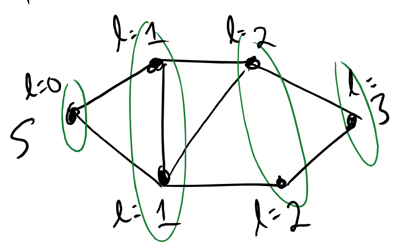
S.KIMMEL

Shortest Paths

Input: Graph $G=(V,E)$, $s \in V$

Output: $\forall\ v \in V$, $\ell(v)$ = shortest path from s to v

$\ell(v) = \infty$ if s, v not connected



Applications: Bacon #
Linked in Degree

Idea, explore layers.

$\ell[v] = \infty \quad \forall\ v \in V$ // will store shortest paths

$Ex[v] = $ False $\forall\ v \in V$ // mark True when "explored"

$A = \{\}$; ← What type of data structure is A?

A . add(s)

★ QUEUE: FIFO ★

$\ell[s] = 0$

or STACK: FILO

$Ex[s]$ = True

Without red: BFS,
With red: BFS for shortest path

```
while(A is not empty) {
    v = A.pop

    For each edge (v,w)        {
        If( Ex[w] = false )   A.add(w) ; Ex[v] = true ; l[w] = l[v] + 1 ;
    }
}
```

This is Breadth First Search - slowly move away in layers from initial node

S. KIMMEL

Q: Under what circumstances does BFS compute shortest paths?

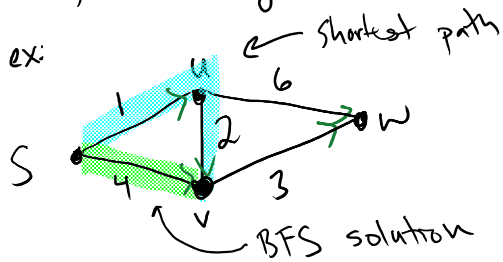
A) Only if graph is undirected

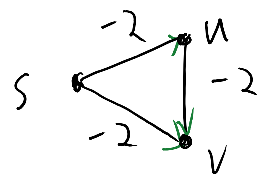
B) Only if graph has no cycles

C) Only if all edges have the same positive weight

D) Only if all edges have the same weight

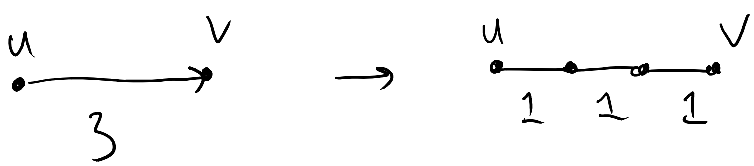
ex:





If all edges have the same weight $w>0$, shortest path is # of steps * w.

Q: Suppose graph edge weights are integers. How can you use BFS to solve shortest paths? When is this efficient?



Now all edges have same positive weight! Use BFS

Then runtime $= O\left(\sum_{i \in E} w_i + n\right)$

if large, can get slow

S.KIMMEL

<u>Reduction</u>: Given G, want to solve problem $P(G)$

$\downarrow$ transform instance

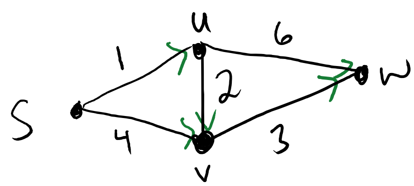
G' & solve problem $Q(G')$

$\downarrow$ transform solution

$P(G)$

S.KIMMEL

Shortest Paths



Input: • Directed graph $G = (E, V)$ with positive weights l_e $\forall\ e \in E$

• vertex $s \in V$

Output: $\forall\ v \in V$, find

$L(v)$ = length of shortest path from s to v

- path following direction of edges
- add weights of edges on path

Q: What is $L(s), L(u), L(v), L(w)$ for graph above?

A) 0, 1, 4, 6 B) 0, 1, 3, 7 C) 0, 1, 1, 2

$\boxed{\text{D) } 0, 1, 3, 6}$

Applications: navigation
financial transactions

Assumptions:
- s is connected to all other vertices (not important, just makes life easier)
- No negative edge weights (Important!)

S. KIMMEL

Dijkstra's Algorithm : Intuition, BFS

Initialization:

$X = \{s\}$ (vertices processed)

$A[s] = 0$ (array storing shortest path distance to v)

$B[s] =$ empty path (array storing shortest path to v)

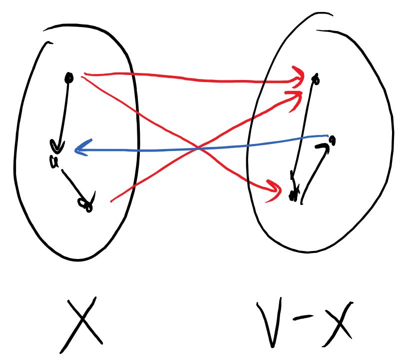
B not necessary for implementation, just helpful for understanding

while $X \neq V$:

- among edges $(v,w) \in E$ with $v \in X$, $w \in V-X$, pick edge that minimizes

$$\boxed{A[v] + \ell_{vw}}$$

Dijkstra's greedy criterion



We care about arcs from X to V-X
tail ↑ ↑ head

Let (v^*, w^*) be minimizing edge

- $X = X + w^*$
- $A[w^*] = A[v^*] + \ell_{v^*w^*}$

already computed, since $v^* \in X$

- $B[w^*] = B[v^*] + (v^*, w^*)$

S.KIMMEL

Q: What kind of algorithm is Dijkstra's Algorithm?

A) Divide & Conquer B) Dynamic Programming $\boxed{\text{C) Greedy}}$

D) Local Search

- Look at all edges from X to V-X, not local

- No smaller subsystems
- Choose best thing right now
- Easy to describe
- Hard to prove

S.KIMMEL

Proof of Correctness of Dijkstra

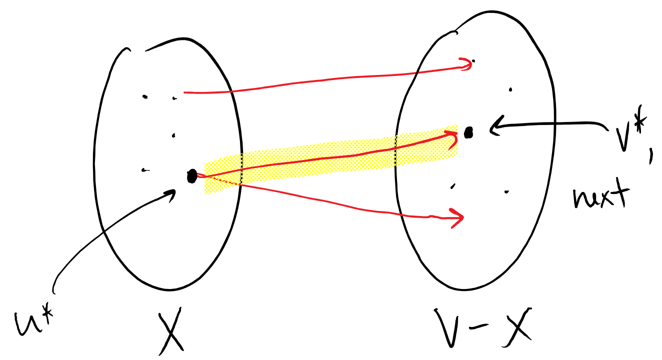
Loop Invariant: $\forall v \in X$, $A[v] = L[v]$, and $B[v]$ is shortest path from s to v

Initialization

$X = \{s\}$, $A[s] = 0$, $B[s] = \phi$

The shortest path from s to s has weight 0, and is empty ✓

Maintenance



Assume: $\forall$ $v \in X$

- $A[v] = L(v)$
- $B[v]$ is shortest path

Let v^* be the next vertex that Dijkstra's alg. adds:

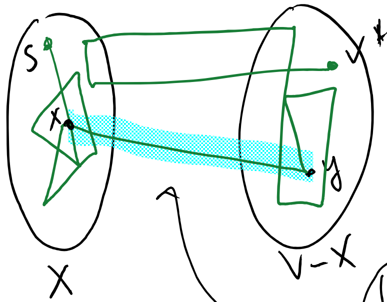
$$P = B[u^*] \cup (u^*, v^*) \quad , \quad A[v^*] = A[u^*] + \ell_{u^* v^*}$$

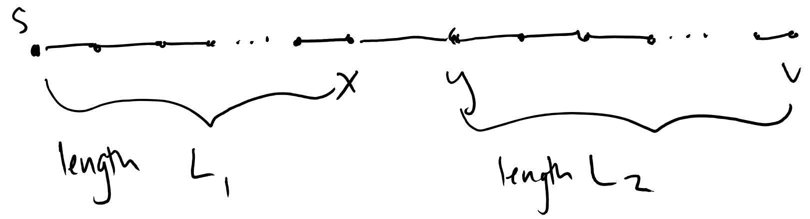
↑ actual shortest path path from s to u^*

Need to show:
- P is shortest path from s to v^*
- Since length of P is $A[u^*] + \ell_{u^* v^*}$, This implies $L[v^*] = A[v^*]$

S.KIMMEL

Suppose for contradiction, there is a shorter path from s to v^* (shorter than p)





(x,y) first crossing from X to $V-X$.

Q: Prove contradiction

A: $L_1 \geq A[x]$ by inductive assumption

$L_2 \geq 0$ by non-negativity of edges

Path length is

$$L_1 + L_2 + l_{xy} \geq A[x] + l_{xy} \geq A[u^*] + l_{u^*,v^*} = \text{length of } p$$

Contradiction!

Termination – obvious

S. KIMMEL

Runtime of Dijkstra's Algorithm

$X = \{s\}$

$A[s] = 0$

$B[s] = \emptyset$

while $X \neq V$

- among edges $(v, w) \in E$ with $v \in X$, $w \in V - X$, pick edge that minimizes

$$\boxed{A[v] + \ell_{vw}}$$

Dijkstra's greedy criterion

Let (v^*, w^*) be minimizing edge

- $X = X + w^*$
- $A[w^*] = A[v^*] + \ell_{v^* w^*}$ already computed, since $v^* \in X$
- $B[w^*] = B[v^*] + (v^*, w^*)$

Q: What is run time using adjacency list graph input?

A: $O(n + m)$ B: $\boxed{O(nm)}$ C: $O(n^2)$

D: $O(nm \log m)$

- Loop runs n times
- Each loop, go through all edges, if from $X \to V - X$, calculate criterion, find min: $O(m)$ time