Learning Goals

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

Announcements

- OH: Mon: 11-12, 2:30-3:30, Tues, Wed: 11-12
- Sunday: Final Drop-In evening hours
- Advice to future students go/cs302groups

Exit Tickets

- Negative weights

Shortest Path Problem

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

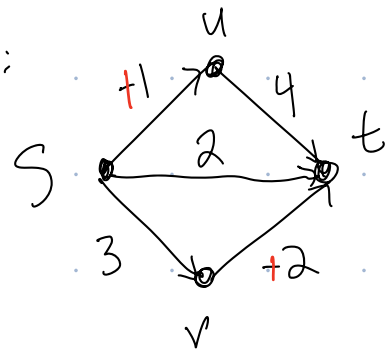
Output: Path P from s to t in G

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: ~~1.0, 1.5, -3~~ 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

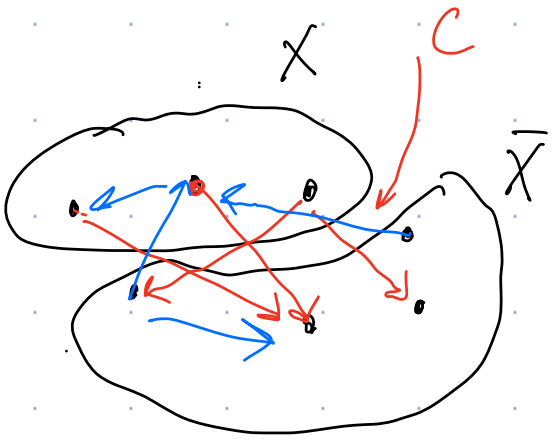
Input: $G = (V, E)$, $s \in V$, $|V| = n$, $w : E \to \mathbb{R}^+$

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



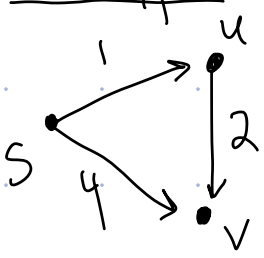
$L[s] \leftarrow 0$ } Base cases

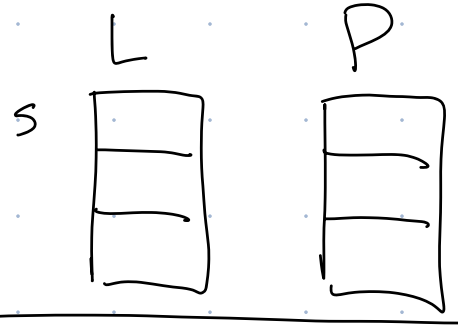
$P[s] \leftarrow \emptyset$ }

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

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

Example





$X = \{s\}$

$L[s] = 0$

$P[s] = \phi$

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

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

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

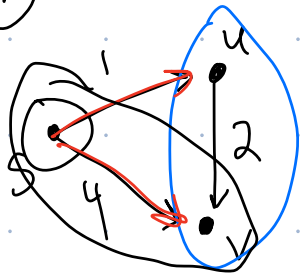
$L[v^*] \leftarrow L[u^*] + w(u^*, v^*)$

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

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

---

(1)



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

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

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

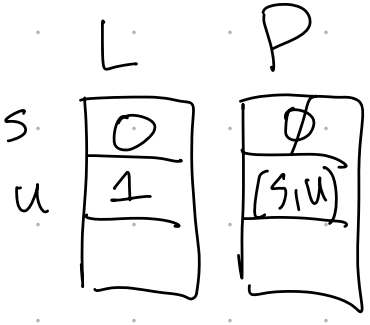
$0 + 1$

$u^* = s$

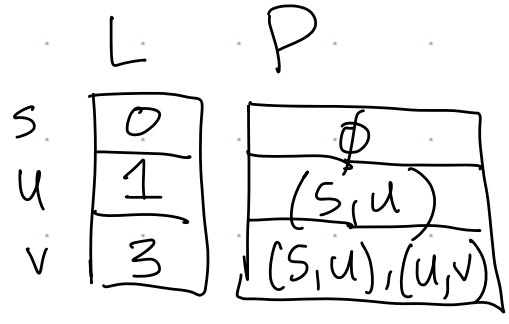
$L[s] + w(s,v)$

$0 + 4$

$v^* = u$



$X = \{s, v\}$

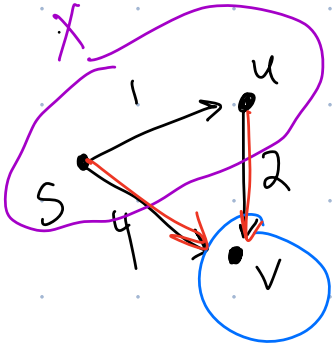


---

(2)

$X = \{s, u\}$

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



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

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

$0 + 4$

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

$1 + 2$

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

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

$L[s] \leftarrow 0$

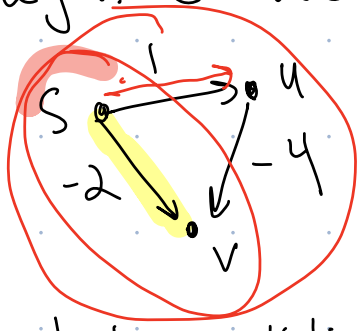
$P[s] \leftarrow \phi$

} Base case

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

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

Show Dijkstra's alg fails with negative weights



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

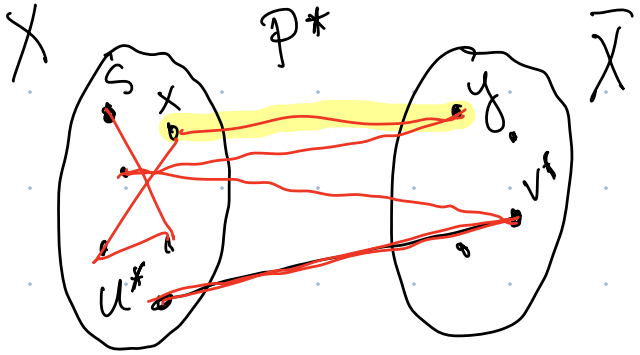
Thm: Dijkstra's alg. Correctly returns the shortest path

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

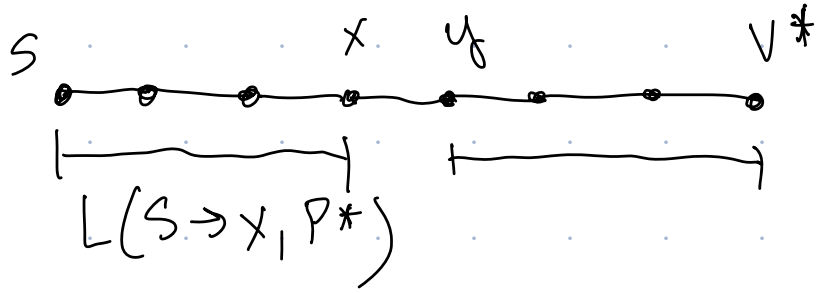
Base Case: When $n = 1$, $X = \{s\}$ and $L[s] = 0$, $P[s] = \emptyset$. This is correct b/c the shortest path from s to itself has length 0 and uses no edges.

Inductive Step: 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 alg. is about to add the $(k+1)^{th}$ vertex to X. Let $(u^*, v^*) = \underset{(u,v) \in C}{\operatorname{argmin}} \{L[u] + w(u,v)\}$, so Dijkstra chooses v^* to be the $(k+1)^{th}$ vertex in X, and it 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 (shortest) path.



$L(s \to x, P^*)$ is the length of the segment of P^* from s to x

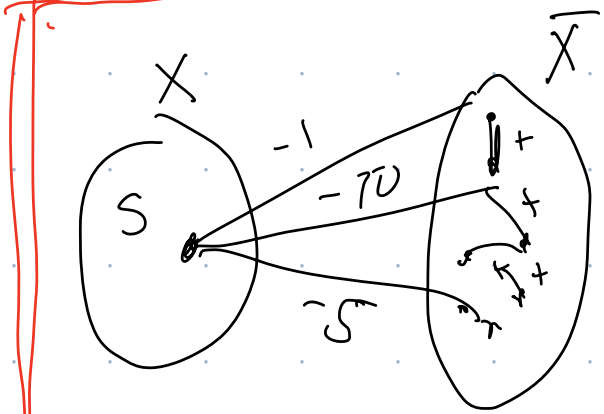


(x,y) is the 1st edge in C to appear in P^*

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

$$L(P) = L(s \to u^*, P) + w(u^*, v^*) \quad \ldots$$

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



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

(u^* and v^* mark x and y)

Because $L(y \to v^*) \geq 0$

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

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

Because P^*'s path from s to x will be at least as long as the shortest path from s to x. By inductive assumption, $L[x]$ is length of shortest path from s to x.

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

Because Dijkstra chose u^*, v^* to minimize Dijkstra's criterion

$$= L(P) = L[v^*]$$

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?

$X \leftarrow \{s\}$
$L[s] \leftarrow 0$
$P[s] \leftarrow \phi$
} $O(1)$

While there is an edge from X to $\bar{X}$: $O(n)$

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

$(u^*, v^*) \leftarrow \underset{(u,v) \in C}{\text{argmin}} \{L[u] + w(u,v)\}$ $\leftarrow O(n^2)$

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

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

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

Min Heap to store vertices

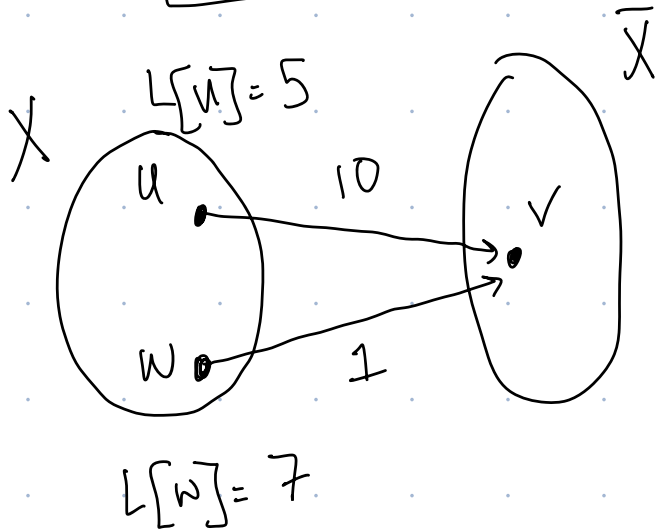
$|C| \leq O(n^2)$

$O(n^3)$ or $O(nm)$

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

attributes Vertex Object

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



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

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

1) 2) 3) 4)

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 object into heap H

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

| $u.key \leftarrow \infty$
| $u.prior \leftarrow \phi$
| 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 $H \neq \emptyset$:

$v^* \leftarrow H.pop$

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

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

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

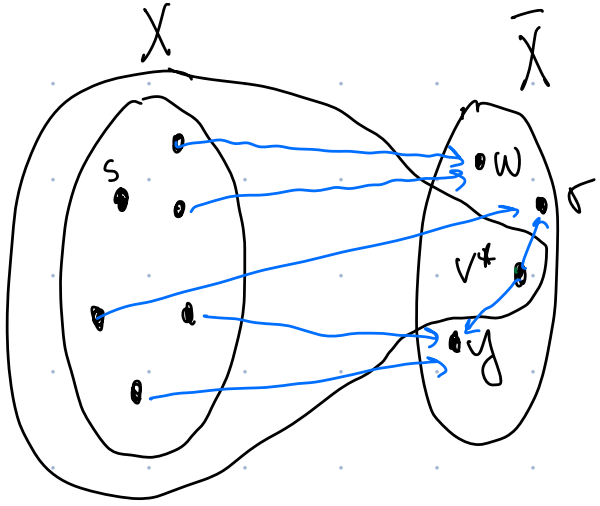
// Update Heap

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

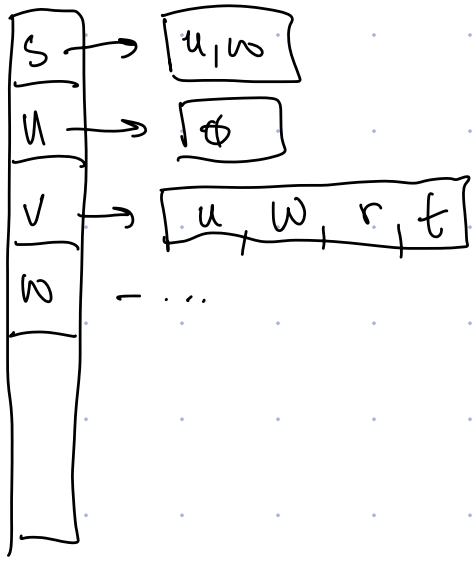
Min Heap

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

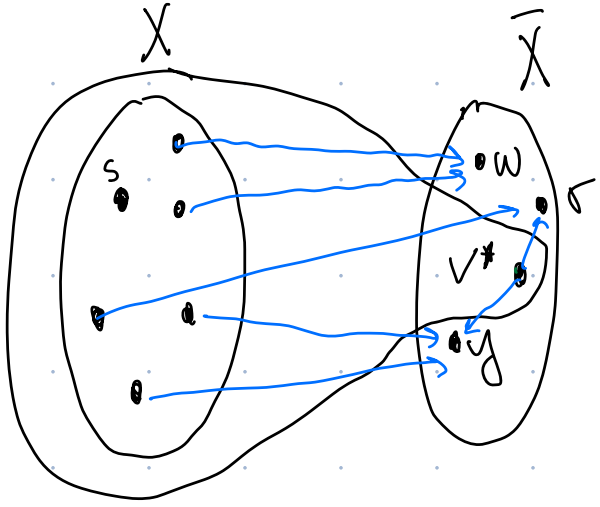
if have already found item

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?



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