Shortest Paths

Consider digraph $G=(V,E)$ with edge weight $w(e)$ associated with each edge e $(w: E \to \mathbb{R})$.

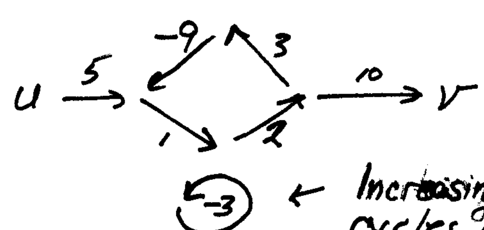
The weight of some path $p = v_1 \to v_2 \to \cdots \to v_k$ is $w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$.

E.g., $v_1 \xrightarrow{4} v_2 \xrightarrow{-2} v_3 \xrightarrow{-5} v_4 \xrightarrow{1} v_5$ $\qquad w(p) = -2$

Shortest path from u to v is a path of minimum weight from u to v. The shortest-path weight is the weight of such a path: $\delta(u,v) = \min\{w(p) : p \text{ is a path from } u \text{ to } v\}$.

Also, $\delta(u,v) = +\infty$ if no path from u to v exists

One subtlety:



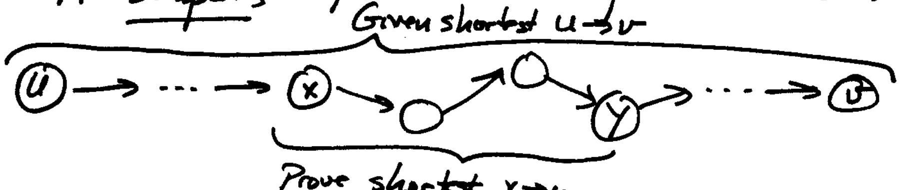
← Increasing # of negative cycles produces ~~increasing negative~~ decreasing weight path

$\delta(u,v) = -\infty$

Optimal substructure:

Theorem: A subpath of a shortest path is also a shortest path.

Proof:

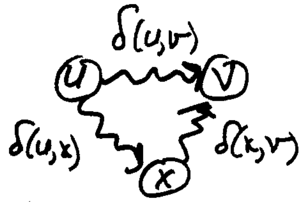


By cut-and-paste, if a shorter $x \to y$ path existed, we could insert it into the $u \to v$ path and produce a shorter $u \to v$ path, contradicting the given that $u \to v$ was a shortest path.

Triangle Inequality

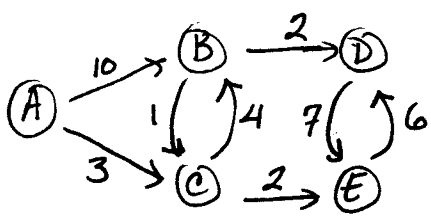
Theorem: For all $u, v, x \in V$, $\delta(u,v) \le \delta(u,x) + \delta(x,v)$

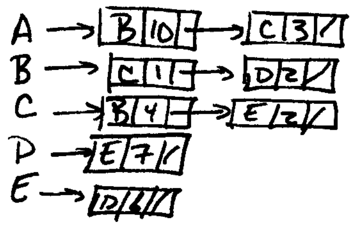
Proof:



If triangle inequality violated, then $u \rightsquigarrow x \rightsquigarrow v$ is a shorter path than $u \rightsquigarrow v$, contradicting statement that $\delta(u,v)$ corresponds to a shortest path.

Adjacency-List Representation





Size = $|E|$ for digraph ($2|E|$ for undirected graph)

Min-Priority Queue

A data structure for maintaining a set S of elements, each with an associated value (key), supporting:

- Insert (S, x) inserts the element x into S.
- Minimum (S) returns element with smallest key.
- Extract-Min (S) returns and removes element with smallest key
- Decrease-Key (S, x, k) decreases the value of element x's key to k

Single-source shortest paths problem

Goal: From a given source vertex $s \in V$, find the shortest-path weights $\delta(s,v)$ for all $v \in V$

Here we assume $w(u,v) \geq 0$, so $\delta(s,v) \geq 0 > -\infty$

Dijkstra's Algorithm (only valid for non-negative weights)

Idea: Greedy Algorithm

① Maintain set S of vertices whose shortest-path distances from s are known.

② At each step, add to S the vertex $u \in V - S$ whose distance estimate from s is minimum

③ Update distance estimates of vertices adjacent to u.

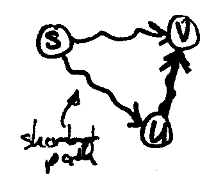
Dijkstra (G, w, s)

Maintain set of distance estimates, d and update those to shortest-path weights before adding to S

$d[s] \leftarrow 0$
$d[v] \leftarrow \infty$ **for** each $v \in V - \{s\}$
$S \leftarrow \emptyset$
$Q \leftarrow V$ (priority queue of vertices keyed by d)
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN (Q)
  $S \leftarrow S \cup \{u\}$
  for each $v \in Adj[u]$
    do if $d[v] > d[u] + w(u,v)$
      then $d[v] \leftarrow d[u] + w(u,v)$

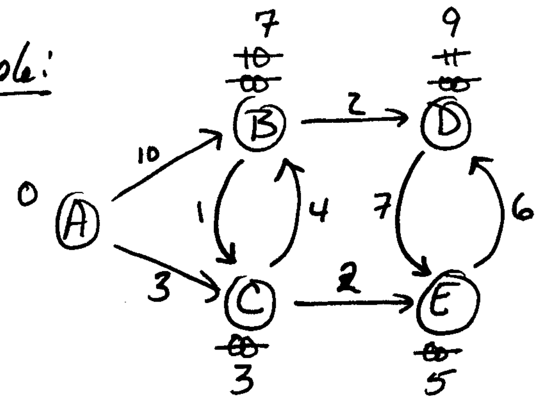
relaxation step

Implicit DECREASE-KEY



Example:

s=A



Q:	A	B	C	D	E
d:	$\boxed{0}$	∞	∞	∞	∞
		10	$\boxed{3}$	∞	∞
		7		∞	$\boxed{5}$
		$\boxed{7}$		11	
				$\boxed{9}$	

$S = \{A, C, E, B, D\}$

Did we explicitly examine all paths and take minimum?

Correctness (Part I)

Lemma: Invariant $d[v] \geq \delta(s,v)$ $\forall v \in V$ at all times.

Proof:

Init $d[s]=0$ and $d[v]=+\infty$ for $v \neq s$; $\delta(s,s)=0$ and $\delta(s,v) \leq \infty$ $\forall v$, so OK

Suppose invariant fails, that v is the first vertex with $d[v] < \delta(s,v)$ and u is the vertex that caused $d[v]$ to change by $d[v] = d[u] + w(u,v)$.

Then $d[v] < \delta(s,v)$ ← supposition

$\leq \delta(s,u) + \delta(u,v)$ ← triangle inequality

$\leq \delta(s,u) + w(u,v)$ ← shortest path ≤ specific path

$\leq d[u] + w(u,v)$ ← v is first violation, so $\delta(s,u) \leq d[u]$

$d[v] < d[u] + w(u,v)$ violates

Correctness (Part II)

Theorem: When Dijkstra's algorithm terminates, $d[v] = \delta(s,v)$ $\forall v \in V$

Proof: $d[v]$ doesn't change once added to S, so suffices to show true when added

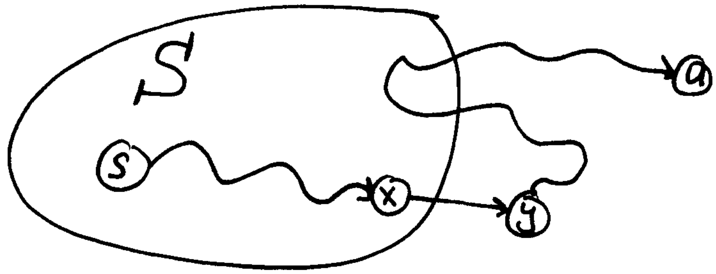
Suppose u is first vertex about to be added to S for which $d[u] \neq \delta(s,u)$

$\Longrightarrow d[u] > \delta(s,u)$ by previous lemma

Let p be a shortest path from s to u $[w(p) = \delta(s,u)]$

Consider first place p exits S [via edge (x,y)]

(y is first vertex along p in $V-S$; x is predecessor of y along p)



Because u is first violation, $d[x] = \delta(s, x)$.

When x was added to S, we relaxed (x, y) and set
$d[y] = \delta(s,x) + w(x,y) = \delta(s,y)$ because subpaths of shortest paths are shortest paths.

Thus $d[y] = \delta(s,y) \le \delta(s,u) \le d[u]$

sub-path — previous lemma

But $d[u] \le d[y]$ by Dijkstra's choice of u ← Emphasizes need for greedy step

So $d[y] = \delta(s,y) = \delta(s,u) = d[u]$

Contradiction ✓.

Analysis

$d[v] \leftarrow \infty$ for each $v \in V - \{s\}$ & initialize priority queue $O(|V|)$

while $Q \neq \emptyset$
 do $u \leftarrow$ EXTRACT-MIN(Q)
  $S \leftarrow S \cup \{u\}$
  for each $v \in Adj[u]$
   do if $d[v] > d[u] + w(u,v)$
    then $d[v] \leftarrow d[u] + w(u,v)$

$|V|$ times

degree(u) times

DECREASE-KEY : $O(|E|)$ worst-case aggregate analysis

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

(same as Prim's MST algorithm)

Q	$T_{\text{Extract-Min}}$	$T_{\text{Decrease-Key}}$	TOTAL
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$ ← for all vertices reachable
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(E + V \lg V)$ worst case

Unweighted Graphs

Suppose $w(u,v)=1 \ \forall (u,v) \in E$. Then Dijkstra's algorithm can be improved using simple FIFO queue in place of priority queue

(Breadth-First-Search) first-in first-out

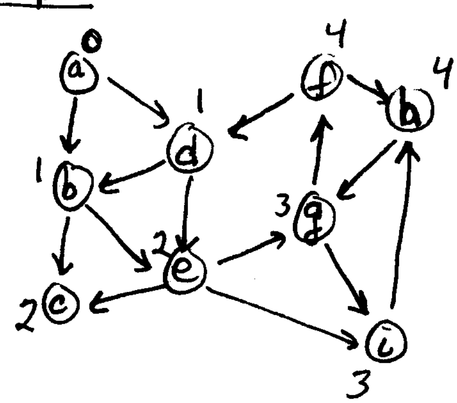
```
BFS(G, w s)
  d[s] ← 0
  d[v] ← ∞  for each v ∈ V - {s}
  Q ← {s}
  while Q ≠ ∅                      |V| vertices
    do u ← Dequeue(Q)
       for each v ∈ Adj[u]         |E| edges
         do if d[v] = ∞
              then d[v] ← d[u]+1
                   Enqueue(Q, v)
```

Analysis

Time: $O(V+E)$ All queue operations are $O(1)$; there is no Decrease-Key

Example:

s=a



$$\begin{array}{cccccccccc} & 0 & 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ Q: & \cancel{a} & \cancel{b} & \cancel{d} & \cancel{c} & \cancel{e} & \cancel{g} & \cancel{i} & \cancel{f} & \cancel{h} \end{array}$$

Correctness of BFS

Key Idea: FIFO queue in BFS mimics priority queue in Dijkstra

Invariant: v immediately after u in queue
$\Rightarrow d[v]$ is either $d[u]$ or $d[u]+1$