# Lec 17 shortest paths I: Dijkstra算法、广度优先搜索

paths

- consider digraph $G=(V,E)$ with edge weights given by function $w: E \to \mathbb{R}$
- path $p = v_1 \to v_2 \to \cdots \to v_k$ has weight $w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$

shortest path from $u$ to $v$ is a path of minimum weight from $u$ to $v$

shortest-path weight from $u$ to $v$ is denoted by $\delta(u,v) = \min\{w(p),\ p: \text{from } u \text{ to } v\}$

negative edge weights $\Longrightarrow$ some shortest paths may not exist


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

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

Optimal Substructure

a subpath of a shortest path is a shortest path

proof: (Cut & Paste)

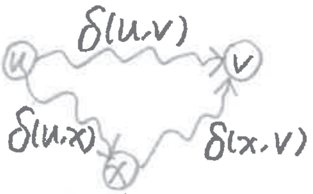

"hypothetical shorter path"

留作习题答案略，读者自证不难。

Triangle Inequality

for all vertices $u, v, x \in V$, $\delta(u,v) \le \delta(u,x) + \delta(x,v)$

proof:




留作习题答案略，读者自证不难。

single-source shorstest paths problem

from given source vertex $s \in V$, to find shortest-path weights $\delta(s,v)$ for all $v \in V$

today: assume $w(u,v) \ge 0,\ \forall u,v \in V \Longrightarrow$ shortest paths exist if paths exist

$\Longrightarrow \delta(u,v) > -\infty$

idea: greedy

① maintain set $S$ of vertices whose shortest-path distance from $s$ is known ($s \in S$)

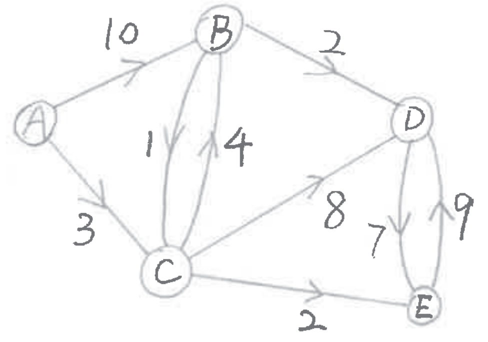
② at each step, add to $S$ the vertex $v \in V-S$, whose estimated distance from $s$ is minimum

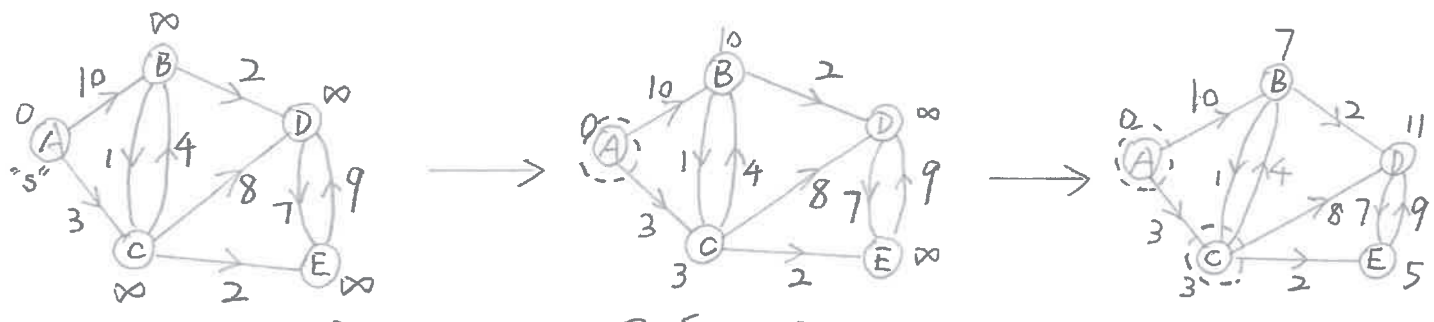
③ update distance estimates of vertices that are adjacent to $v$

## Dijkstra's Algorithm

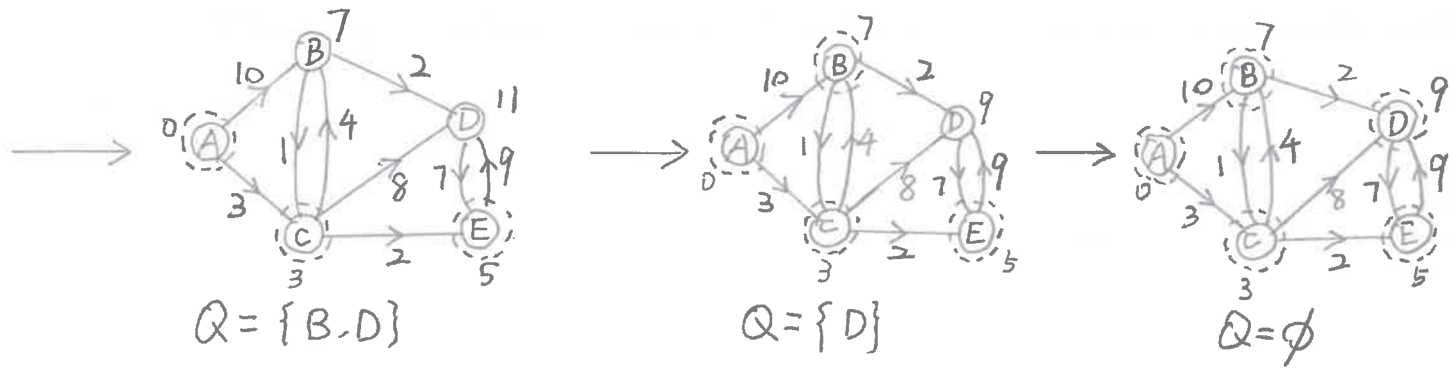
```
s.distance ← 0          // x.distance is the estimated distance from s to x
                        //   = δ(s.x) when add x to S
for each v∈V-{s}
    do v.distance ← ∞
S ← ∅
Q ← V     // priority queue, keyed on distance

while Q≠∅
    do u ← Extract-Min(Q)
    S ← S∪{u}
    for each v∈Adj[u]
        do if v.distance > u.distance + w(u.v)
            then v.distance ← u.distance + w(u.v)   } relaxation step
                              (u.distance = δ(s.u))
```

Example:




$Q=\{A,B,C,D,E\}$ → $Q=\{B,C,D,E\}$ → $Q=\{B,D,E\}$




shortest-path tree

for each vertex v
consider last edge (u, v) relaxed

Correctness I

invariant: $v.distance \geq \delta(s,v)$ for all $v \in V$

holds after initialization, and any sequence of relaxation steps

proof: (induction)

initially, $s.distance = 0$ and $v.distance = \infty \ \forall v \in V - \{s\}$

$\delta(s,s) = 0$ and $\delta(s,v) \leq \infty$

suppose for contradiction that invariant is violated

consider first violation $v.distance < \delta(s,v)$ is caused by

relaxation $v.distance \leftarrow u.distance + w(u,v)$

so $u.distance + w(u,v) < \delta(s,v)$

but $u.distance + w(u,v) \geq \delta(s,u) + w(u,v) \geq \delta(s,u) + \delta(u,v) \geq \delta(s,v)$

contradiction! Q.E.D.

Correctness Lemma

suppose $s \to \cdots \to u \to v$ is a shortest path from s to v,

if $u.distance = \delta(s,u)$ and we relax that edge $(u,v)$,

then $v.distance = \delta(s,v)$ after relaxation

proof

$\delta(s,v) = w(s \to \cdots \to u) + w(u,v) = \delta(s,u) + w(u,v)$

Correctness I $\Rightarrow v.distance \geq \delta(s,v)$

either $v.distance = \delta(s,v)$ before relaxation $\Rightarrow$ done

or $v.distance > \delta(s,v) = u.distance + w(u,v)$ before relaxation

$\Rightarrow$ we relax and set $v.distance \leftarrow u.distance + w(u,v) = \delta(s,v)$

Q.E.D.

## Correctness II

when Dijkstra terminates, $v.distance = \delta(s,v)$ for $\forall v \in V$

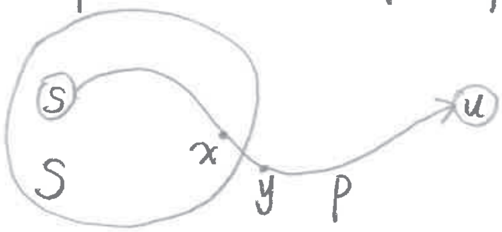
proof:

$v.distance$ doesn't change once $v$ is added to $S$

$\Rightarrow$ it suffices to prove that $v.distance = \delta(s,v)$ when $v$ is added to $S$

suppose for contradiction that $u$ is the first vertex (about to be) added to $S$, for which $u.distance \neq \delta(s,u) \Rightarrow u.distance > \delta(s,u)$

let $p$ be the shortest path from $s$ to $u$ $\Rightarrow w(p) = \delta(s,u)$




consider first edge $(x,y)$ where $p$ exits $S$

$\because$ first violation

$\therefore$ $x.distance = \delta(s,x)$

when we add $x$ to $S$, we relax $(x,y)$

by Correctness Lemma, $y.distance = \delta(s,y) \leq \delta(s,u)$

but when we are about to add $u$ to $S$, that means $u.distance \leq y.distance$

so $u.distance \leq y.distance \leq \delta(s,u)$

contradiction! Q.E.D.

## Unweighted Graphs

BFS!

use FIFO queue instead of priority queue

relax if $v.distance = \infty$ then $v.distance \leftarrow u.distance + 1$

add $v$ to the end of queue

# Lec 18 shortest path Ⅱ Bellman和差分约束系统

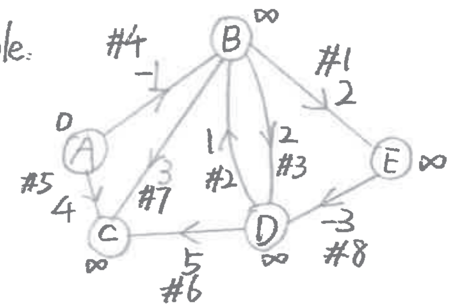
## Bellman-Ford Algorithm

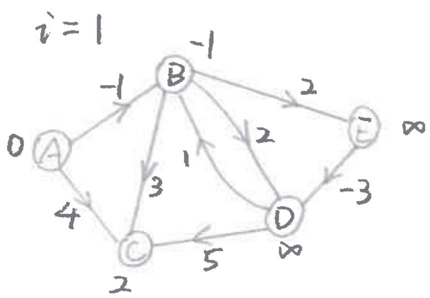
- computes shortest-path weights $\delta(s,v)$ from source vertex $s \in V$ to all vertices $v \in V$
- OR reports that a negative-weight cycle exists

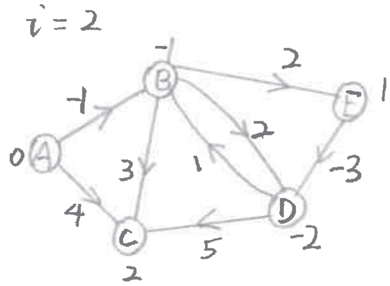
Bellman-Ford

```
s.distance ← 0
for each v∈V-{s}
    do v.distance ← ∞
for i ← 1 to |V|-1
    do for each edge (u,v)∈E
        do if v.distance > u.distance + w(u,v)
            then v.distance ← u.distance + w(u,v)
for each edge (u,v)∈E
    do if v.distance > u.distance + w(u,v)
        then report that a negative-weight cycle exists
    else
        v.distance = δ(s,v)
```

Example:







$i=3$

no change

$i=4$

no change

Correctness

if $G=(V,E)$ has no negative-weight cycle

then Bellman-Ford terminates with $v.distance = \delta(s,v)$ for all $v \in V$

proof:

consider any $v \in V$

by monotonicity of distance values, and Correctness I ( $v.distance \geq \delta(s,v)$ )

only need to show that $v.distance = \delta(s,v)$ at some time

let $p = \underset{\underset{s}{\parallel}}{v_0} \to v_1 \to v_2 \to \cdots \to \underset{\underset{v}{\parallel}}{v_k}$ be the shortest path from s to v

with minimum number of edges 可能有零权环

$\Longrightarrow$ p is a simple path

$\delta(s,v_i) = \delta(s,v_{i-1}) + w(v_{i-1},v_i)$ by optimal substructure

$v_0.distance = 0$ initially

$\delta(s,v_0)$ has to be 0 or there exists a negative-weight cycle

so $v_0.distance = \delta(s,v_0)$. base case check!

assume by induction that $v_j.distance = \delta(s,v_j)$ after j rounds, $j<i$

after $i-1$ rounds, $v_{i-1}.distance = \delta(s,v_{i-1})$

during the i-th round, relax $(v_{i-1},v_i)$

Correctness Lemma $\Longrightarrow$ $v_i.distance = \delta(s,v_i)$

after k rounds, $v_k.distance = \delta(s,v_k)$, $k \leq |V|-1$ because p is simple

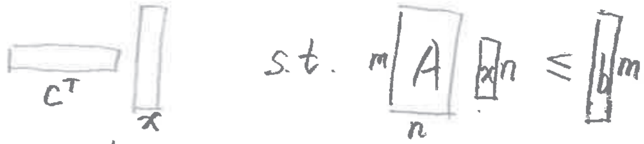
Corollary

if Bellman-Ford fails to converge after $|V|-1$ rounds,

then there has to be a negative-weight cycle

# Linear Programming

given $m \times n$ matrix $A$, $m$-vector $b$, $n$-vector $c$,
to find $n$-vector $x$ that maximizes $c^T x$ subject to $Ax \le b$,
or no such $x$ exists.




many efficient algorithms to solve LPs,

① simplex algorithm 单-算法
② ellipsoid algorithm 椭球算法
③ interior point algorithm 内点法
④ random sampling 随机抽样

# Linear Feasibility Problem

no objective $c$, just find $x$ s.t. $Ax \le b$

## Difference Constraints

linear feasibility problem where each row of matrix $A$ has one 1 and one -1
rest is 0

each constraint is of form $x_j - x_i \le w_{ij}$

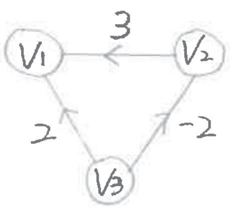
Example

$x_1 - x_2 \le 3$
$x_2 - x_3 \le -2$
$x_1 - x_3 \le 2$

## Constraint Graph

$x_j - x_i \le w_{ij} \Longrightarrow v_i \xrightarrow{w_{ij}} v_j$

$|V| = n$
$|E| = m$



$$x_j - x_i \le W_{ij}$$
$$\Rightarrow x_j \le x_i + W_{ij}$$
$$\Rightarrow j.distance \le i.distance + W_{ij}$$

Theorem:

if constraint graph has a negative-weight cycle,
then difference constraints are unsatisfiable

proof:

$V_1 \to V_2 \to \cdots \to V_k \to V_1$ is a negative-weight cycle

$$\because \begin{aligned} x_2 - x_1 &\le W_{12} \\ x_3 - x_2 &\le W_{23} \\ \cdots\ \ &\cdots\ \ \cdots \\ x_k - x_{k-1} &\le W_{k-1,k} \\ x_1 - x_k &\le W_{k,1} \end{aligned}$$

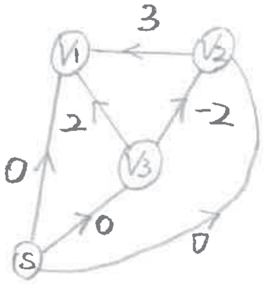
$$\therefore\ 0 \le W_{1,2} + W_{2,3} + \cdots + W_{k-1,k} + W_{k,1} = W(cycle) < 0$$

"这些约束是矛盾的! 没有这样一组 $\{x_1, x_2, \cdots, x_k\}$ 可以满足所有的约束!"

Theorem:

if no negative-weight cycle in constraint graph G,
then difference constraints are satisfiable

proof: add a new vertex s to G with a weight-0 edge from s to all $v \in V$




modified graph has no negative-weight cycles and has paths from s
$\Rightarrow$ has shortest paths from s

assign $x_i = \delta(s, v_i)$,

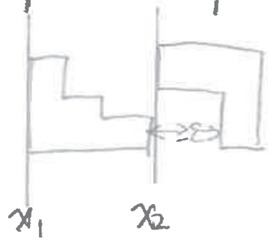
$$x_j - x_i \le W_{ij} \Leftrightarrow \delta(s, v_j) - \delta(s, v_i) \le W_{ij}$$
$$\Leftrightarrow \delta(s, v_j) \le \delta(s, v_i) + W_{ij} \qquad Q.E.D. ???$$

Corollary:

Bellman-Ford solves a system of m difference constraints on n variables in $O(mn)$ time

VLSI layout

place IC features without putting any two of them too close to each other




$x_2 - x_1 \geq \text{distance} + \varepsilon$

Bellman-Ford solves these constraints and minimizes the spread

"compactness"

# Lec 19 shortest paths Ⅲ 点的最短路径

## All-Pairs Shortest Paths

- unweighted graph: $|V| \times$ BFS $= O(|V||E|)$
- non-negative edge weights: $|V| \times$ Dijkstra $= O(|V||E| + |V|\lg|V|)$
- general:

  $|V| \times$ Bellman-Ford $= O(|V|^2 \cdot |E|)$

  three algorithms today

## Problem

input: digraph $G=(V,E)$, say $V=\{1,2,\cdots,n\}$, edge-weight function $W: E \to R$

output: $n \times n$ matrix of shortest-path weights, $\delta(i,j)$ for $\forall i,j \in V$

① dynamic programming $O(n^4)$

matrix multiplication $O(n^3)$

② Floyd-Warshall Algorithm

③ Johnson's Algorithm