Shortest Path Problem

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

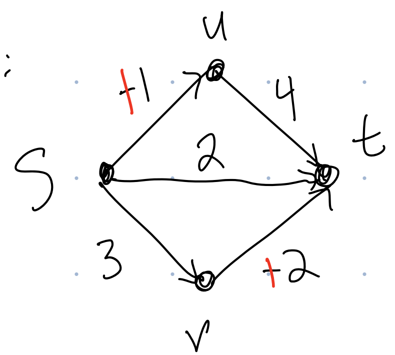
Output: Path P from s to t in G

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

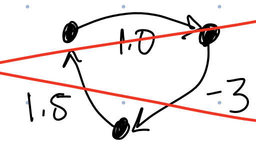
Example:

G:



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

~~Negative Cycle:~~



Not allowed

Various Approaches

- 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

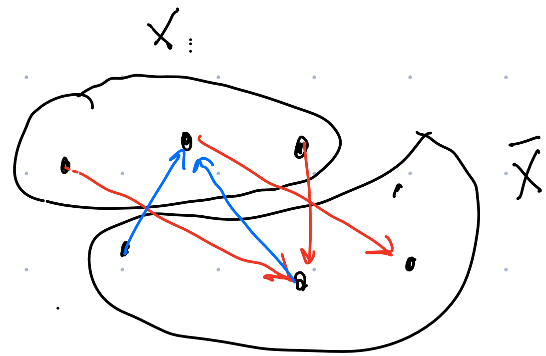
Input: $G = (V, E)$, $s \in V$, $|V| = n$, $w: E \to \mathbb{R}^+$

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

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



C is set of red edges

$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$ } Base case

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

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

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

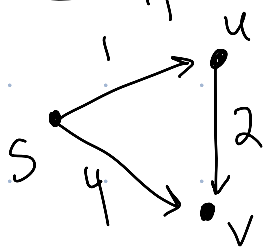
"(u^*, v^*) has minimal Dijkstra's criterion"

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

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

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

Example



	L	P
s	0	ϕ

$X = \{s\}$

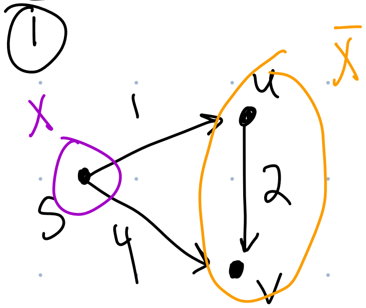
$L[s] = 0$

$P[s] = \phi$

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

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

①



$X = \{s\}, \bar{X} = \{u, v\}$

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

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

$0 + 1$

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

$0 + 4$

	L	P
s	0	ϕ
u	1	(s,u)

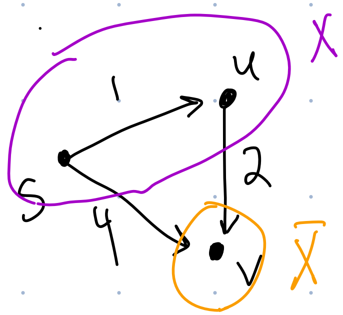
$X = \{s, u\}$

	L	P
s	0	ϕ
u	1	(s,u)
v	3	(s,u),(u,v)

②

$X = \{s, u\}$

$\bar{X} = \{v\}$



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

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

$0 + 4 = 4$

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

$1 + 2 = 3$

$X = \{s, u, v\}$

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

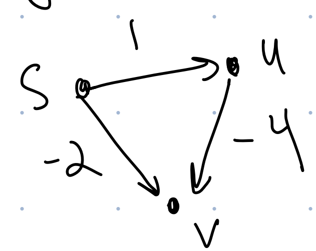
$L[s] \leftarrow 0$

$P[s] \leftarrow \phi$ } Base case

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

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

Show 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 $P[v]$ $\forall v \in X$.

When $n=1$, $X = \{s\}$ and $L[s]=0$, $P[s]=\emptyset$ are correct b/c you can get from s to itself with no edges in 0 length.

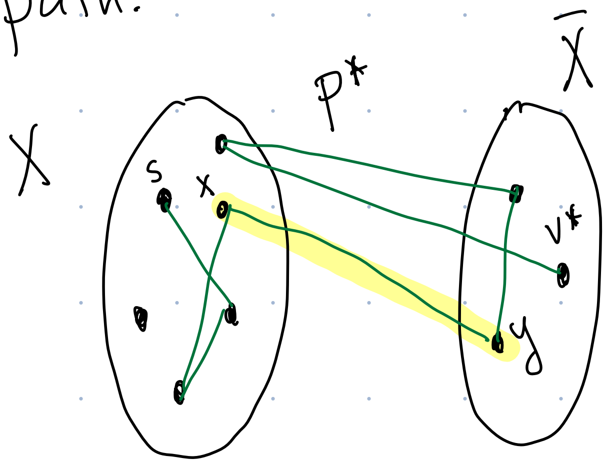
Let $k \geq 1$. Assume for induction that Dijkstra's alg. Correctly assigns $L[v]$ and $P[v]$ $\forall v \in X$ when $|X| = k$

Suppose Dijkstra's alg. is about to add the $(k+1)^{th}$ element to X.

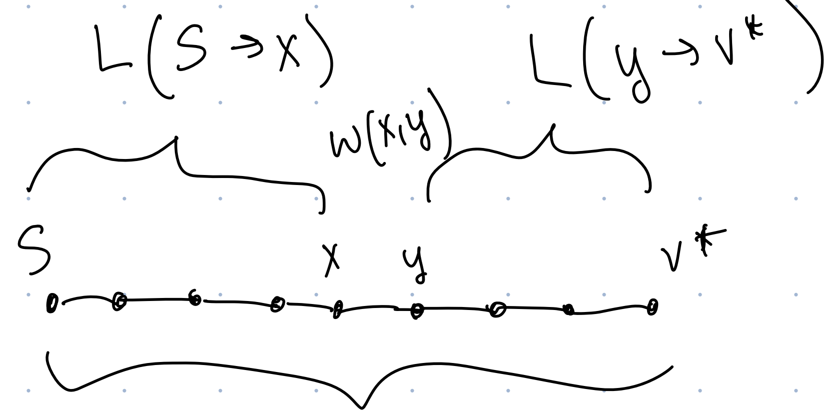
Let $(u^*, v^*) = \underset{(u,v) \in C}{\arg\min} \{ L[u] + w(u,v) \}$, so Dijkstra's chooses v^* to be $(k+1)^{th}$ element of X and sets $P[v^*] = P[u^*] + (u^*, v^*)$ and $L[v^*] = L[u^*] + w(u^*, v^*)$. We need to prove these assignments are correct.

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

$L(a \to b)$: length of P^*'s path from a to b



$\Rightarrow$



(x,y): 1^{st} edge in C to appear in P^*

P^* (stretched out)

So $$L(P^*) = \underbrace{L(s \to x) + w(x,y)}_{\geq\, ?} + \underbrace{L(y \to v^*)}_{\geq\, ?}$$

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

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

$$\geq L(s \to x) + w(x,y) \qquad (\text{b/c } L(y \to v^*) \geq 0)$$

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

b/c P^*'s path from s to x will be at least as long as shortes path from s to x. By induction assumption, $L[x]$ is length of shortest path from s to v.

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

since Dijkstra chose (u^*, v^*) to minimize Dijkstra's criterion

$$= L(P)$$

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 $\bar{X}$:

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

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

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

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

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

(faster!)

vertices in $\bar{X}$

Store ~~edges in C~~ in MinHeap!

(priority queue)

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

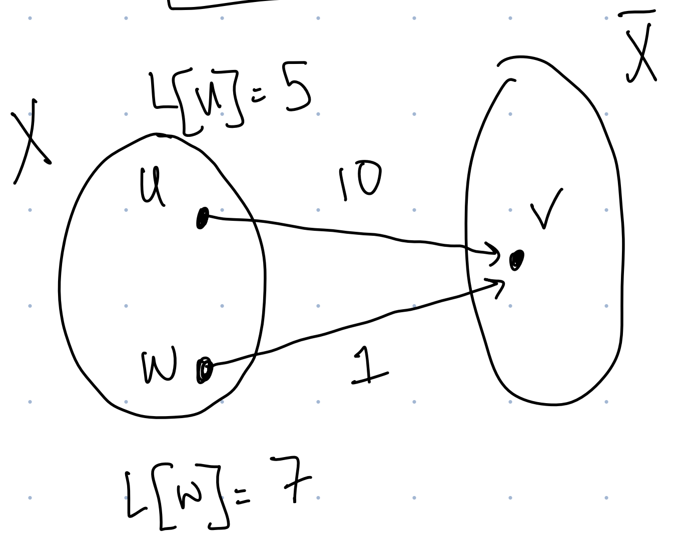
attributes Vertex Object

name: $\boxed{v}$

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

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

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

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

$u.prior \leftarrow s$

Insert u into heap H

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

$u.key \leftarrow \infty$

$u.prior \leftarrow \phi$

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 $H \neq \emptyset$:

$v^* \leftarrow H.pop$ // u 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

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

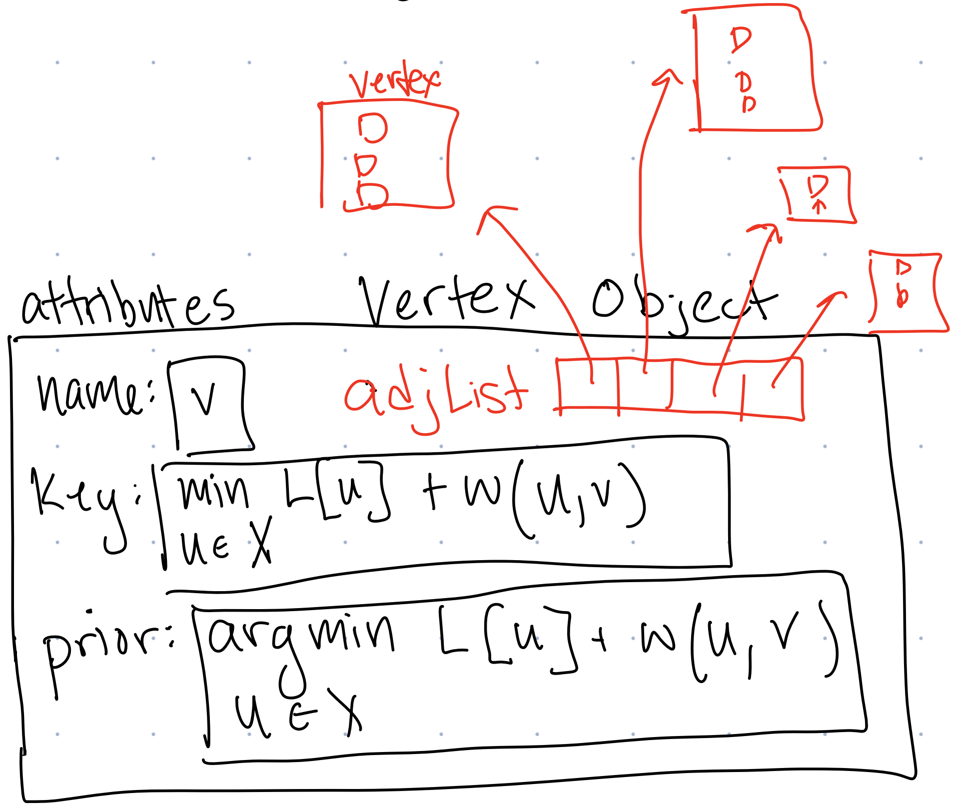
Remove r from H

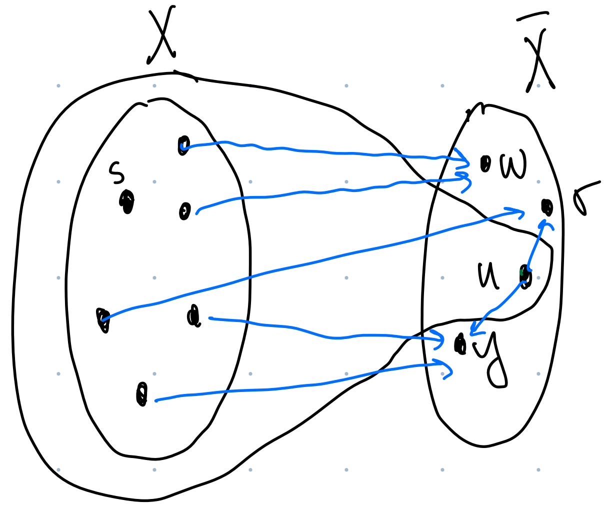
If $r.key > L[v^*] + w(v^*, r)$:

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

$r.prior \leftarrow v^*$

Reinsert r into H





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

Dijkstra's

$X \leftarrow \{s\}$
$L[s] \leftarrow 0$
$P[s] \leftarrow \phi$

$O(1)$

// 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

$O(n\log(n))$

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 $H \neq \emptyset$: $O(n)$ $O(n \log n)$

$\quad u \leftarrow H.pop$ $O(\log n)$

$\quad X \leftarrow X \cup \{u\}$

$\quad L[u] \leftarrow u.key$ $O(1)$

$\quad \cancel{P[u] \leftarrow P[u.prior] + (u.prior, u)}$

$\quad$ // Update Heap $O(m)$ $O(m \log n)$

$\quad$ For $v : (u,v) \in E$ $\leftarrow O(\#\text{ edges of } u)$

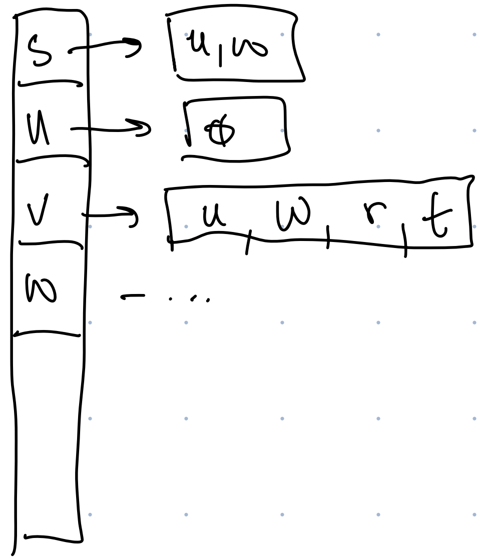
$\quad\quad$ Remove v from H $O(\log n)$

$\quad\quad$ If $v.key > L[u] + w(u,v)$: $O(1)$

$\quad\quad\quad v.key \leftarrow L[u] + w(u,v)$

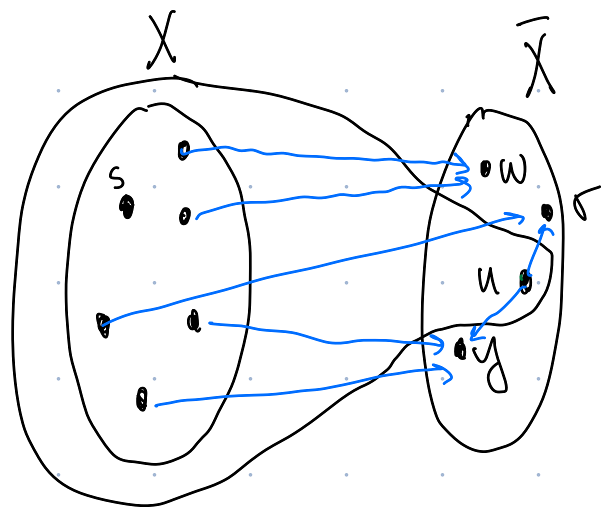
$\quad\quad\quad v.prior \leftarrow u$

$\quad\quad$ Reinsert v into H $O(\log n)$

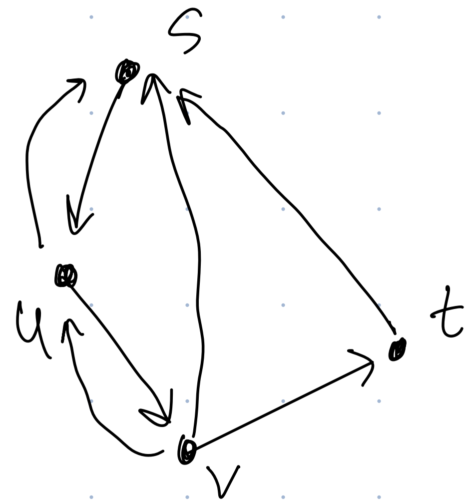


Why is runtime $O((n+m)\log n))$ with adj. list?

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

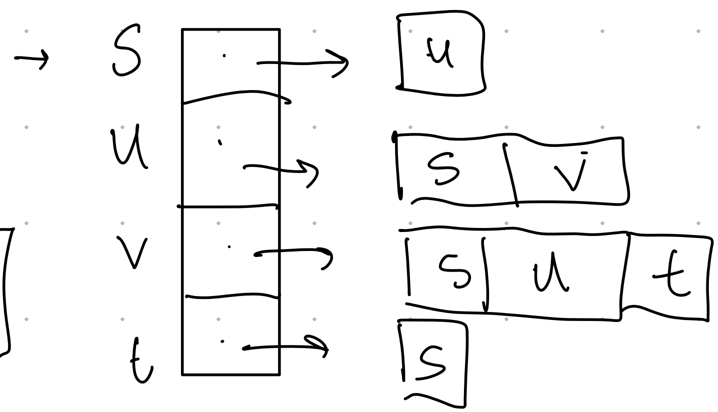


Compare to Bellman Ford: $O((n+m)n)$



7 edges

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



7 = sum of items in list

$\boxed{*\ u\ *\ sv\ *\ sut\ *\ s}$

$n+m$