dynamic Prog. Alg.

- Recurrence relation for $P_{u,i}$ = shortest path from s to u that uses at most i edges

$$P_{u,i} = \begin{cases} \text{\_\_\_\_\_} & \text{if shortest path with } i \text{ edges goes through } v \text{ prior to } u \\ \text{\_\_\_\_\_} & \text{if shortest path with } i \text{ edges goes through } w \text{ prior to } u \\ \vdots & \\ \text{\_\_\_\_\_} & \text{if shortest path to } u \text{ uses less than } i \text{ edges} \end{cases}$$

- Recurrence relation for objective function ($w(u,u) = 0$, $w(u,v) = \infty$ if no edge)

$$A(u,i) = \begin{cases} \text{\_\_\_\_\_} \\ \text{Base case: } \text{\_\_\_\_\_} \end{cases}$$

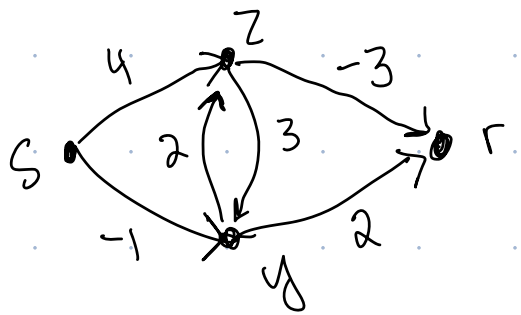
Helpful notation $\rightarrow \min_{u \in S} \{f(u)\}$

- Write Pseudocode to fill in A

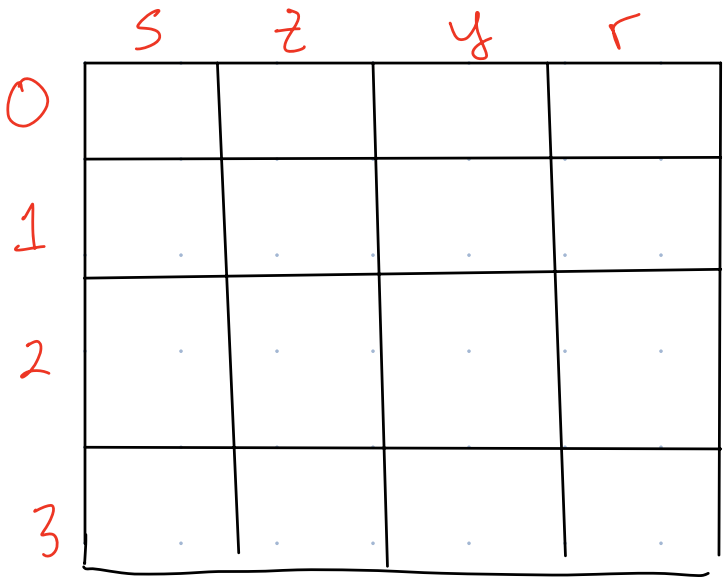
Input: $n \times n$ array w where $w[u,v]$ = weight of edge (u,v), $s, t \in [n]$.
$w[u,v] = \infty$ if no edge. $w[u,u] = 0 \ \forall \ u \in [n]$

Output: ?×? array A s.t. $A[u,i] = L(P_{u,i})$

Use your code to create A for the graph



A



What is the Runtime of Bellman-Ford?

A) $O(n)$ B) $O(n^2)$ C) $O(n^3)$ D) $O(nm)$ (n = # vertices, m = # edges)

Recall

Recurrence relation for objective function ($w(u,u) = 0$, $w(u,v) = \infty$ if no edge)

$$L(P_{u,i}) = \begin{cases} \min\left\{ \min_{\substack{v \in V \\ v \neq u}} \left\{ L(P_{v,i-1}) + w(v,u) \right\}, L(P_{u,i-1}) \right\} \\ \text{Base case: } \begin{array}{l} L(P_{u,0}) = \infty \text{ if } u \neq s \\ L(P_{s,0}) = 0 \end{array} \end{cases}$$

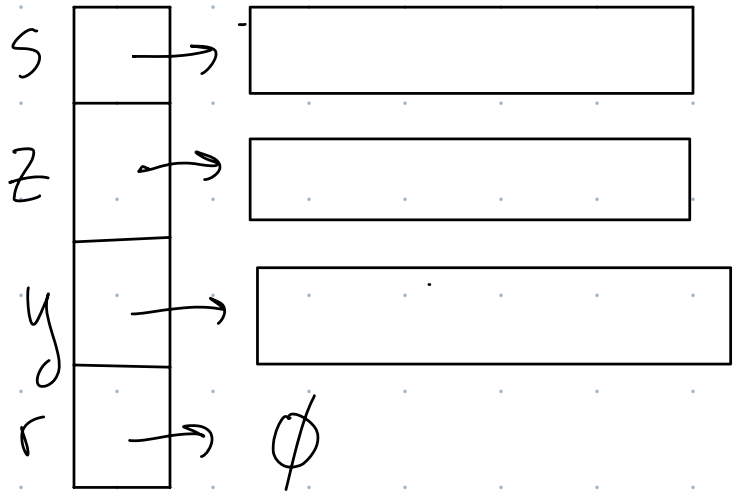
Don't want to check every $v \in V$. Only want to check $v : (v,u) \in E$.

Adjacency Matrix

	s	z	y	r
s	∞	4	-1	∞
z	∞	∞	3	-3
y	∞	2	∞	2
r	∞	∞	∞	∞

s→r weight

Adjacency List



list of vertices with edges leaving a vertex

Reverse Adjacency List

list of vertices with edges entering a vertex

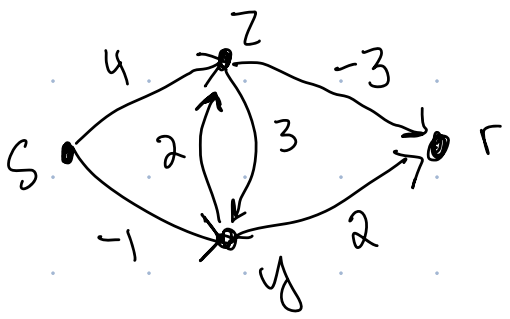
$w[s] = \emptyset$

$w[z] = [(s,4), (y,2)]$

$w[r] = [(z,-3), (y,2)]$

$w[y] = [(s,-1), (z,3)]$

e.g.



Initialize $A \leftarrow n \times n$ array filled with ∞'s.

$A[s,0] \leftarrow 0$

For $i \leftarrow 1$ to $n-1$:

  For $u \leftarrow 1$ to n:

    $A[u,i] = A[u,i-1]$

    For $v \leftarrow 1$ to n:

      If $A[u,i] > A[v,i-1] + w[v,u]$

        $A[u,i] \leftarrow A[v,i-1] + w[v,u]$

What is the Runtime of Bellman-Ford with Reverse Adj Matrix?

A) $O(n)$ B) $O(n^2)$ C) $O(n^3)$ D) $O(nm + n^2)$

(n: # vertices, m: # edges)

Initialize $A \leftarrow n \times n$ array filled with ∞'s.

$A[s,0] \leftarrow 0$

For $i \leftarrow 1$ to $n-1$:

  For $u \leftarrow 1$ to n:

    $A[u,i] = A[u,i-1]$

    ~~For $v \leftarrow 1$ to n~~ For $v : (v,u) \in E$:

      If $A[u,i] > A[v,i-1] + w[v,u]$

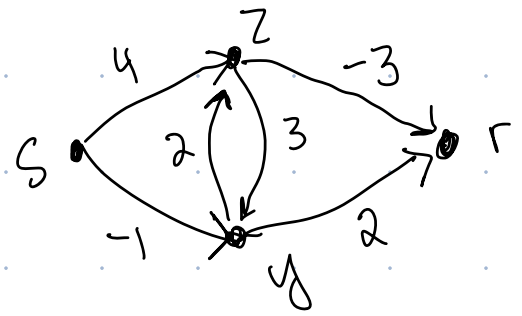
        $A[u,i] \leftarrow A[v,i-1] + w[v,u]$

$W[s] = \phi$

$W[z] = [(s, 4), (y, 2)]$

$W[r] = [(z, -3), (y, 2)]$

$W[y] = [(s, -1), (z, 3)]$



Distributed Bellman-Ford

Vertex v can calculate $A[v, i]$ if gets passed $A[u, i-1]$ info from neighbor u: $(u, v) \in E$.

Then it passes $A[v, i]$ onto its neighbors.

Useful for network

Bartering	Stakeholders	Well-Being	Autonomy	Justice
Arbitrage				