Bellman-Ford Algorithm

Input: Graph $G = (V, E)$ – directed graph

$w: E \to \mathbb{Z}$ – positive and negative weights

$(\mathbb{R})$

source $s \in V$

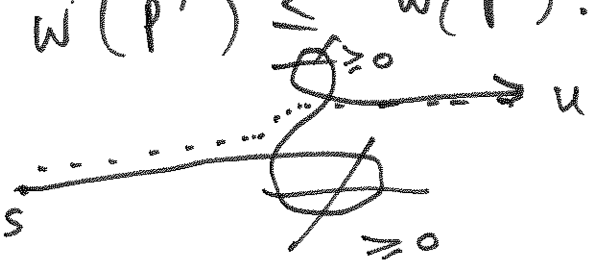
Output: either: (1) For each $u \in V$: $\delta(s,u)$, $\pi[u]$ if G has no negative cycles

or (2) Discover a negative cycle in G.

Negative cycle: cycle C: $\sum_{e \in C} w(e) < 0$.

Idea: (1) Def: $d_k(u)$ = length of a shortest (simple) path from s to u using at most k edges.

(2) If a graph has no negative cycles, a path P from s to u that is not simple can be used to get a path P' from s to u that is simple, $w(P') \leq w(P)$.



$\Rightarrow$ shortest path with no constraints on simplicity is fine.

Recursion for d_K:

$$d_0(s) = 0 \qquad d_0(u) = \infty \text{ for } u \neq s.$$

$$d_K(u) = \min_{(p,u)\in E} \left\{ d_{K-1}(p) + w(p,u),\ d_{K-1}(u) \right\}, \text{ for } K > 0.$$

$$\delta(s,u) = d_{|V|-1}(u)$$ because a simple shortest path uses at most $|V|-1$ edges.

Bellman-Ford Algorithm - Take 1:

Dynamic Program that stores $d_K(u)$ in $D[K,u]$.

```
// Base case:
for u ∈ V do  D[0, u] ← ∞
  D[0, s] ← 0
// Recursive case: Solve in ordered increasing values of K
O(|V|·|E|)  for K ← 1 to |V| - 1 do
              for u ∈ V do
                D[K,u] ← D[K-1, u]
  O(|E|)        for each edge e = (p,u) into u do
                  if D[K,u] > D[K-1,p] + w(p,u) then
                    D[K,u] ← D[K-1,p] + w(p,u).
for u ∈ V do
  for each edge (p,u) ∈ E do
    if D[|V|-1, u] > D[|V|-1, p] + w(p,u) then
```

return null // Graph has a negative cycle.

return D. // $\delta(s,u) = D[|V|-1, u]$.

Bellman-Ford Take 2:

1. No need to have a dimension for k in D.

$$\underset{k=0..|V|-1}{D[k,u]} \longrightarrow D[u] \longrightarrow u.d$$

2. Order in which nodes/edges are processed is unimportant.

// Version given in algorithms text books:

```
initialize(s)
for k ← 1 to |V| - 1 do
    for each edge e = (u,v) ∈ E do
        relax(u,v, e).

// Verification phase
for each edge e = (u,v) ∈ E do
    if (relax(u,v,e)) then
        return null
return instance of Bellman-Ford that stores .d, .π for vertices.
```

Bellman-Ford Take 3 - practical version

Queue q of vertices. $q.add(s)$

initialize(s) $\leftarrow$ include $u.count \leftarrow 0$

while q is not empty do

  $u \leftarrow q.remove()$; $u.count{+}{+}$; if $u.count > |V|-1$ then ...

  for each edge $e = (u,v)$ out of u do

    if relax(u,v,e) and $v \notin q$ then

      $q.add(v)$.

Good news: RT is much better for many graphs.

Bad news: infinite loop if G has a negative cycle.

Prevent infinite loop: count how many times u is added to q for each $u \in V$. If $u.count > |V|-1$ then G has a negative cycle.

Worst case RT is still $O(|V| \cdot |E|)$.

All-pairs shortest paths

Input: $G = (V,E)$, $w : E \to \mathbb{Z}$ (or $\mathbb{R}$), no negative cycles.

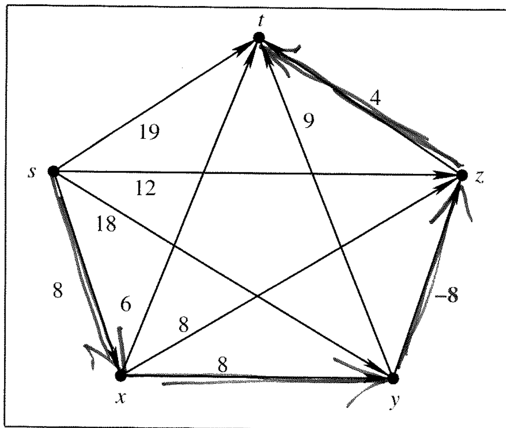
Output: $\delta(u,v)$ for all $u,v \in V$, $\pi(u,v)$

  $\pi(u,v)$ = predecessor of v in shortest path from u to v.

If edge weights are positive $\to$ run Dijkstra's algorithm from every node as source.

Shortest path worksheet

Bellman-Ford algorithm. Take 2



u	k=0 $u.d$	$u.\pi$
s	0	
t	~~∞~~ ~~19~~ ~~16~~ ~~14~~ 12	~~s~~ ~~z~~ ~~x~~ z
x	~~∞~~ 8	s
y	~~∞~~ ~~18~~ 16	~~s~~ x
z	~~∞~~ ~~12~~ ~~10~~ 8	~~s~~ ~~x~~ y

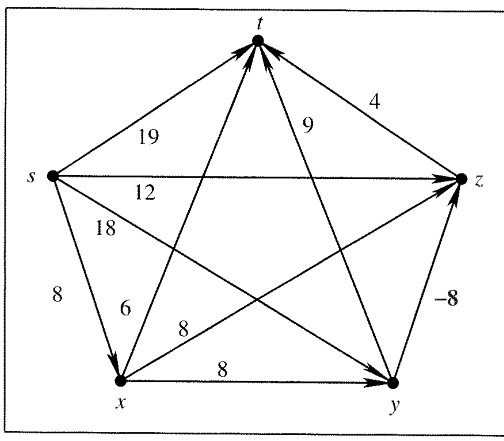
Edge order (say): (s,t) (s,z) (z,t) (s,y) (y,t) (y,z) (s,x) (x,t) (x,z) (x,y)

K = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

© Balaji Raghavachari 219 Algorithm Design and Analysis

Shortest path worksheet

Bellman-Ford Take 3



u	$u.d$	$u.\pi$
s	0	
t	~~∞~~ ~~19~~ ~~16~~ ~~14~~ 12	~~s~~ ~~z~~ ~~x~~ z
x	~~∞~~ 8	s
y	~~∞~~ ~~18~~ 16	~~s~~ x
z	~~∞~~ ~~12~~ ~~10~~ 8	~~s~~ ~~y~~ y

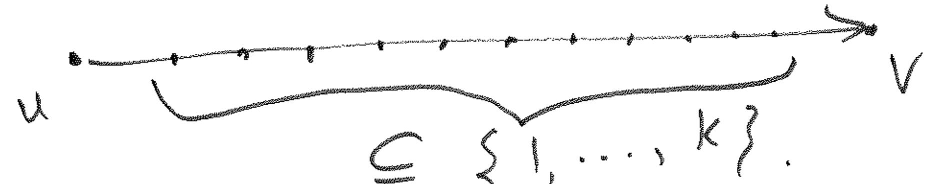
q: ~~s~~ ~~t~~ ~~z~~ ~~y~~ ~~x~~ ~~t~~ ~~z~~ ~~y~~ ~~z~~ ~~t~~

Floyd-Warshall's algorithm

G is directed, some edges have negative weights.

Based on the following recurrence:

<u>Def</u>: $d^{(k)}(u,v)$ = Weight of shortest path from u to v, all of whose internal nodes are only from $\{1, 2, \dots, k\}$.



Recurrence for $d^{(k)}$:

$$d^{(0)}(u,v) = w(u,v) \quad (\infty \text{ if no such edge})$$

$$d^{(k)}(u,v) = \min\left\{ d^{(k-1)}(u,v),\ d^{(k-1)}(u,k) + d^{(k-1)}(k,v) \right\}$$

case 1: if k is not on the path

case 2: k is on the path.

$$\delta(u,v) = d^{(|V|)}(u,v).$$

Dynamic program for this recurrence $\quad d^{(k)}(u,v) \longrightarrow D[u,v]$

```
D ← W        // Weight matrix, with ∞ in missing
             //   edges, 0 in the diagonals
for k ← 1 to |V| do
    for u ∈ V do
        for v ∈ V do
            if D[u,v] > D[u,k] + D[k,v] then
                D[u,v] ← D[u,k] + D[k,v]
return D
```

$\Theta(|V|^3)$.