Bellman-Ford Algorithm

Learning Goals

- Describe the Shortest Path problem + discuss applications ✓
- Design a dynamic programming alg for shortest path ~

Announcements

- MiddDev Soft. Eng. Alum Talk: Sun @ 1 in 755 HS 203.
- A4 (Dyn. Prog.)
- JTerm Workshop on Leetcode + coding interview prep

Exit Tickets

- 2022: Integer ± weights ⇒ randomized $O(m \log^8(n) \log W)$ ($-W$ is smallest neg weight)
- Reverse Adjacency list
- Avoiding negative cycles
- Dense to Sparse?

Shortest Path Problem

(directed)

Input: $G=(V,E)$, $w: E \to \mathbb{R}$, $s, t \in V$ $\quad (|V|=n, |E|=m)$

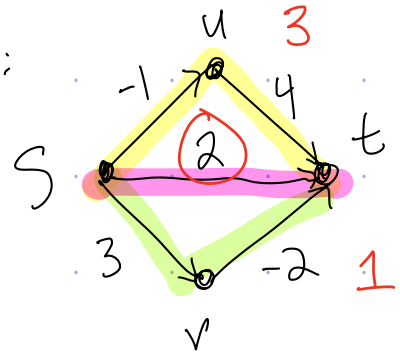
Output: Path from s to t in G

ex: $P = ((s,u),(u,v), \dots (r,t))$

s.t. $L(P) = \sum_{e \in P} w(e)$ is minimized

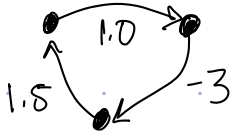
Example:

G:



Shortest Path: $((s,v),(v,t))$

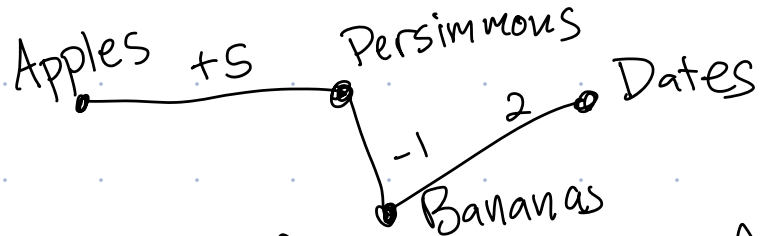
Negative Cycle:



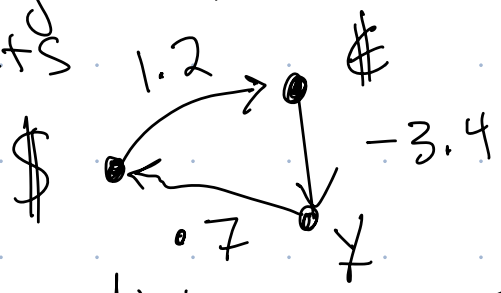
Various Approaches

- Greedy → Dijkstra's → edges have positive weight
- Brute Force → BFS → edges all have weight 1.
- Dynamic → Bellman-Ford → edge have neg weight
  → fails if path must avoid neg cycles → NP-Hard

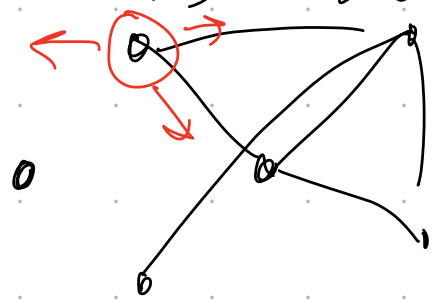
Applications: Bartering



Arbitrage: Finding inefficiencies in financial markets



Data Routing in Networks - Distributed Graph



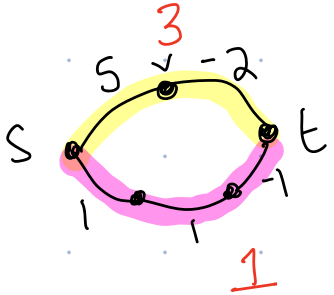
Cold Weather Plans?

Bartering	Stakeholders	Well-Being	Autonomy	Justice
	Merchants	:(taken advantage of, :) get customers	Won't every know interacting with alg	unjust
	Customer / Buyers	:) $ $	:)	Wealthy might have more success → more $ → access to algorithm
Arbitrage				
	Traders	:) freeish $	:)	Richer traders have access to better servers,
	Environment	:(	:(	location location of data centers

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



$L(P_{t,2}) = 3$

$P_{t,2} = ((s,v),(v,t))$

no path, no edge

A) $\infty, 3, 1$ ✓ ✓

B) $\infty, 2, 3$

C) $0, 2, 3$

D) $0, 3, 1$ ✓ ✓

Last choice a strategy makes? Last vertex prior to t.

$s \to \to \to v \to t$

$s \to \to u \to t$

$s \to \to \to w \to t$

$s \to \ldots$

Designing a Dynamic Prog. Alg.

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

$$P_{u,i} = \begin{cases} P_{v,i-1} + (v,u) & \text{if shortest path with } i \text{ edges goes through } v \text{ directly prior to } u \\ P_{w,i-1} + (w,u) & \text{if shortest path with } i \text{ edges goes through } w \text{ directly prior to } u \\ P_{r,i-1} + (r,u) & \\ \vdots & \\ P_{u,i-1} & \text{if shortest path to } u \text{ uses less than } i \text{ edges} \end{cases}$$

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

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

$A[u,i] = L(P(u,i))$

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

- Write Pseudocode to fill in A with $L(P_{u,i})$ values

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. $A[u,i] = L(P_{u,i})$ $\quad i = 0 \to n-1$, $u = 1 \to n$

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

$A[s,0] = 0$ // Base Cases

For $i \leftarrow 1$ to $n-1$: ← $i=3$

  For $u \leftarrow 1$ to n: // vertices labelled from 1 to n. ← $u=t$

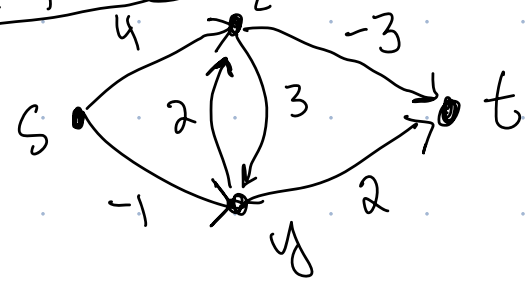
    $A[u,i] \leftarrow A[u,i-1]$

    For $v \leftarrow 1$ to n s.t. $v \neq u$: ← 0

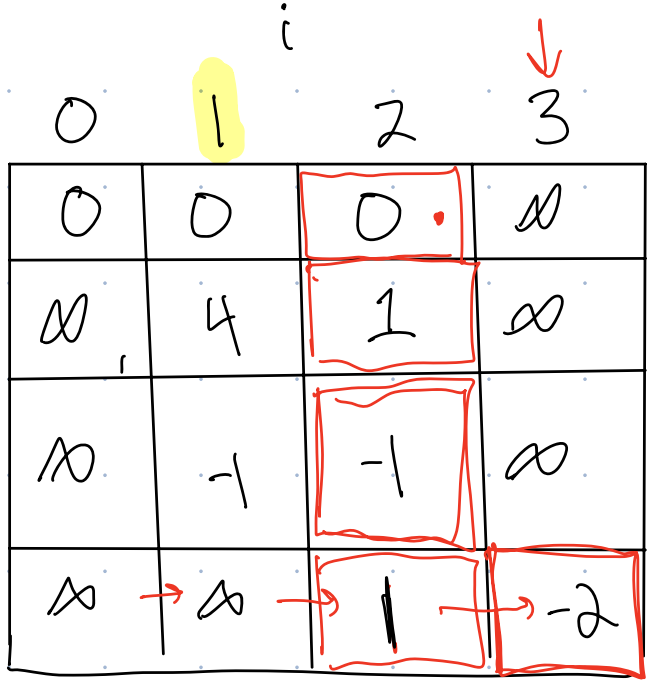
      If $A[u,i] > A[v,i-1] + w[v,u]$:

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

Use your code to create A for the graph



A



$A[s,0] + w[s,z]$

$0 + 4 = 4$

What is the Runtime of Bellman-Ford?

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

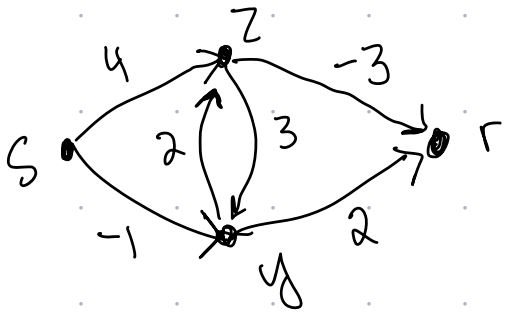
n: # vertices

m: # edges

Recall

Recurrence relation for objective function $\begin{pmatrix} w(u,u) = 0 \\ w(u,v) = \infty \text{ if no edge} \end{pmatrix}$.

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



Reverse Adjacency List

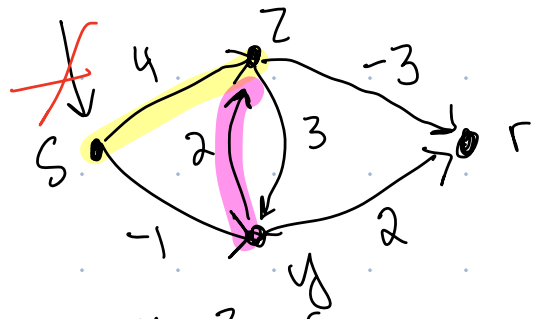
$W[s] = \emptyset$

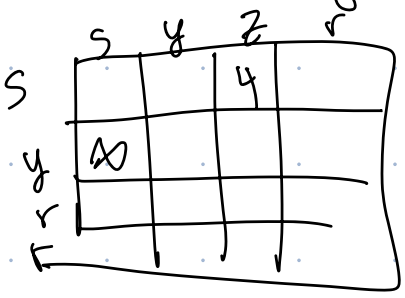
$W[z] = [(s,4), (y,2)]$

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

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

e.g. $W[r,2] = (y,2)$





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 ~~$*$~~ ~~1 to n~~ $v \in W[u]$

      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 $n-1$: $\leftarrow O(n)$

  For $u \leftarrow 1$ to n:

    $A[u,i] \leftarrow A[u,i-1]$ $\leftarrow O(n^2) \rightarrow O(m)$

    For $v \in w[u]$

      If $A[u,i] > A[v,i-1] + w[v,u]$ $\leftarrow O(m)$

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

Most edges possible

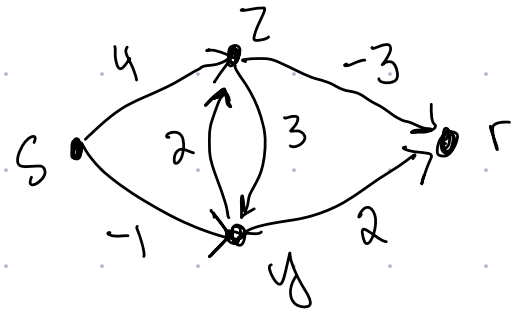
$m = n^2$

$n \cdot m = n^3$

Sparse graphs

$m = O(n)$

$m = O(\sqrt{n})$



$W[s] = \phi$

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

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

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

$W[k] = [$

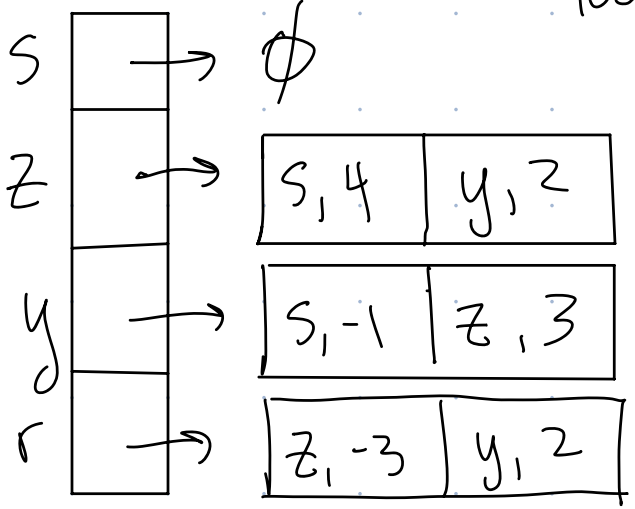
Adjacency Matrix

$A[s,r]$: $s \to r$ weight

	s	z	y	r
s	0	4	-1	∞
z	∞	0	3	-3
y	∞	2	0	2
r	∞	∞	∞	0

Reverse Adj List

columns of A with ∞ removed, self loops removed



edges m

Distributed Bellman-Ford