Learning Goals

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

Shortest Path Problem

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

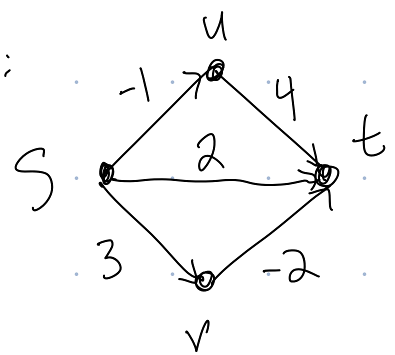
Output: Path P from s to t in G

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

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

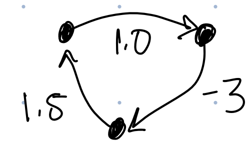
Example:

G:



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

Negative Cycle:



Not allowed

Various Approaches

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

Dijkstra's Algorithm

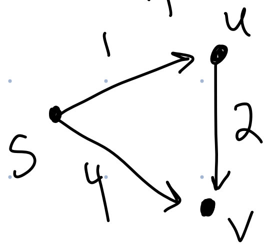
Input: $G = (V, E)$, $s \in V$, $|V| = n$, $w: E \to \mathbb{R}^+$

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

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

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

Example



	L	P
s		

$X = \{s\}$

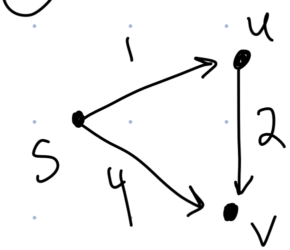
$L[s] = 0$

$P[s] = \emptyset$

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

- $C \leftarrow \{(u,v) : u \in X,\ v \in \bar{X}\}$
- $(u^*, v^*) \leftarrow \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$
- $L[v^*] \leftarrow L[u^*] + w(u^*, v^*)$
- $P[v^*] \leftarrow P[u^*] + (u^*, v^*)$
- $X \leftarrow X \cup \{v^*\}$

①



$X =$, $\bar{X} =$

$C = \{ \quad \}$

	L	P
s		
u		

$X = \{ \quad \}$

	L	P
s		
u		
v		

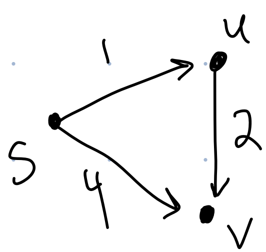
$C = \{ \quad \}$

$X = \{ \quad \}$

②

$X = \{ \quad \}$

$\bar{X} = \{ \quad \}$



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 $\bar{X}$:

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

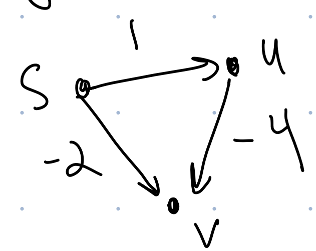
$(u^*, v^*) \leftarrow \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$

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

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

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

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 pal

Pf:

$$
\begin{aligned}
L(p^*) &= \\
&\geq \\
&\geq \\
&\geq \\
&=
\end{aligned}
$$

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 $\bar{X}$:

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

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

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

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

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

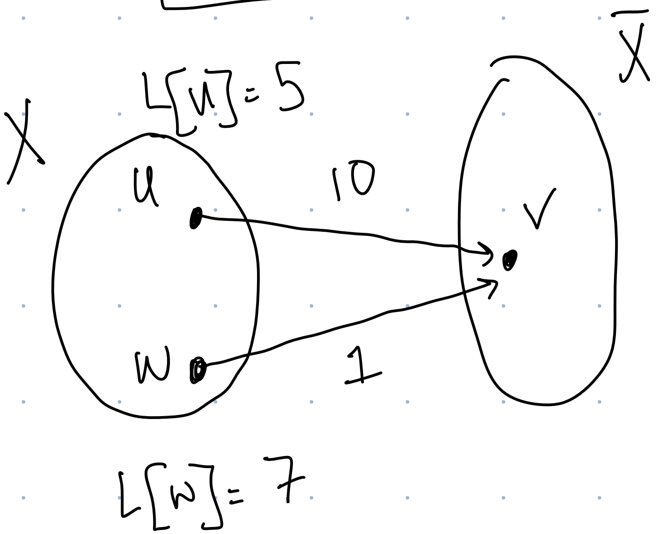
Objects in Priority Queue: vertices $v \in \bar{X}$

attributes   Vertex Object

name:

key:

prior:



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

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 already found item

While

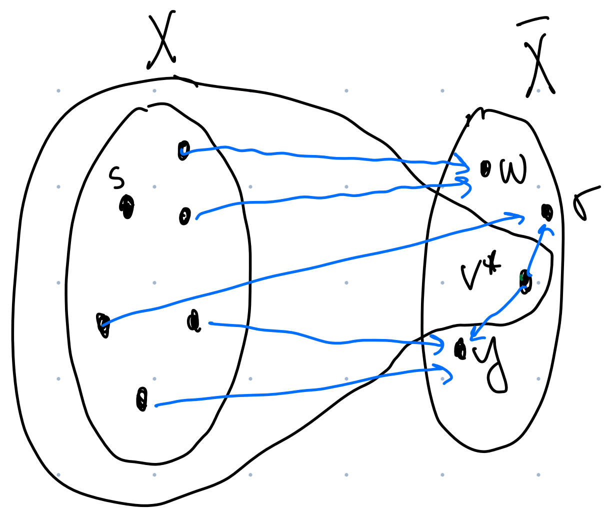
// Update Heap

attributes Vertex Object

name: $\boxed{v}$

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

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



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

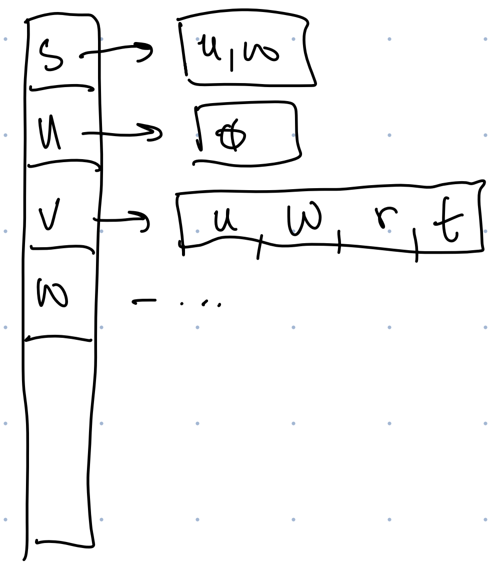
Min Heap

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

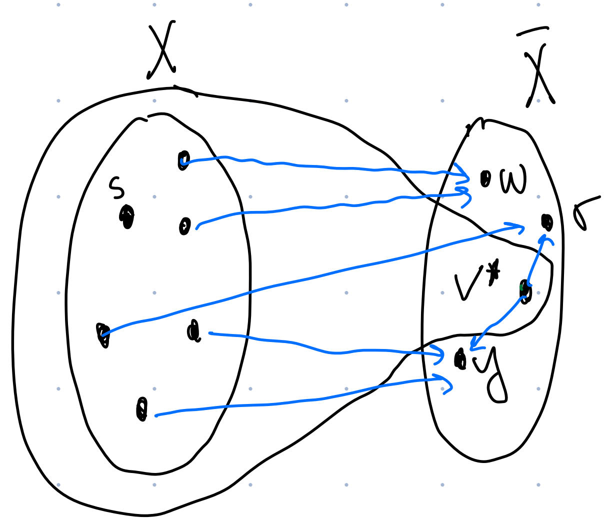
if have already found item

While $H \neq \emptyset$:

- $v^* \leftarrow H.pop$
- $X \leftarrow X \cup \{v^*\}$
- $L[v^*] \leftarrow v^*.key$
- $P[v^*] \leftarrow P[v^*.prior] + (v^*.prior, v^*)$
- // Update Heap
- For $r : (v^*, r) \in E$
  - Remove r from H
  - If $r.key > L[v^*] + w(v^*, r)$
    - $r.key \leftarrow L[v^*] + w(v^*, r)$
    - $r.prior \leftarrow v^*$
  - Reinsert r into H



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



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