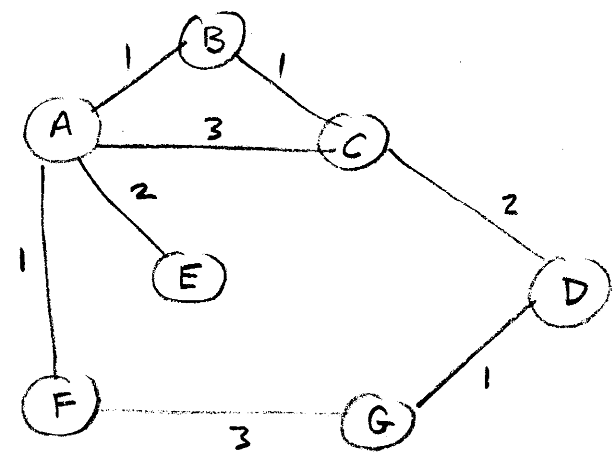
HOW TO BUILD ROUTING TABLES

- CONSIDER NETWORK AS A GRAPH



- EVERY EDGE (LINK) HAS A WEIGHT
- WEIGHT INDICATES DESIRABILITY OF SENDING TRAFFIC OVER LINK
  - COST
  - CONGESTION
  - SPEED
- THE BASIC PROBLEM IS TO FIND THE SHORTEST PATH BETWEEN EVERY PAIR OF NODES
- THE LENGTH OF PATH = SUM OF WEIGHTS OF ITS EDGES

SHORTEST PATH ROUTING

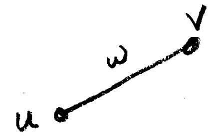
- EACH NODE DETERMINES THE NEXT NODE ON ITS PATH TO SOME SOURCE NODE S
- UPON RECEIVING A PACKET DESTINED TO S, THE NODE SENDS THE PACKET TO THE NEXT NODE OF THE PATH (NEXT-HOP)
- FOR NOW ASSUME ONE SOURCE FOR SIMPLICITY

Basic idea

- Start with a guess $d(s,v)$ = v's guess of its distance from source
- Initially

$$d(s,s) = 0$$
$$d(s,v) = \infty$$

- Relaxation : pick an edge (u,v)



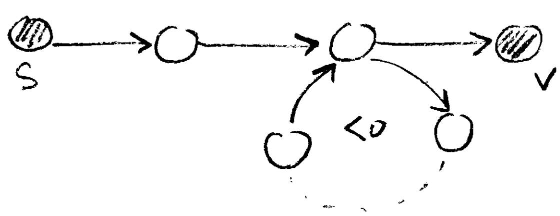
$$\text{if} \quad d(s,u) + w(u,v) < d(s,v)$$
$$\text{then} \quad d(s,v) \leftarrow d(s,u) + w(u,v)$$

(Get better estimate)

- Repeat until no better estimates can be obtained

Convergence

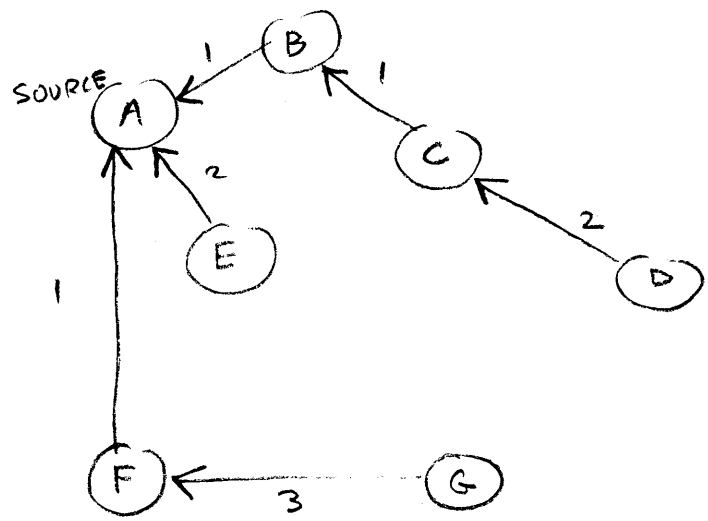
- Convergence is not always guaranteed
  - e.g. consider a negative weight cycle



Every time we go through cycle we get better estimate $\Rightarrow$ solution does not exist.

- If no negative weight cycles $\Rightarrow$ convergence
- If no zero-weight cycles $\Rightarrow$ shortest path tree can be obtained if every node keeps track of where it heard best information.

EXAMPLE



BUT WHEN CAN WE STOP PERFORMING RELAXATIONS?
IS THERE AN UPPER BOUND?

BELLMAN-FORD

GIVEN GRAPH $G = (V, E)$

VERTICES EDGES

INIT $\quad d(s,v) = \infty \quad \forall \ v \neq s$

$\quad d(s,s) = 0$

ALG.

for $i \leftarrow 1$ to $|V| - 1$

  for each edge $(u,v) \in E$

    do if $d(s,u) + w(u,v) < d(s,v)$

      then $d(s,v) \leftarrow d(s,u) + w(u,v)$

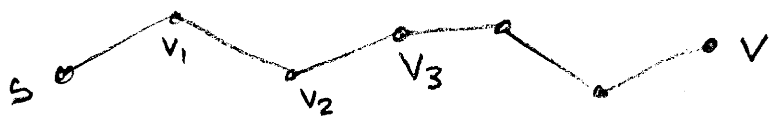
        $f(s,v) \leftarrow u$

Relax(u,v)

THIS CONVERGES IF NO NEGATIVE WEIGHT CYCLES

CONVERGENCE ARGUMENT

- CONSIDER THE SHORTEST PATH FROM s to v



- BY PROPERTY OF SHORTEST PATH, THE PATH FROM s to v_i IS ALSO SHORTEST PATH
- PROOF BY INDUCTION: DISTANCE TO V_1 WILL STABILIZE AFTER FIRST ROUND, DISTANCE TO V_2 WILL STABILIZE AFTER SECOND ROUND, etc...
- SINCE EACH PATH CAN HAVE AT MOST $|V|-1$ EDGES (NO NEG. WEIGHT CYCLES) ALL DISTANCES WILL STABILIZE AFTER $|V|-1$ ROUNDS

DISTRIBUTED BELLMAN-FORD

- NEIGHBOR NODES EXCHANGE MESSAGES OF THE FORM

$$d(s, Y)$$

"I AM Y AND I AM AT DISTANCE d FROM SOURCE

<u>NODE X</u>

<u>INIT</u>

if $(x = s)$
  then $d(s,x) = 0$
  else $d(s,x) = \infty$

<u>ALG.</u>

if $d(s,Y)$ RECEIVED (FROM Y)
  then Relax (x,y)
if $d(s,X)$ CHANGES
  then SEND $d(s,X)$ to NEIGHBORS

ALL PAIRS

- RUN BELLMAN-FORD FOR EVERY SOURCE IN PARALLEL
- EVERY NODE IS A SOURCE

<u>NODE X</u>

<u>INIT.</u> $d(X,X) = 0$
$d(V,X) = \infty$ (UNDEFINED)

<u>ALG.</u> if $d(V,Y)$ RECEIVED (FROM Y)
then if $d(V,Y) + w(X,Y) < d(V,X)$
then $d(V,X) \leftarrow d(V,Y) + w(X,Y)$
$f(V,X) \leftarrow Y$

Relax(X,Y,V)

if $d(V,X)$ CHANGES
then SEND $d(V,X)$ to All NEIGHBORS

LINK OR NODE FAILURE

- ASSUME NODE X CAN DISCOVER THAT LINK (X,Y) IS DOWN
  - LINK DOWN
  - Y DOWN (NO PERIODIC UPDATES)
  - NODE X CAN SEND A CONTROL PACKET & WAIT FOR ACK
- IN GENERAL $w(X,Y)$ CHANGES by Δ

<u>NODE X</u>

for each V s.t. $f(V,X) = Y$ (PATH THROUGH Y)
do $d(V,X) \leftarrow d(V,X) + \Delta$
SEND $d(V,X)$ TO ALL NEIGHBORS

<u>NODE Z</u>

if $d(V,X)$ RECEIVED & $f(V,Z) = X$
then $d(V,Z) = d(V,X) + w(X,Z)$

ALGORITHM

NODE X

INIT $d(X,X)=0$

ALG. if $d(V,Y)$ RECEIVED (from Y)
 then if $(f(V,X)=Y)$
  then $d(V,X)=d(V,Y)+w(X,Y)$
 else Relax(X,Y,V)

MESSAGE RECEIVED

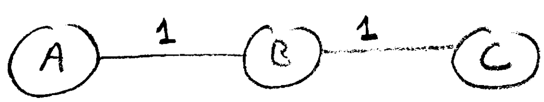
if $w(X,Y)$ CHANGED BY Δ
 then for each V s.t. $f(V,X)=Y$
  do $d(V,X)\leftarrow d(V,X)+\Delta$

LINK WEIGHT CHANGE

if $d(V,X)$ CHANGES
 then SEND $d(V,X)$ to ALL NEIGHBORS

(*) AN ALTERNATIVE IS FOR NODE X TO KEEP THE SMALLEST DISTANCE HEARD SO FAR FOR EVERY V & WHERE IT HEARD IT FROM

COUNTING TO INFINITY PROBLEM



- AFTER STABILIZATION

A	B
$d(C,A)=2$	$d(C,B)=2$
$f(C,A)=B$	$f(C,A)=C$

- ASSUME LINK B-C FAILS & B DETECTS THAT & SET $w(B,C)=\infty$ & INFORMS ITS NEIGHBORS (i.e. A)

- DEPENDING ON THE TIMING OF MESSAGES, A WEIRD BEHAVIOR CAN HAPPEN

COUNTING TO INFINITY

A	B
$d(C,A)=2$ $f(C,A)=B$	$d(C,B)=\infty$ $f(C,B)=C$
$d(C,A)=\infty$ $f(C,A)=B$	$d(C,B)=3$ $f(C,B)=A$
$d(C,A)=4$ $f(C,A)=B$	$d(C,B)=\infty$ $f(C,B)=A$
$d(C,A)=\infty$ $f(C,A)=B$	$d(C,B)=5$ $f(C,B)=A$

⋮

NO ONE REALLY KNOWS THAT C IS NOT REACHABLE

SOLUTIONS

- CONSIDER SOME LARGE ENOUGH INTEGER TO BE ∞
- SPLIT HORIZON : DON'T SEND INFORMATION TO NEIGHBOR YOU LEARNED IT FROM
  - UNFORTUNATELY DOES NOT WORK IN GENERAL FOR LARGER LOOPS, WORKS FOR ABOVE EXAMPLE (loop size 2)
- UPON INCREASE IN WEIGHT OF AN EDGE, STOP ACCEPTING MESSAGES FOR SOME TIME
  - MAKES IT SLOWER

LINK STATE

- EACH NODE KNOWS STATE OF ITS LINKS
- FLOOD THAT INFO SO THAT EACH NODE HAS A KNOWLEDGE OF WHOLE GRAPH
- EACH NODE COMPUTES THE SHORTEST PATH TREE ASSUMING IT'S SOURCE

PHASE 1 FLOODING

- EACH NODE ITS NEIGHBOR LIST & LINKS STATES TO ALL ITS NEIGHBORS IN A MESSAGE CALLED LSP (LINK STATE PACKET)
  - NODE ID
  - LIST OF NEIGHBORS & LINK WEIGHTS TO REACH THEM
  - SEQUENCE # (AVOID STALE INFO)
  - TTL (TIME TO LIVE)
- IF NODE X RECEIVES MOST RECENT LSP FROM NODE Y IT UPDATES ITS KNOWLEDGE & FORWARDS A COPY ON ALL LINKS EXCEPT (X,Y) (TO STOP FLOODING EVENTUALLY); OTHERWISE DISCARDS LSP

DIJKSTRA'S ALGORITHM

PHASE 2. COMPUTE SHORTEST PATH USING DIJKSTRA'S ALGORITHM (FASTER THAN BELLMAN-FORD, BUT ASSUMES ALL WEIGHTS ≥ 0)

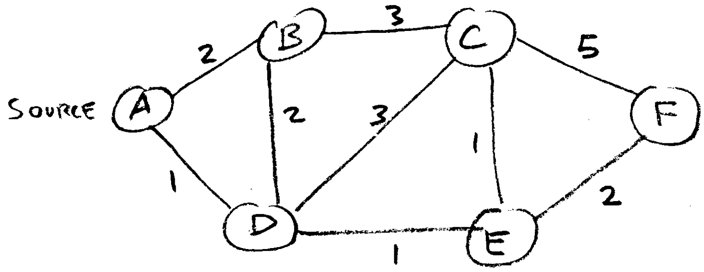
INIT

$d(s,s)=0$
$d(s,u)=\infty \quad u \neq s$

ALG

$S \leftarrow \phi$
while $(S \neq V)$
  choose $u \notin S$ with $\min_{u \notin S} d(s,u)$
  $S \leftarrow S \cup \{u\}$
  for each $v \notin S$ s.t. $(u,v) \in E$
    Relax (u,v)

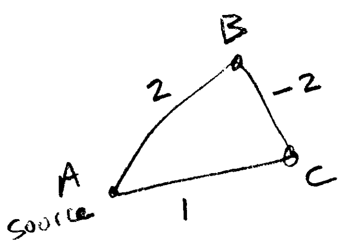
EXAMPLE



S	A	B	C	D	E	F
ϕ	0	∞	∞	∞	∞	∞
A	0	2	∞	1	∞	∞
A, D	0	2	4	1	2	∞
A, D, B	0	2	4	1	2	∞
A, D, B, E	0	2	3	1	2	4
A, D, B, E, C	0	2	3	1	2	4
A, D, B, E, C, F	0	2	3	1	2	4

WHY POSITIVE WEIGHTS?

- COUNTER EXAMPLE



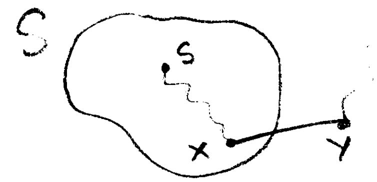
S	A	B	C
ϕ	0	∞	∞
A	0	2	1
A, C	0	-1	1
A, C, B	0	-1	1

d[A] = 0
d[B] = -1
d[C] = 1 (WRONG)

WHY IT WORKS ?

- WHEN A VERTEX U IS CHOSEN TO GO INTO S, LET'S CALL THIS FREEZING THE VERTEX
- CLAIM: WHEN u FREEZES $\Rightarrow$ computed $d(s,u) = \delta(s,u)$ WHERE $\delta(s,u)$ IS CORRECT SHORTEST DISTANCE FROM s

<u>PROOF:</u> CONSIDER 1st TIME VIOLATION OF CLAIM



← CHOSEN TO FREEZE BUT $d(s,u) > \delta(s,u)$

↖ shortest path from s to u MUST cross SET S JUST BEFORE FREEZING u

CONTINUE

$d(s,x) = \delta(s,x)$ (U IS FIRST VIOLATION)

$\Rightarrow d(s,Y) = \delta(s,Y)$ $s \cdots X - Y$ IS SHORTEST PATH FROM s to Y BY PROPERTY OF A SHORTEST PATH & EDGE (X,Y) IS RELAXED ALREADY

Now if $d(s,u) > \delta(s,u) \Rightarrow d(s,u) > \underbrace{\delta(s,Y)}_{=d(s,Y)} + \underbrace{\delta(Y,u)}_{\geq 0}$

$$d(s,u) > d(s,Y) + \varepsilon \Rightarrow$$

$$d(s,u) > d(s,Y)$$

So u WOULD NOT BE CHOSEN FOR FREEZING