# 13s: Dijkstra's Algorithm

Saturday, May 8, 2021
12:15 PM

## Shortest Path Problem

Input: • $G=(V,E)$, $|V|=n$, $|E|=m$
• $s,t \in V$

Output: Shortest path from $s$ to $t$

* with appropriate adjacency list

| Conditions | Algorithm | Runtime* |
|---|---|---|
| negative edge weights | Bellman-Ford | $O(nm)$ |
| no negative edges weights | Dijkstra's | $O((n+m)\log n)$ |
| all edges have weight 1 | Breadth First Search | $O(n+m)$ |

**Goals:**
- Describe Dijkstra's Algorithm
- Analyze runtime of Dijkstra's Algorithm

**Announcements:**
- Antiprocrastination Station go/ace
- Final Reflection
- Final Assessment
- Programming Assignment

**Questions**
- Application of BFS? LinkedIn, Kevin Bacon

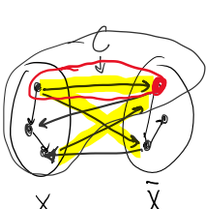
## Dijkstra's Algorithm:

$X = \{s\}$ // set of visited vertices

$\forall v$: $A[v] = \infty$ // length of shortest path from $s$ to $v$

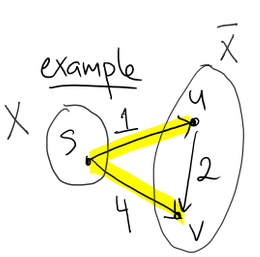
$A[s] = 0$

While there are edges from $X$ to $\bar{X}$:
- $C = \{(u,v) \in E : v \in X, v \in \bar{X}\}$
- Let $f(u,v) = A[u] + w(u,v)$ ← weight/length of edge $(u,v)$
- Pick $(u^*, v^*)$ in $C$ with smallest f-value.
- $A[v^*] = A[u^*] + w(u^*, v^*)$
- $X = X \cup \{v^*\}$



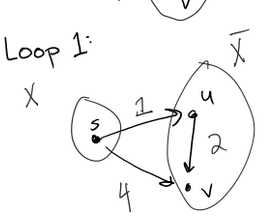

Work backwards through $A$ to find shortest path.

example




$X = \{s\}$ $\quad$ $A$: $s$: $0$, $u$: $\infty$, $v$: $\infty$

Loop 1:

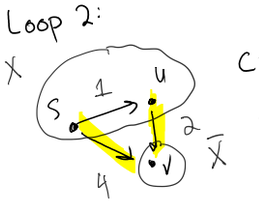



$C = \{(s,u), (s,v)\}$ $\quad$ $A[u] + w(u,v)$

$A[s] + w(s,u)$: $0 + 1 = 1$ $\quad$ $A[s] + w(s,v)$: $0 + 4 = 4$

$X = \{s,u\}$ $\quad$ $A$: $s$: $0$, $u$: $1$, $v$: $\infty$

Loop 2:



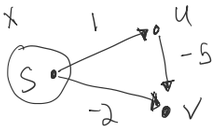

$C = \{(s,v), (u,v)\}$

$A[s] + w(s,v)$: $0 + 4 = 4$ $\quad$ $A[u] + w(u,v)$: $1 + 2 = 3$

$X = \{s,u,v\}$ $\quad$ $A$: $s$: $0$, $u$: $1$, $v$: $3$

## Group Work:

Come up with a counterexample that Dijkstra's algorithm does not always work when there are negative weight edges. (Once vertex is added to X, algorithm thinks it has found shortest path...)




$A[v] = -2$

## Proof of Correctness of Dijkstra

Exchange / Contradiction

## Analyzing Runtime

What do we repeatedly do ... what data structure would help us?

Priority Queue / MinHeap

What do we use...

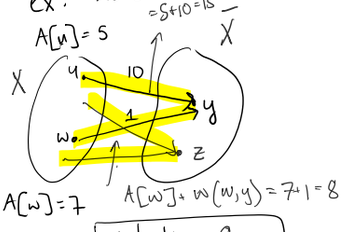
- Edges → Natural
- Vertices → Faster

Objects in Heap: $y \in \bar{X}$

$y$ | key: $\min_{\substack{u:\, u \in X \\ (u,y) \in E}} A[u] + w(u,y)$
prior: $u^*$ that minimizes key

attributes: key
prior

ex: $A[u] + w(u,y) = 5 + 10 = 15$




$y$ | · key = 8
· prior = w

## Dijkstra's Algorithm:

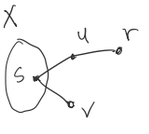
$X = \{s\}$ // X is set of visited vertices

$A[s] = (0, \phi)$ // A[u] = shortest distance $s \to u$

shortest path length  prior

// initialize heap

for veritices $u \neq s$:
  if $(s,u) \in E$
    $u.key = w(s,u)$
    $u.prior = s$
  else
    $u.key = \infty$
    $u.prior = \phi$
  insert u into our heap H



while $H$ is not empty

- $v^* = H.pop$
- $A[v^*] = (v^*.key, v^*.prior)$
- $X = X \cup \{v^*\}$
- Update heap

Use priors to work backwards quickly