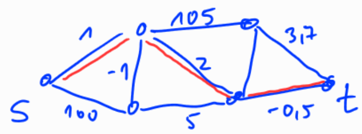
3. Shortest Path Problems

- given $G=(V,E)$ (directed/undirected)

  Find a path between $s,t \in V$ of minimum length

  length of path P is $w(P) = \sum_{e \in P} w(e)$, $w: E \to \mathbb{R}$



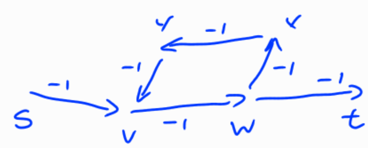
P with length 2,5 is shortest path.

The "difficulty of" this optimization problem depends on the weighting fct!

3.1. Difficult Shortest Path Problems

Let $G=(V,E)$ be a digraph & $w(e) = -1 \ \forall e \in E$

Assume,



$\Rightarrow$ we get shortes path wenn traverse cycle vwxy infintely often.

$\Rightarrow$ so we should avoid to have cycles of negativ length.

We may avoid this by searching for shortest simple paths

every vertex is contained at most ones.

Shortest Simple Path (SSP)

Input: digraph $G = (V, E)$, $w: E \to \mathbb{R}$, int. $k \in \mathbb{Z}$

Q: $\exists$ simple path in G of length $\leq k$?

Theorem: SSP is NP-complete

proof: $SSP \in NP$ ✓

let G be instance of HAM-P in digraphs,

& choose $w(e) = -1 \quad \forall e \in E(G)$.

If G has HAM. P $\iff$ #edges in P is $|V| - 1$

$\iff w(P) = (-1)(|V|-1) = 1 - |V|$

$\iff G$ has SSP of length $1 - |V|$

✓□

3.2 "Simple" shortest path problems

Lemma 3.1: Let $G = (V, E)$ be a digraph with weighting fct $w: E \to \mathbb{R}$.
Let $P = \langle v_0, v_1, \dots, v_k \rangle$ be a shortest v_0-v_k-path in G.
Then, for all i, j $[0 \leq i \leq j \leq k]$, the subpath $P_{ij} = \langle v_i, \dots, v_j \rangle$ of P is shortest v_i-v_j-path.

proof: $v_0 \overset{P_{0i}}{\rightsquigarrow} v_i \underset{P'_{ij}}{\overset{P_{ij}}{\rightsquigarrow}} v_j \overset{P_{jk}}{\rightsquigarrow} v_k \Rightarrow w(P) = w(P_{0i}) + w(P_{ij}) + w(P_{jk})$

by contradiction assuming $w(P'_{ij}) < w(P_{ij})$

$< w(P_{0i}) + w(P'_{ij}) + w(P_{jk})$ ↯

✓□

NOTE! this proof is only so simple since we did not use "simple path" for which the statement of L 3.1 is not true!

[Exercise: L. 3.1 remains true for simple path, if there are no cycles of negative lengths]

Distance between u & v is

$$\delta(u,v) = \begin{cases} \min\limits_{u\text{-}v\text{-path}} w(P) & \text{, if such path exist} \\ \infty & \text{, else} \end{cases}$$

IF G contains: 

$\Rightarrow$ $\delta(u,v) = -\infty$ not well-defined, since no "finite" shortest path exists.

In general, only digraphs with edge weights st no negative cycles appear are of interest for us.

$\hat{=}$ conservative weighting functions

Let G be a digraph with conservative weight. fct. w

Assume shortest path contains cycles: 

$w(C) > 0$ ↯ since then we can find a shorter path

$w(C) = 0 \Rightarrow$ can remove C from P & get a new shortest path

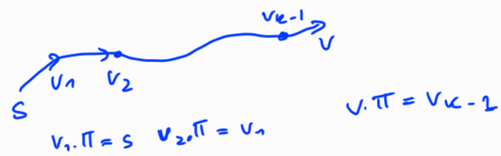
$\Rightarrow$ W.l.o.g we can assume that shortest paths are simple!

$\Rightarrow$ shortest paths have at most $|V|-1$ edges!

NOTATION: we want to know $w(P)$ of shortest path P, but also the vertices on P:

$\forall v \in V(G)$ maintain predecessor $v.\pi$ that either a vertex or NIL

The algorithms that we come up with will update $v.\pi$ & then lead to vertices in shortest path



To print these paths, we use:

```
Print_Path (G, s, v)
    IF (v = s) print s
    ELSEIF (v.π = NIL)
        print "∄ path from s to v"
    ELSE
        Print_Path (G, s, v.π)
        print v
```

[runtime $O(|V|)$ since $|E(P)| \le |V|-1$]

Moreover we use a technique called Relaxation: (misleading name)

$\forall v$ maintain attribute $v.d$ which is upper bound on distance from source vertex s to v [shortest path estimate]

1. init, then 2. "relaxing":

```
Init_single_source (G, s)
    FOR (all v ∈ V(G)) DO
        v.d = ∞
        v.π = NIL
    s.d = 0
```

[runtime $O(|V|)$]

RELAX (u, v, w) // u,v vertices, w weight.

IF $v.d > u.d + w(u,v)$
 $v.\pi = u$
 $v.d = u.d + w(u,v)$

[runtime $O(1)$]

$u.d = 7$

& $w(u,v) = 1$

RELAX

$\Rightarrow$ $v.d = 8$

$v.\pi = u$

$v.d = 10$

↑ upper bound on shortest path from s to v is 10

Properties

Lemma 3.2 [Δ-inequality]

G digraph with $w : E \to \mathbb{R}$.

$\Rightarrow$ $\delta(s v) \le \delta(s u) + w(u v)$ $\forall (u v) \in E$

proof: "$\delta(s v) < \infty$"

P_{su} with $w(P_{su}) = \delta(s u)$ shortest path.

$P' = P_{su} + (uv)$ & per Def $w(P') \ge w(P_{sv})$

"$\delta(s v) = \infty$"

$\Rightarrow$ no path exists $\Rightarrow$ since $(uv) \in E$ $\Rightarrow$ P_{su} cannot exist

$\Rightarrow$ $\delta(s, u) = \infty$ ✓

□

Lemma 3.3 [upper bound property]

G digraph with $w : E \to \mathbb{R}$, source $s \in V$

call Init_single_source (G, s)

$\Rightarrow$ $v.d \ge \delta(s, v)$ $\forall v \in V$

& this property is maintained over any sequence of calls of RELAX

In particular, if $v.d = \delta(s, v)$ then $v.d$ never changes again.

proof: by induction over nr of calls of RELAX.

Base case (no call): $v.d = \infty \ge \delta(s, v)$ $\forall v \in V \setminus \{s\}$

$s.d = 0 \ge \delta(s, s)$ [$\delta(s,s) = 0$, or $\delta(ss) = -\infty$ if negative cycle from s to s]

consider call RELAX $(u v w)$

& assume $x.d \ge \delta(s, x)$ $\forall x \in V$ prior to this call.

$\Rightarrow$ only $v.d$ may change and if it changes

we have $v.d = u.d + w(uv) \underset{\text{Ind hyp}}{\ge} \delta(s, u) + w(uv) \underset{\text{Δ-ineq. L. 3.2}}{\ge} \delta(s, v)$ ✓

We have shown $v.d \ge \delta(s, v)$ & RELAX does not increase values $\Rightarrow$ if $v.d = \delta(s, v)$ it never changes ✓

□

Corollary 3.4

[NO-PATH property]

G digraph with $w: E \to \mathbb{R}$, source $s \in V$ & $\nexists$ path from s to some $v \in V$

$\Rightarrow$ $v.d = \delta(s,v) = \infty$ &

this property is maintained over any sequence of calls of RELAX

Lemma 3.5:

[convergence property]

G digraph with $w: E \to \mathbb{R}$, source $s \in V$,

& let $P =$ "$s \rightsquigarrow u \to v$" shortest s–v path, for some $u, v \in V$

Suppose we use Init_single_source(G, s)

+ sequence of calls of RELAX$(\ldots)$

including call of RELAX(u, v, w)

IF $u.d = \delta(s,u)$ at any time prior to this call

THEN $v.d = \delta(s,v)$ at all times after this call.

Proof: IF $u.d = \delta(s,u)$ this never changes again (L. 3.3)

AFTER call of RELAX(u,v,w) we have:

$$v.d \leq u.d + w(uv)$$

[either $v.d > u.d + w(uv)$ & we put $v.d = u.d + w(uv)$ or $v.d \leq u.d + w(uv)$]

$$= \delta(s,u) + w(uv)$$

$$= \delta(s,v)$$

shortest path $s \rightsquigarrow u \to v$ & L.3.1

By L.3.3: $v.d \geq \delta(s,v)$ $\Rightarrow$ $v.d = \delta(s,v)$ & this property is maintained $\square$

Lemma 3.6

[path-relaxation property]

G digraph with $w: E \to \mathbb{R}$, source $s \in V$,

$P = \langle v_0, v_1, \ldots, v_k \rangle$ any shortest s–v_k path. $(k < \infty)$, $v_0 = s$

Suppose we use Init_single_source(G, s)

+ sequence of calls of RELAX$(\ldots)$

that includes – in this order (!) –

calls RELAX$(v_0, v_1, \cdot)$

RELAX$(v_1, v_2, \cdot)$

$\vdots$

RELAX$(v_{k-1}, v_k, \cdot)$

Then $v_k.d = \delta(s, v_k)$ after these calls & this property is maintained.

<u>Proof:</u> by induction that after i-th edge of P is called in RELAX(v_{i-1}, v_i)

$i = 0$ (no edge of P) : $v.d = s.d = 0 = \delta(s,s)$

↑ since $k < \infty$ in P

Assume $v_{i-1}.d = \delta(s, v_{i-1})$

$\Rightarrow$ $s \rightsquigarrow v_{i-1} \to v_i$ & by L. 3.5 the statement is true. $\square$

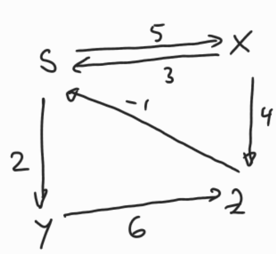
The Bellmann-Ford Algorithm

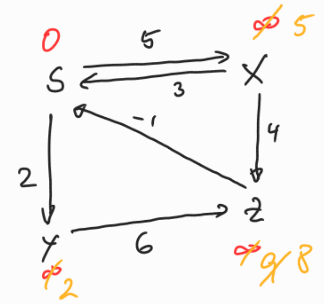
- solves "single source shortest path problem"
  = find from source to all other vertices the shortest path.

```
BELLMANN-FORD (G, w, s)        // input G = (V,E), conservative weight
                                                    w: E -> R,
   Init-single-source (G, s)                        source s
   FOR  i = 1 .. |V|-1 )
        FOR ( every (uv) ∈ E ) DO
             RELAX (u, v, w)
```

(each edge is simply "relaxed" $|V| - 1$ times)

Can be extended to deal with arbitrary weights by reporting negative cycles, if there are such cycles from s to to some v, then algorithms returns FALSE.





change = c // nochange = nc

sx	c	nc
sy	c	nc
xs	nc	nc
xz	c	nc
yz	c	nc
zs	nc	nc.

edges are relaxed in order

(s x)(s y)(x s)(x z)(y z)(z s)

	s	x	y	z	
π	NIL	NIL	NIL	NIL	after init-single-source.
		s	s	~~x~~ y	i = 1
		s	s	y	i = 2

[nothing changes in last iteration i = 3 = 4 - 1]

$\boxed{\begin{array}{l}(uv): \text{ changes may happen in RELAX for } v: \\ v.d > u.d + w(uv) \to v.d \,/\, v.\pi\end{array}}$

Theorem 3.7: Let $G = (V, E)$ be digraph with conservative weighting function $w: E \to \mathbb{R}$

[correctness + runtime]

Then BELLMANN-FORD terminates in $O(|V||E|)$ time & after it terminates it holds $v.d = \delta(s,v)\ \forall v \in V$.
& PRINT-path(G, s, v) "prints shortest path from s to v"

proof: **Runtime**: $O(|V| + |V| \cdot |E|) = O(|V||E|)$ clear.

init-single-source

FOR : $|V|-1$
FOR : $|E|$

Correctness: let $v \in V$.

1. Assume $\exists$ s-v-path. & let $P = \langle s = v_0, v_1, \dots, v_k = v \rangle$ shortest path from s to v

! shortest path are simple $\Rightarrow |E(P)| \leq |V| - 1 \Rightarrow k \leq |V| - 1$
since we have conservative weights

In each of $|V| - 1$ iterations, all edges are "relaxed".
& in i-th iteration, we also have RELAX(v_{i-1}, v_i, w)
& in previous iterations, we had:

RELAX(v_0, v_1, w)
RELAX(v_1, v_2, w)
$\vdots$
RELAX(v_{i-2}, v_{i-1}, w)

By induction & L 3.6 $\Rightarrow$ in i-th iteration $v_i.d = \delta(s, v_i)$

$\Rightarrow v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$

2. IF $\nexists$ s-v-path THEN COR 3.4 implies $v.d = \delta(s,v) = \infty$.

Print graph: [Exus.]

$\square$

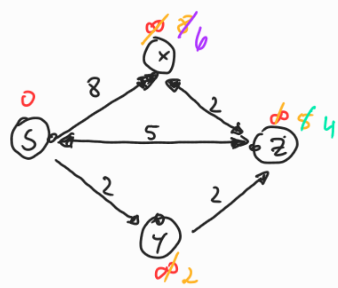
Dijkstra's Algorithm

- solves "single source shortest path problem"
  = find from source to all other vertices the shortest path.

This algorithm is "faster" but works only for $w: E \to \mathbb{R}_{\geq 0}$

```
DIJKSTRA (G, w, s)
    Init_single-source (G, s)
    S = ∅
    Q := V(G)
    WHILE (Q ≠ ∅)
        u = Extract_MIN (Q)     // return u = min_{w∈Q} { w.d } & remove u from Q
        S = S ∪ {u}             // just needed for correctness proof
        FOR (all v ∈ N⁺(u))     // for all edges (u,v)
            RELAX (u, v, w)
```

INIT



1. u = s RELAX (s, x, w) RELAX (s, z, w) RELAX (s, y, w) $\xrightarrow{S=\{s\}}$ x.d = 8, z.d = 5, y.d = 2 ✓

2. u = y RELAX (y, z, w) $\xrightarrow{S=\{s,y\}}$ z.d = 4 ✓

3. u = z RELAX (z, s, w) RELAX (z, x, w) $\xrightarrow{S=\{s,y,z\}}$ s.d = 0 no change, x.d = 4+2 = 6

4. u = x RELAX (x z w) $\xrightarrow{S=\{s,y,z,x\}}$ ✓

S	X	Y	Z
NIL	NIL	NIL	NIL
NIL	S	S	S
	S	S	Y
	Z	S	Y

Theorem 3.8

Dijkstra's-Alg on digraph $G = (V, E)$, non-neg. $w: E \to \mathbb{R}_{\geq 0}$, source s

terminates & $u.d = \delta(s,u)\ \forall u \in V$

& print-path(G, s, u) "prints shortest path from s to u".

proof: suffices to show that when u is added to S then $u.d = \delta(s,u)$

1. init $S = \emptyset$ ✓

2. assume, for contradiction, $\exists\ u \in V$ that when added to S does not satisfy $u.d = \delta(s,v)$.

let u be the first vertex for which this happens.

Note $u \neq s$ since 12.7_single-source $s.d = 0 = \delta(s,v)$
& s added to S in "first step". $\Rightarrow S \neq \emptyset$

& there is a $s-u$ path in G
[otherwise, COR 3.4 states $u.d = \infty = \delta(s,u)$ for all steps]

$\Rightarrow \exists$ shortest $s-u$ path P_{su}

S contains vertices of P
since $s \in S$.

Let y be first vertex on P_{su}
that is not in S
& x its predecessor in P_{su}

Since u first vertex added to S st $u.d \neq \delta(su)$

$\underset{x \in S}{\Rightarrow}$ $x.d = \delta(s,x)$

Claim: $y.d = \delta(s,y)$ when u is added to S

proof: $x \in S \Rightarrow x$ was examined before u
$\Rightarrow y \in N^+(x)$ we called $\rightarrow$ RELAX(x,y,w)
$\overset{L.3.5}{\Longrightarrow}$ $y.d = \delta(s,y)$. //

Since $w(e) \geq 0\ \forall e \in E$ & y lies on shortest path $P_{su} \Rightarrow \delta(sy) \leq \delta(su)$

$\Rightarrow y.d = \delta(sy) \leq \delta(s,u) \underset{L\ 3.3}{\leq} u.d$

Since $u, y \in V \setminus S$ at the time where $u =$ EXTRAC-MIN $(Q) \Rightarrow u.d \leq y.d$

$\Rightarrow y.d = \delta(sy) = \delta(su) = u.d$ ↯ choice of u.

at termination we have $Q = \emptyset$ & $S = V \Rightarrow \forall v \in S:\ v.d = \delta(s,v)$.

[print-path: Exuc]

/ □

Runtime:

DIJKSTRA (G, w, s)

```
Init_single-source (G, s)
S = ∅
Q = V(G)
WHILE (Q ≠ ∅)
    u = Extract_MIN (Q)
    S = S ∪ {u}
    FOR (all v ∈ N⁺(u))
        RELAX (u, v, w)
```

WHILE-loop $|V|$ times

Let Extract-MIN have runtime $f(|V|)$

each vertex $v \in V$ extracted ones

& its $\underline{N^+(v)}$ is used

$\Rightarrow \sum_{v \in V} deg^+(v) \leq 2|E|$ [Exus]

$\Rightarrow$ Total runtime

$O(|V| \cdot f(|V|) + |E|)$

Trivially $f(|V|) \in O(|V|)$ // check all $O(|V|)$ entries of Q

However using a fancy datastructure (min-priority queue via Fibonacci-Heap)

$f(|V|) \in O(\log_2(|V|))$

$\Rightarrow$ faster than Bellmann-Ford.

To determine shortest paths between all vertices u & v we could simply run one of the previous algorithms $|V|$ times with source $s = v \ \forall v \in V$

We will introduce a further algorithm <u>Floyd-Warshall-Alg</u> to address the latter problem

& to show a principle algorithm-design technique:

<u>Dynamic Programming</u>