Spring 2026 Computer Science Project Fair

Featuring projects from CSCI 500, 701 and 702!

Your AI Memory for Every Meeting, Chat, and Idea – Memora AI
Mohammad Abbas, PB

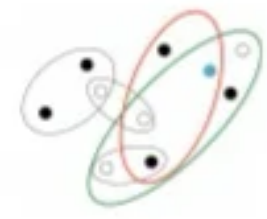
Pocket Survey: A Crowdsourced Ecological Survey
Kailey Mendoza, Grayson Wade

Automatic Schedule Optimization
Sebastian Cruz, Alex Talamba

Midd-Lineups: Revolutionizing College Basketball Analytics
Dante Aguilar, Marli Cohen

Tag, You're It: Robots Co-Evolve to Play Tag Through Competitive Reinforcement Learning
Toby Penner

Implementation and Evaluation of a WebSocket-to-gRPC Bidirectional Streaming Gateway
Esdras Ntuyenabo



Community Detection in Edge-Copying Hypergraphs
Frannie Cataldo

Friday May 8th, 11am - 1pm, 75 Shannon Street (outside Room 206)

Middlebury

DIJKSTRA'S ALGORITHM

Learning Goals

- Describe Dijkstra's alg
- Prove correctness of Dijkstra's alg.
- Analyze runtime of Dijkstras alg

3" x 5" notecard

Announcements

Will share slides with 701

Exit Tickets

- Who ensures ethics review in real world?
- Why does Dijkstra commit to a path + never update?

Shortest Path Problem

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

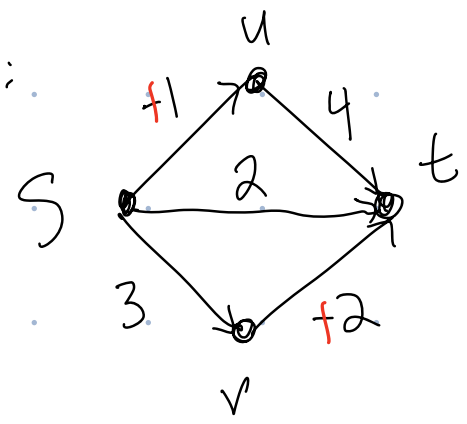
Output: Path P from s to t in G

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

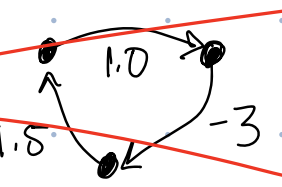
Example:

G:



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

Negative Cycle:



Not allowed

Various Approaches

- Brute Force → Breadth-First Search → edges all have weight 1
- Greedy → Dijkstra's → edges have positive wt.
- Dynamic → Bellman-Ford → edges can have negative wt
  - → can work with global or distributed G
  - → fails if must avoid neg. cycles

Dijkstra's Algorithm

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

Output: n-dimensional arrays L, P s.t.

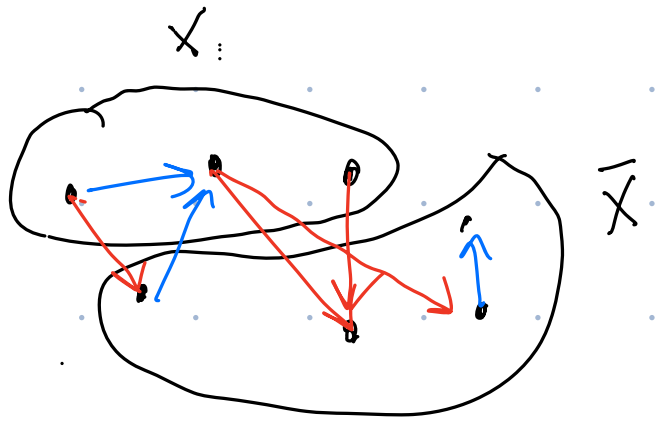
$L[v]$: length of shortest path from s to v in G

$P[v]$: shortest path from s to v in G

$X \leftarrow \{s\}$ // X stores visited vertices

$L[s] \leftarrow 0$

$P[s] \leftarrow \emptyset$



C is set of red edges

While there is an edge from X to $\bar{X}$:

$C \leftarrow \{(u,v): u \in X, v \in \bar{X}\}$

$(u^+, v^+) \leftarrow \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$ "Dijkstra's criterion"

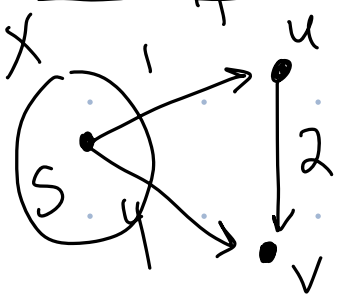
"(u^*, v^*) has minimal Dijkstra's criterion"

$X \leftarrow X \cup \{v^+\}$

$L[v^+] \leftarrow L[u^+] + w(u^+, v^+)$ addition

$P[v^+] \leftarrow P[u^+] + (u^+, v^+)$ append

Example

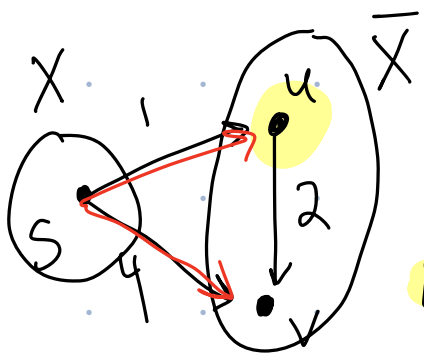


L

s	0

P

s	ϕ



$C = \{(s,u), (s,v)\}$

$L[s] + w(s,u)$

1

$L[s] + w(s,v)$

4

L

s	0
u	1

P

s	ϕ
u	(s,u)

$X = \{s\}$

$L[s] = 0$

$P[s] = \phi$

While there is an edge from X to $\bar{X}$:

$C \leftarrow \{(u,v) : u \in X, v \in \bar{X}\}$

$(u^+, v^+) \leftarrow \underset{(u,v) \in C}{\text{argmin}} \{L[u] + w(u,v)\}$

$L[v^+] \leftarrow L[u^+] + w(u^+, v^+)$

$P[v^+] \leftarrow P[u^+] + (u^+, v^+)$

$X \leftarrow X \cup \{v^+\}$

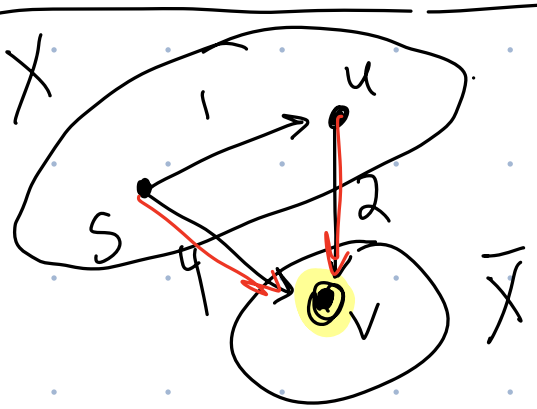
Show failure with negative weights

L

s	0
u	1
v	3

P

s	ϕ
u	(s,u)
v	(s,u)+(u,v)



$C = \{(s,v), (u,v)\}$

$L[s] + w(s,v)$

$0 + 4$

$L[u] + w(u,v)$

$1 + 2$

Dijkstra's Algorithm

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

Output: n-dimensional arrays L, P s.t.

$L[v]$: length of shortest path from s to v in G

$P[v]$: shorstest path from s to v in G

$X \leftarrow \{s\}$ // X is set of visited vertices

$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$

} Base case

While there is an edge from X to $\bar{X}$:

$C \leftarrow \{(u,v): u \in X, v \in \bar{X}\}$

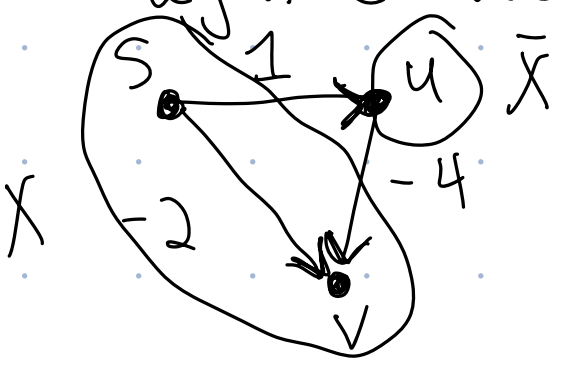
$(u^+, v^+) \leftarrow \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$

$L[v^+] \leftarrow L[u^+] + w(u^+, v^+)$

$P[v^+] \leftarrow P[u^+] + (u^+, v^+)$

$X \leftarrow X \cup \{v^+\}$

Show Dijkstra's alg fails with negative weights



Under what conditions can Dijkstra's alg have neg. weights but be successful?

Thm: Dijkstra's alg. Correctly returns the shortest path

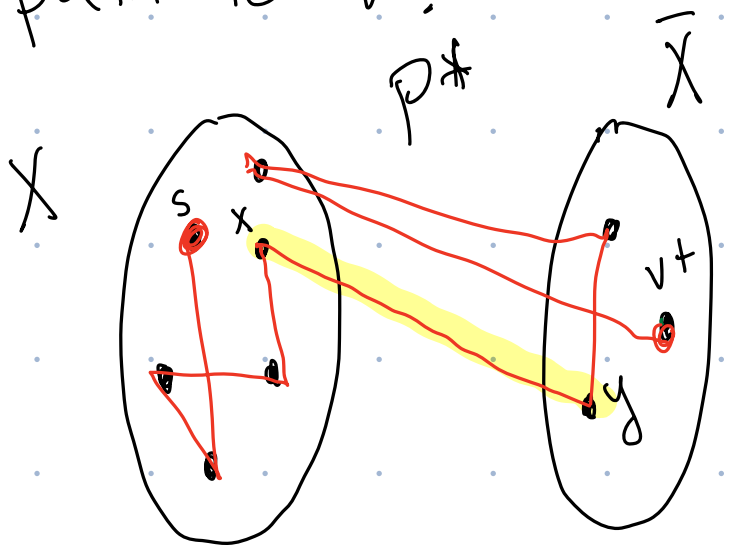
Pf: We will prove using induction on $n=|X|$ that Dijkstra's alg correctly assigns $L[v]$ and $P[v]$ $\forall\, v \in X$, for $n \geq 1$

When $n=1$, $X=\{s\}$, and code sets $L[s]=0$, $P[s]=\phi$, which is correct.

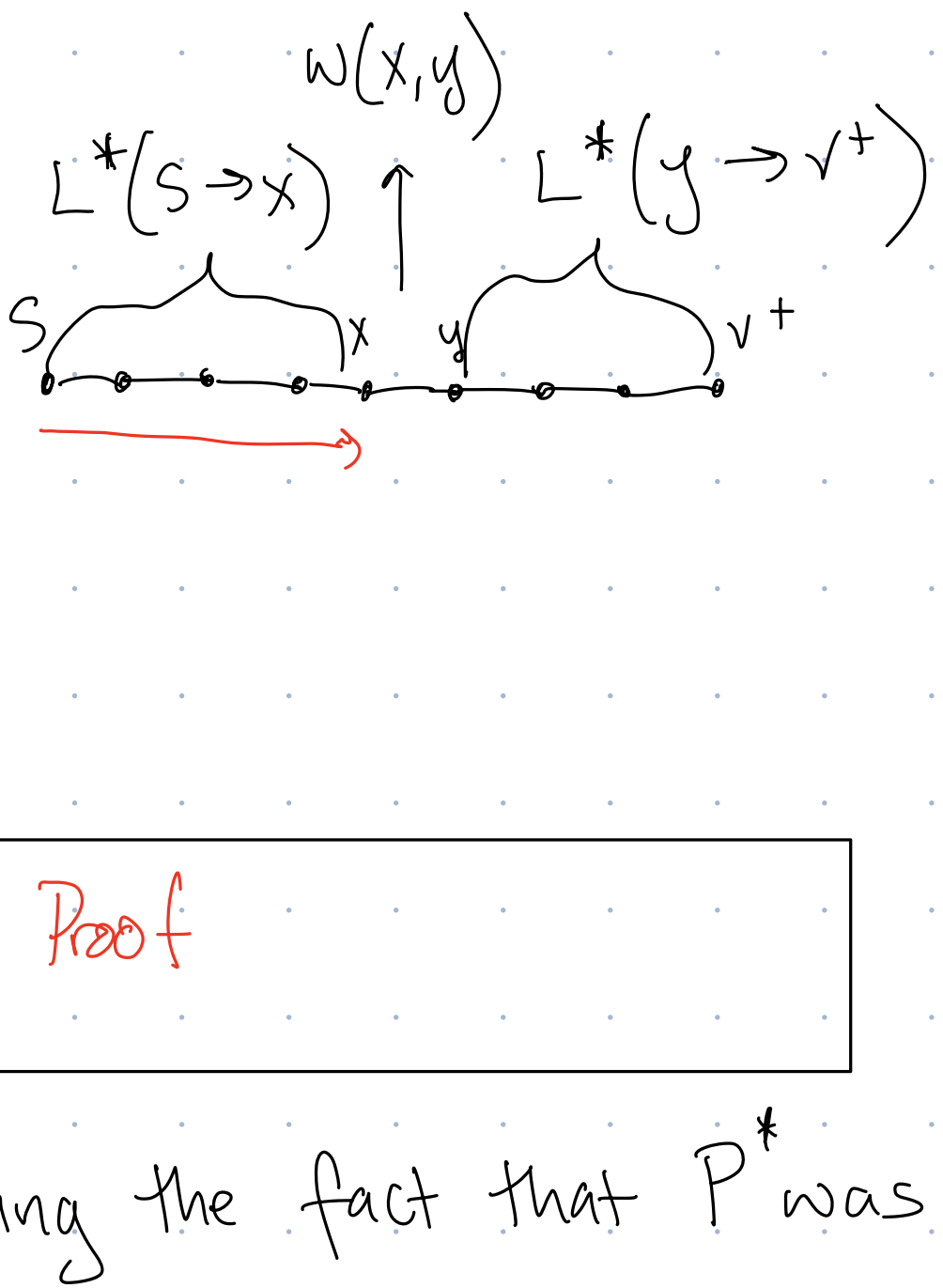
Let $k \geq 1$. Assume for induction that Dijkstra's alg. correctly assigns $L[v]$ and $P[v]$ $\forall\, v \in X$ when $|X|=k$

Suppose Dijkstra's alg. is about to add the $(k+1)^{th}$ element to X. Let $(u^+, v^+) = \underset{(u,v)\in C}{\operatorname{argmin}} \{L[u]+w(u,v)\}$, so Dijkstra's chooses v^+ to be $(k+1)^{th}$ element of X and sets $P[v^+]=P[u^+]+(u^+,v^+)$ and $L[v^+]=L[u^+]+w(u^+,v^+)$. We need to prove these assignments are correct.

Suppose for contradiction $P = P[v^+] = P[u^+] + (u^+, v^+)$ is not the shortest path. Let $P^* \neq P$ be the optimal path to v^+.



$(x, y) = 1^{st}$ edge in C to appears in P^*



Missing Part of Proof

Thus $L(P^*) \geq L(P)$, contradicting the fact that P^* was optimal and P was not.

$$L(P^*) = L^*(s \to x) + w(x, y) + L^*(y \to v^+)$$

$$\geq L^*(s \to x) + w(x, y)$$

$$\geq L[x] + w(x, y)$$

$$\geq L[u^+] + w(u^+, v^+)$$

$$= L(P)$$

$L(P^*) < L(P)$

← no edges have negative weight

← By inductive assumption, since x is a visited vertex $L[x]$ must be the length of shortest path from s to x.

Because (u^+, v^+) has minimal Dijkstra criteria

What is the runtime of Dijkstra's alg as written?
What data structure should you use to improve?
What is runtime with improved data structure?

$X \leftarrow \{s\}$

$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$

While there is an edge from X to $\bar{X}$:

$C \leftarrow \{(u,v) : u \in X, v \in \bar{X}\}$

$(u^+, v^+) \leftarrow \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$

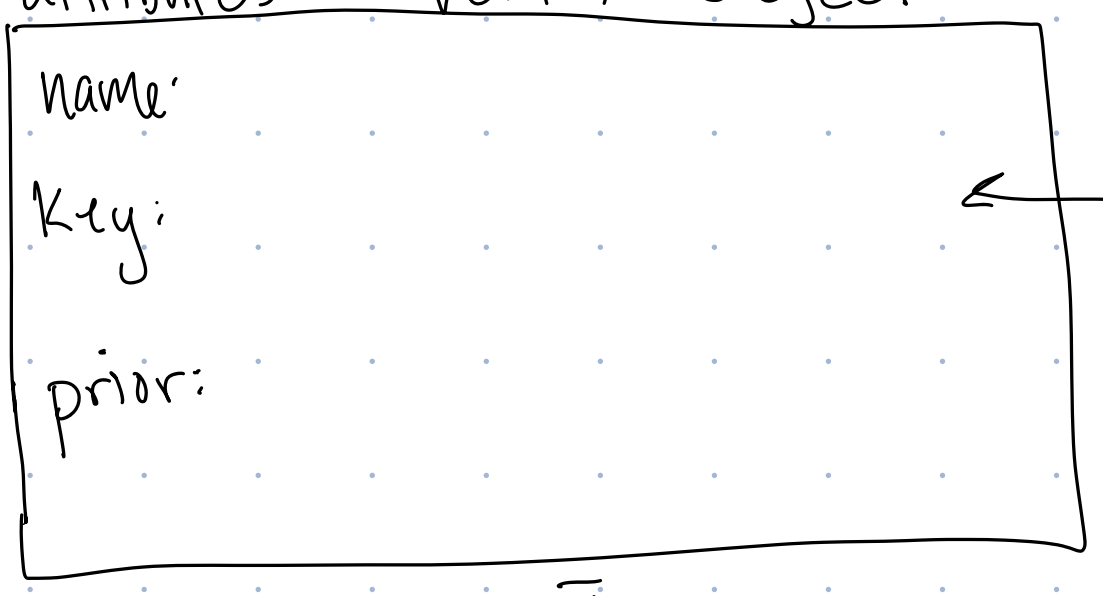
$L[v^+] \leftarrow L[u^+] + w(u^+, v^+)$

$P[v^+] \leftarrow P[u^+] + (u^+, v^+)$

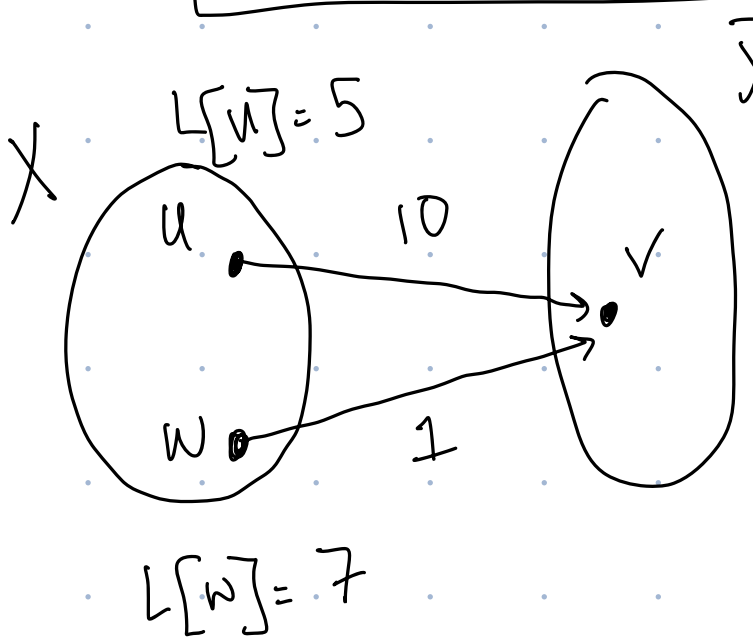
$X \leftarrow X \cup \{v^+\}$

Objects in Priority Queue: vertices $v \in \bar{X}$

attributes Vertex Object



Priority Queue ordered by key value



In this situation, what should v.prior be set to?

A) u B) w C) 15 D) 8

Dijkstra's

$X \leftarrow \{s\}$

$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$

// Initialize Heap

For $u : (s,u) \in E$:

|

For other $u \in \overline{X}$ not yet in heap:

|

Min Heap

- $O(n \log n)$ • Initialize n items in heap
- $O(\log n)$ • Remove ~~min~~ any item
- $O(\log n)$ • Insert new item

if have a pointer to item

While $H \neq \emptyset$:

$v^+ \leftarrow H.pop$ // v^+ automatically has minimum Dijkstra's criterion

$X \leftarrow X \cup \{v^+\}$

$L[v^+] \leftarrow v^+.key$

$P[v^+] \leftarrow P[v^+.prior] + (v^+.prior, v^+)$

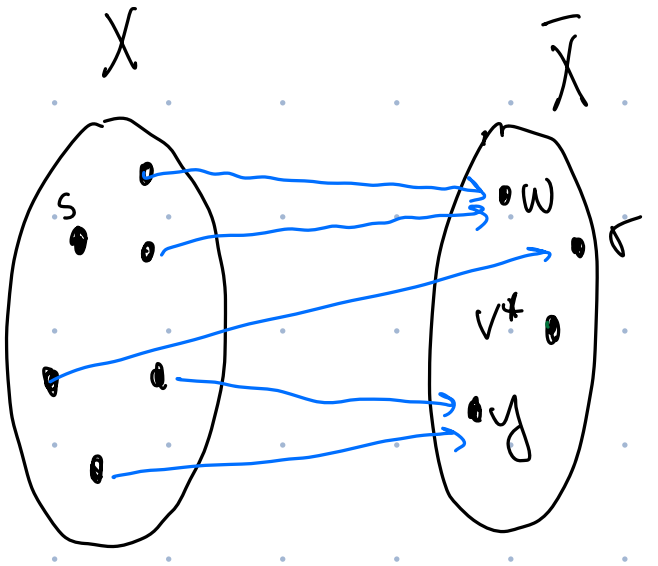
// Update Heap

attributes — Vertex Object

name: v

key: $\min_{u \in X} L[u] + w(u,v)$

prior: $\operatorname{argmin}_{u \in X} L[u] + w(u,v)$



Why is runtime $O((n+m)\log n))$ with adj. list?

Dijkstra's

$X \leftarrow \{s\}$

$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$

// Initialize Heap

For $u \in V - \{s\}$:

| If $(s,u) \in E$:

| | $u.key \leftarrow w(s,u)$

| | $u.prior \leftarrow s$

Else:

$u.key \leftarrow \infty$

$u.prior \leftarrow \phi$

Insert u into heap H

Min Heap

$O(n \log n)$	• Initialize n items in heap
$O(\log n)$	• Remove ~~min~~ any item
$O(\log n)$	• Insert new item

if have a pointer to item

While $H \neq \emptyset$:

$v^+ \leftarrow H.\text{pop}$

$X \leftarrow X \cup \{v^+\}$

$L[v^+] \leftarrow v^+.\text{key}$

$P[v^+] \leftarrow P[v^+.\text{prior}] + (v^+.\text{prior}, v^+)$

// Update Heap

For $r : (v^+, r) \in E$

Remove r from H

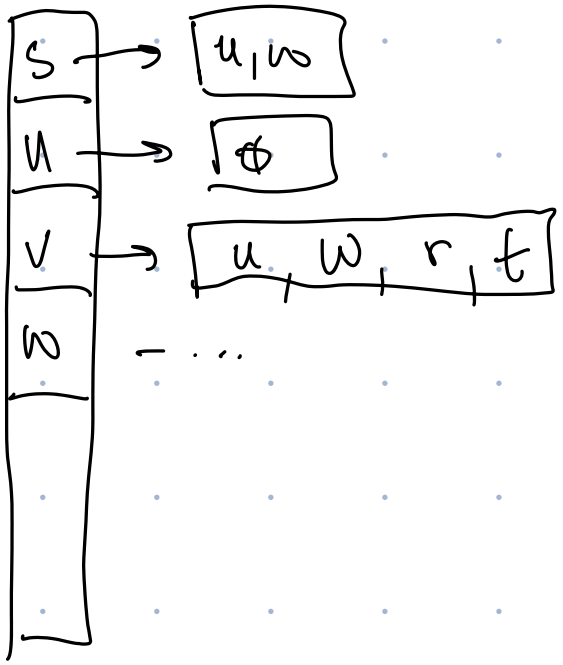
If $r.\text{key} > L[v^+] + w(v^+, r)$:

$r.\text{key} \leftarrow L[v^+] + w(v^+, r)$

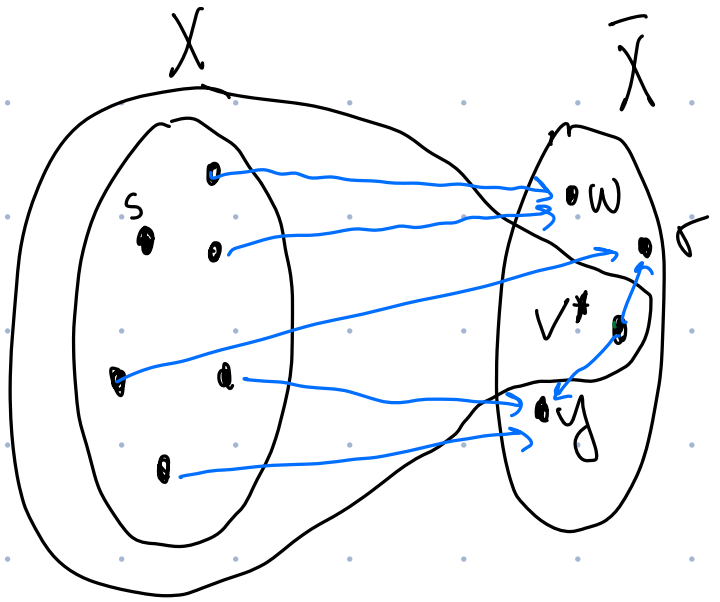
$r.\text{prior} \leftarrow v^+$

Reinsert r into H

Why is runtime $O((n+m)\log n))$ with adj. list?

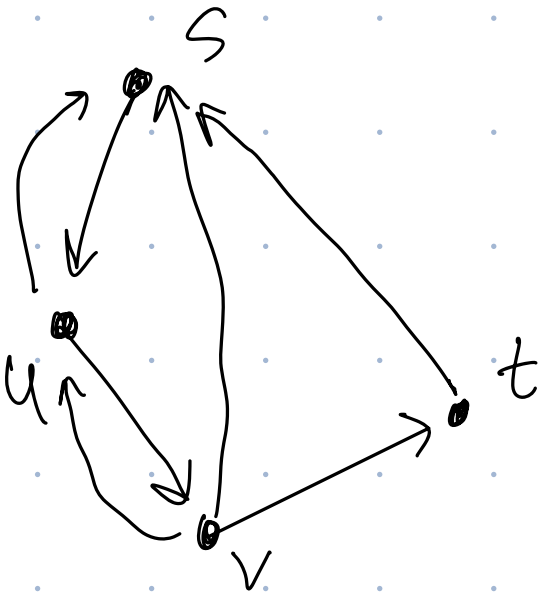


Each vertex is popped once. When popped, go thru its adj list. Never check that list again...

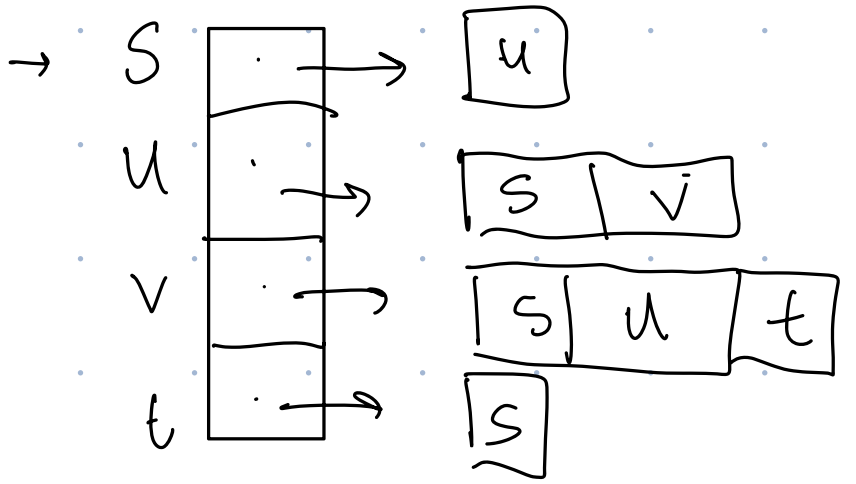


Compare to Bellman Ford: $O((n+m)n)$

For $u \in V$:
  Print u
  For $v \in (u,v) \in E$:
    Print v



7 edges



7 = sum of items in list