湖南大學
HUNAN UNIVERSITY

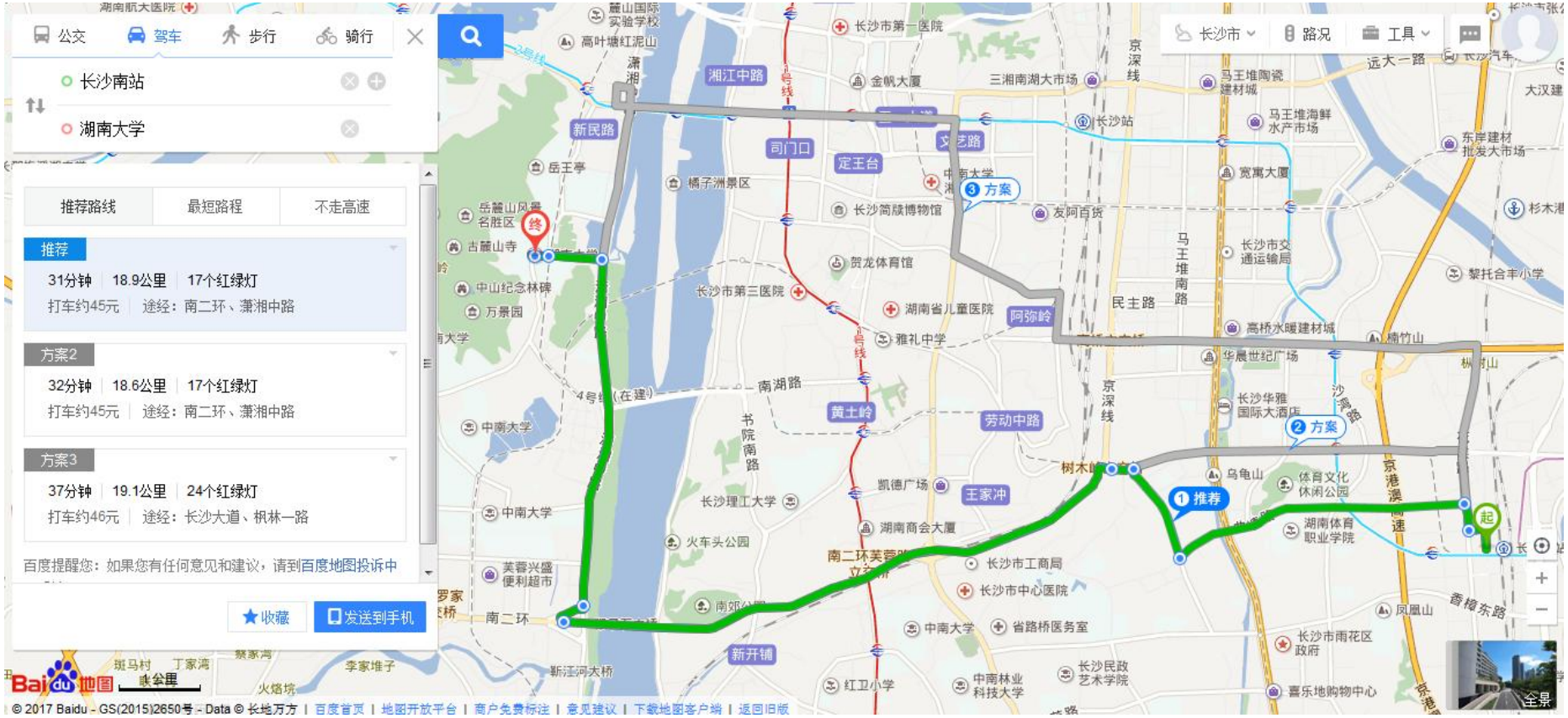
第十二章 单源最短路径

湖南大学信息科学与工程学院

湖南大学 HUNAN UNIVERSITY

目录

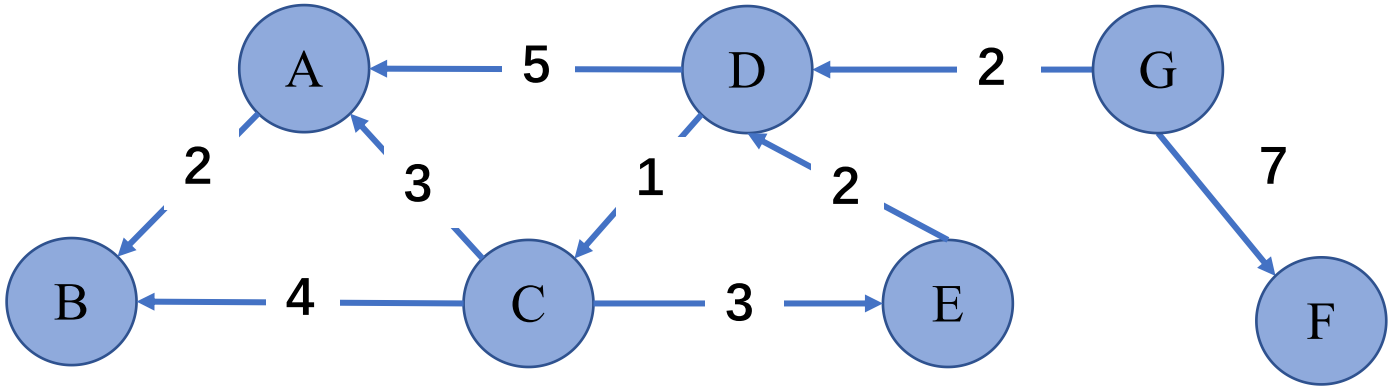
最短路径问题——应用


公交
驾车
步行
骑行
长沙南站
湖南大学
推荐路线
最短路程
不走高速
推荐
31分钟 | 18.9公里 | 17个红绿灯
打车约45元 | 途经：南二环、潇湘中路
方案2
32分钟 | 18.6公里 | 17个红绿灯
打车约45元 | 途经：南二环、潇湘中路
方案3
37分钟 | 19.1公里 | 24个红绿灯
打车约46元 | 途经：长沙大道、枫林一路
百度提醒您：如果您有任何意见和建议，请到百度地图投诉中
收藏
发送到手机
长沙市
路况
工具
湘江中路
新民路
司门口
定王台
南湖路
黄土岭
劳动中路
阿弥岭
方案
推荐
起
全景
© 2017 Baidu - GS(2015)2650号 - Data © 长地万方

最短路径问题

带权图：

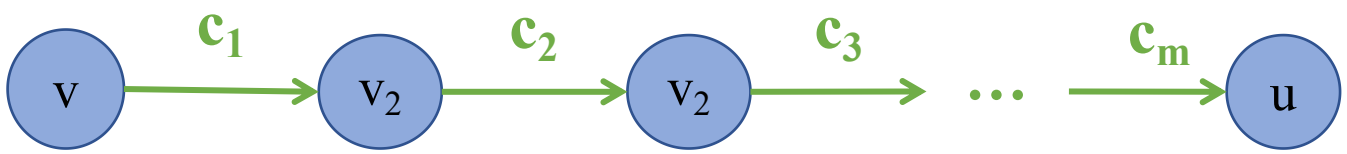
带权图可以表示成一个三元组G=<V, E, W>，其中V 是点集合，$E \subseteq V \times V$ 表示边集合，其中每条边由V 中两个点相连接所构成，W: $E \to R$将E中每条边(u,v)被赋上一个权重，记为w (u,v)



最短路径问题

最短路径：

考虑带权有向图，把一条路径（仅仅考虑简单路径）上所经边的权值之和定义为该路径的路径长度或称带权路径长度。



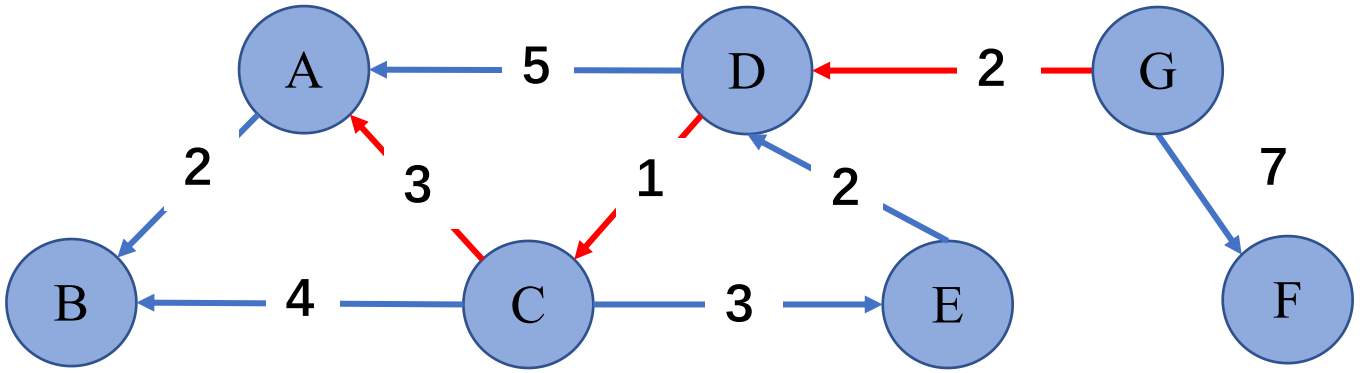
路径长度=$c_1 + c_2 + \ldots + c_m$

路径：　$(v,\ v_1,\ v_2,\ \ldots,\ u)$

从源点到终点可能不止一条路径，把路径长度最短的那条路径称为最短路径。

最短路径问题

示例：



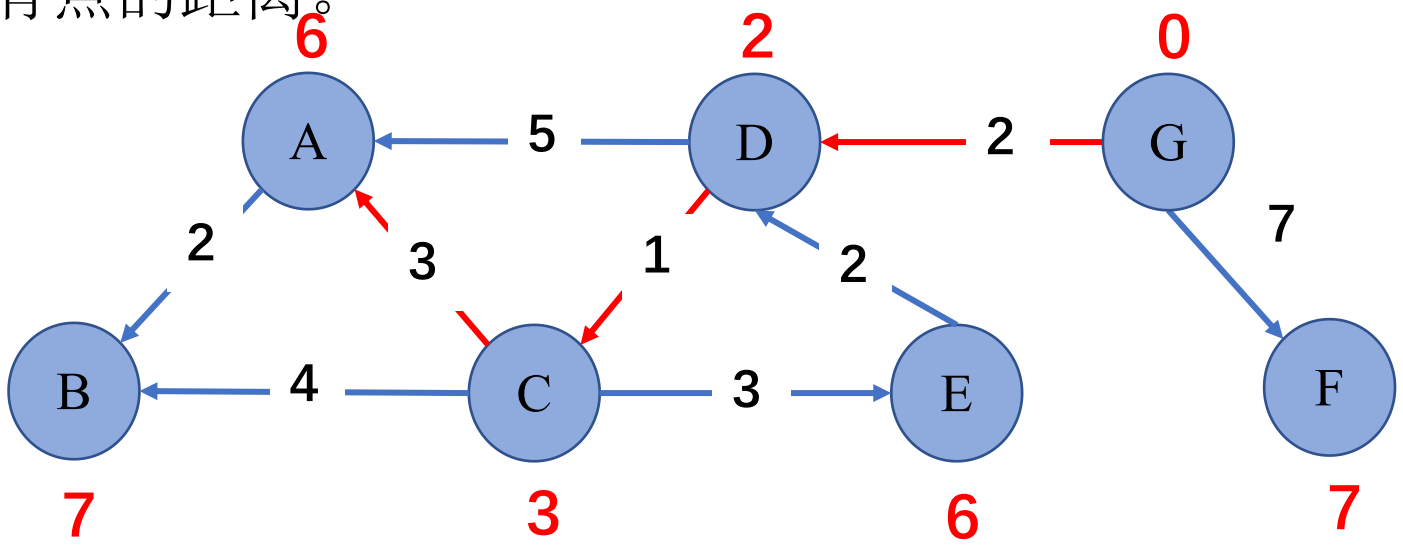
$\delta(G, A)=W(p)=2+1+3=6$

$\delta(A, G)=\infty$

最短路径问题

单源最短路径问题：

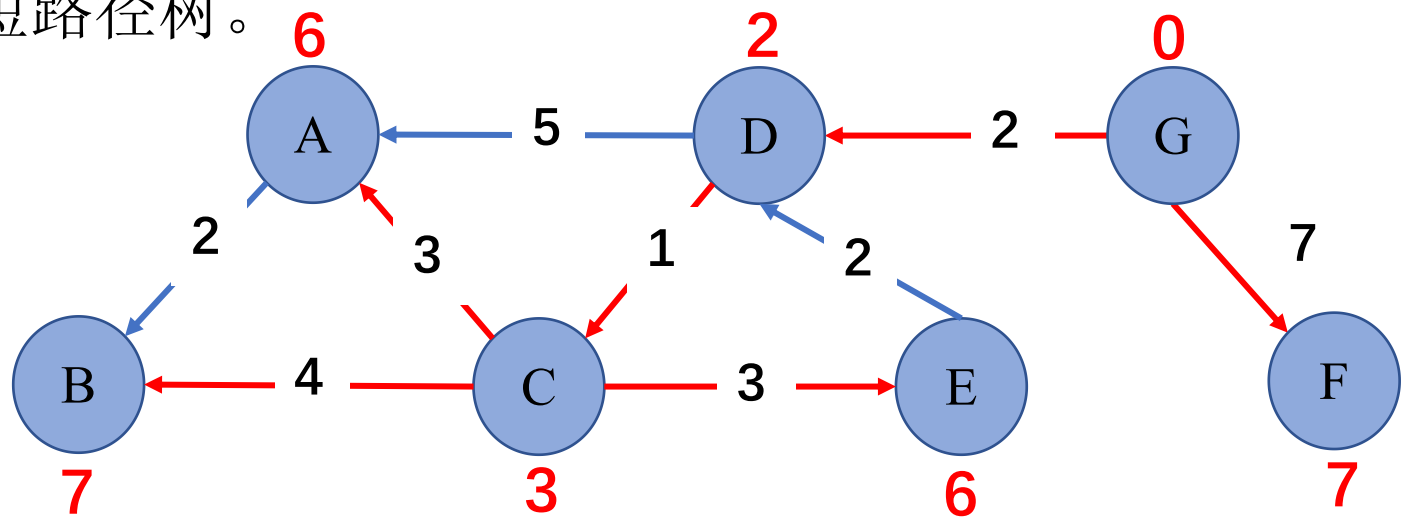
给定一个带权图G=(V, E, W)和一个源点s，计算s到其他V中所有点的距离。



最短路径问题

最短路径树：

由源点s到所有点的最短路径合起来就形成一棵树，称为s的最短路径树。



为了得到最短路径树，计算任一点v在树上的父节点，记为p[v]。

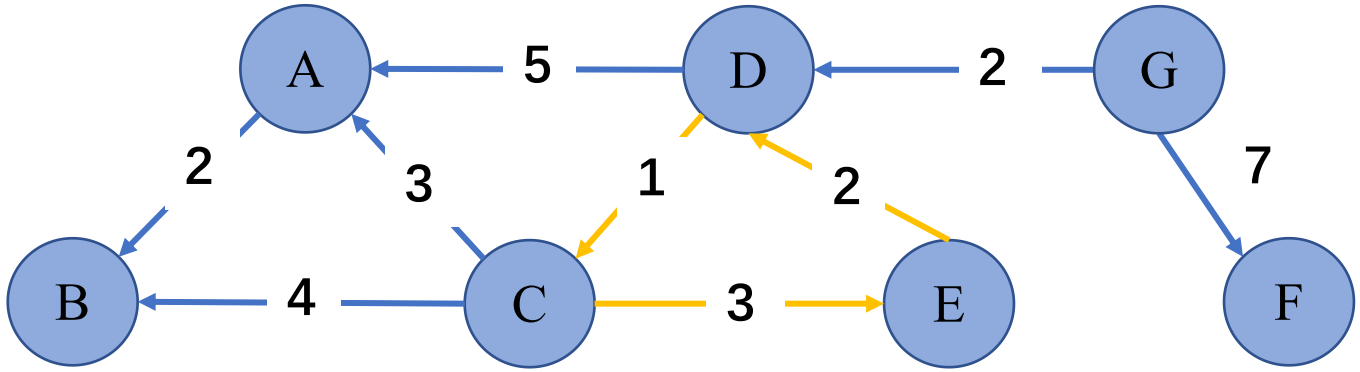
湖南大学 HUNAN UNIVERSITY

目录

松弛算法

求解最短路径的蛮力法：

为了计算s到t的最短路径，枚举出s到t的所有路径并计算最小值，如果有环，路径数量是无穷多。



松弛算法

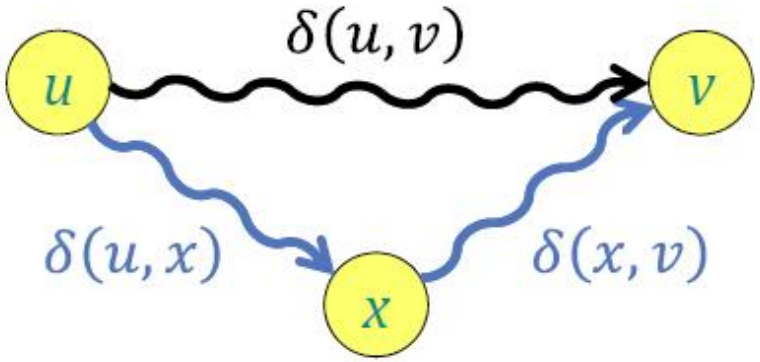
求解最短路径的蛮力法：

为了计算s到t的最短路径，枚举出s到t的所有简单路径并计算最小值，路径数量是指数级别的O(|V|!)。

松弛算法

三角不等式：

对于图上任意三个点u、v和x而言，δ(u,v)≤δ(u,x)+δ(x,v)。



推论：对于图上任意三个点u、v和x且存在边(x,v)而言，

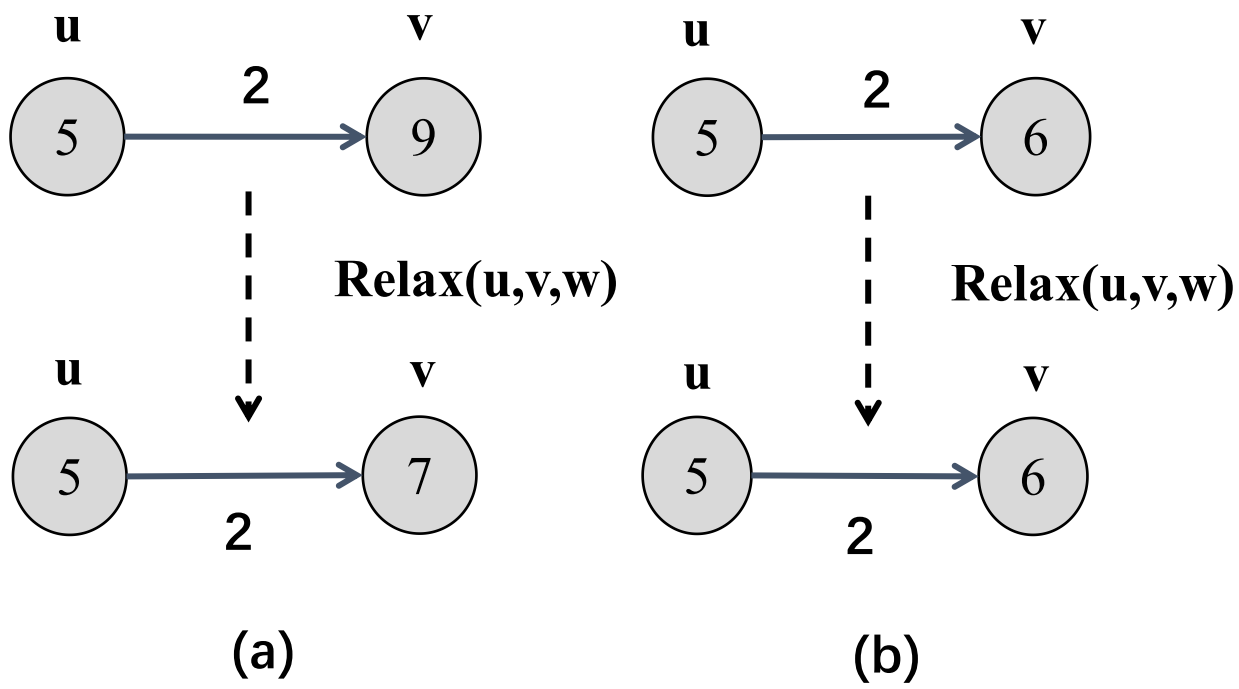
δ(u,v)≤δ(u,x)+w(x,v)。

松弛算法

对每个点v而言，维护一个属性d[v]，记录源点s到点d的距离上界，不断地降低对d[v]的估计，直到d[v]等于δ(s,v)。

- Relax(G, s)

```
1   for v in V:
2        d[v]= ∞;
3       p[v]=NULL;
4   d[s]=0; p[s]=s;
5    while some e (u, v) has
d[v]>d[u]+w(u,v):
6        pick such an e (u, v)
7       d[v]=d[u]+w(u,v);
8       p[v]=u;
```



松弛算法

收敛性质：

对于某些节点u, v ∈V，如果s -> u-> v是图G中的一条最短路径，并且对边(u, v)进行松弛前的任意时间有d[u] = δ(s, u)，则在对边(u, v)松弛之后的所有时间d[v] =δ(s, v)

松弛算法

路径松弛性质：

给定带权图G=(V, E, W)，设从源点s到点v_k的最短路径为$se_1v_1e_2 \dots e_kv_k$，如果对边$e_1, e_2, \dots, e_k$按次序进行松弛操作之后，$d[v_k]=\delta(s, v_k)$，且该性质的成立与其他边的松弛操作以及次序无关。

证明：数学归纳法，对于路径边数进行归纳

基础步：边数为0的最短路径就是源点，初始化阶段就有$d[s]=\delta(s, s)=0$；

归纳步：边数为n的最短路径都成立，边数为n+1最短路径$se_1v_1e_2 \dots e_nv_n e_{n+1}v_{n+1}$有两部分组成：边数为n的最短路径和只有一条边$e_{n+1}$的一条最短路径。

边数为n的最短路径按归纳假设有$d[v_n]=\delta(s, v_n)$，于是按照松弛操作有$d[v_{n+1}]=d[v_n]+w(e_{n+1})=\delta(s, v_{n+1})$

松弛算法

松弛算法正确性（上界性质）：

定理：在松弛算法中，对于V中任意的v，d[v]一直大于等于$\delta(s,v)$（最短路径），且一旦d[v]到达$\delta(s,v)$之后将不再发生变化。

证明：数学归纳法，针对调用松弛操作的次数

基础步：当松弛操作次数为0的时候，即初始化阶段，$dist[s]=0\geq\delta(s,s)=0$，而其它点v的d[v]都是∞都是大于等于$\delta(s,v)$。

归纳步：当松弛操作次数小于等于n时候，设第n+1次松弛操作加入了边(u,v)。

由于d[u]是在前n中操作中得到的值，所以$\delta(s,u)\leq d[u]$，于是，由三角不等式，$\delta(s,v)\leq\delta(s,u)+w(u,v)\leq d[u]+w(u,v)$，而$d[u]+w(u,v)=d[v]$就是第n+1次松弛操作的结果。

松弛算法

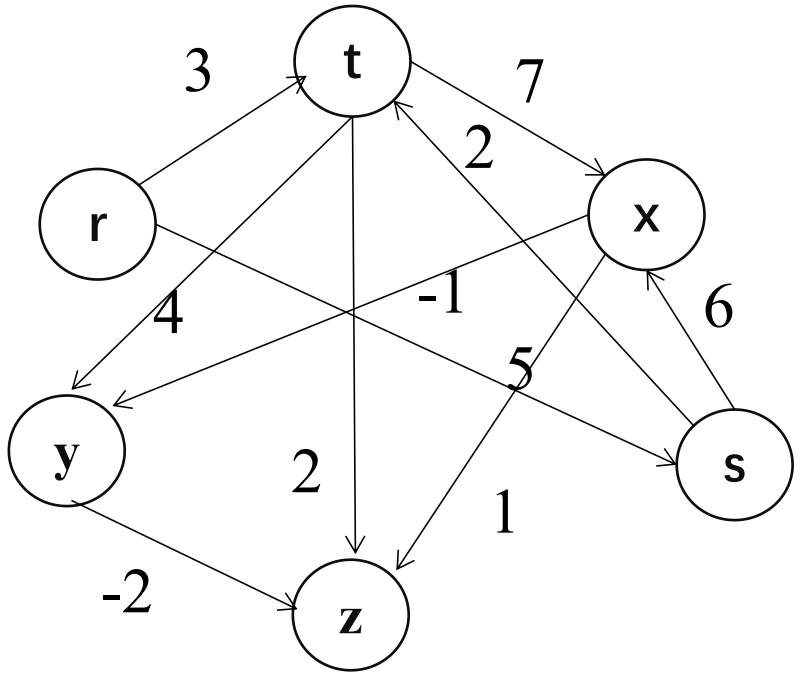
前驱子图性质：

对于所有的节点$v \in V$，一旦$d[v] = \delta(s, v)$，则前驱子图是一颗根节点为s的最短路径树。

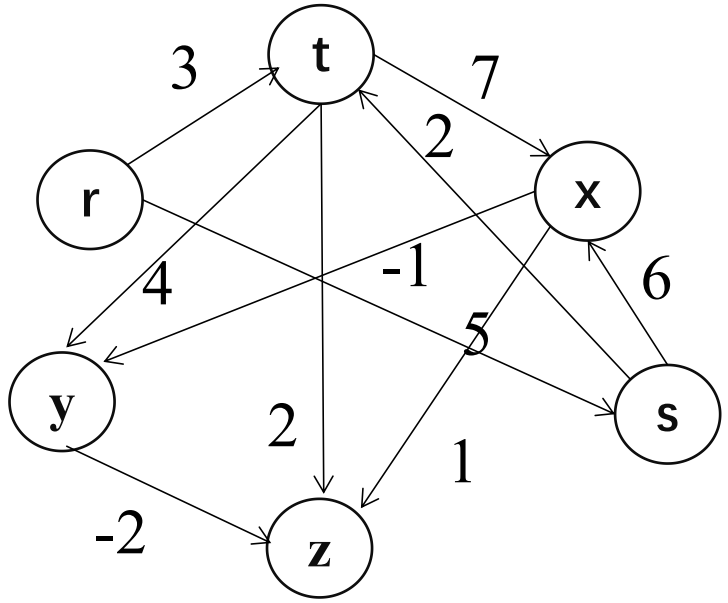
目录

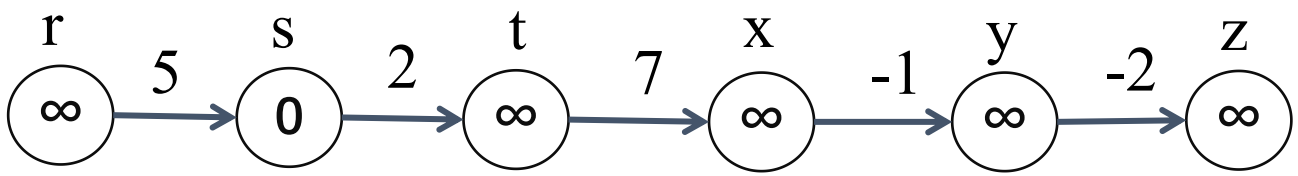
有向无环图上的最短路径计算

- 首先，对有向无环图中所有点进行拓扑排序；
- 然后，按照拓扑排序的顺序对所有边进行操作松弛。
- 时间复杂度O(|V|+|E|)。

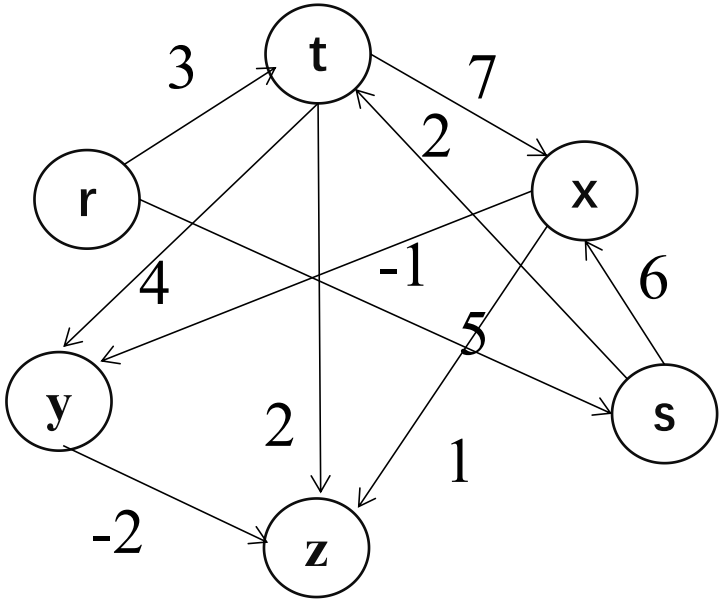


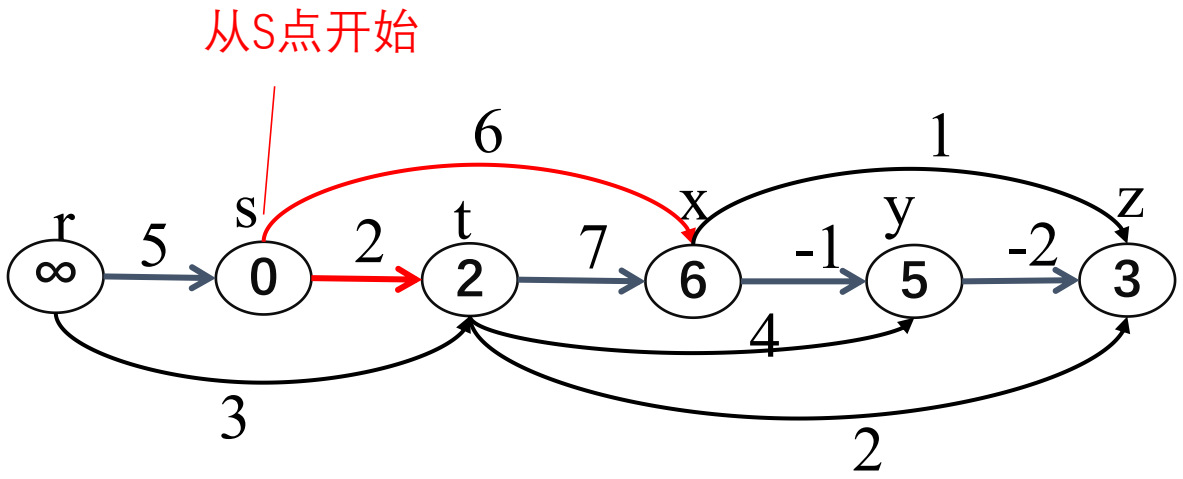
有向无环图上的最短路径计算





有向无环图上的最短路径计算

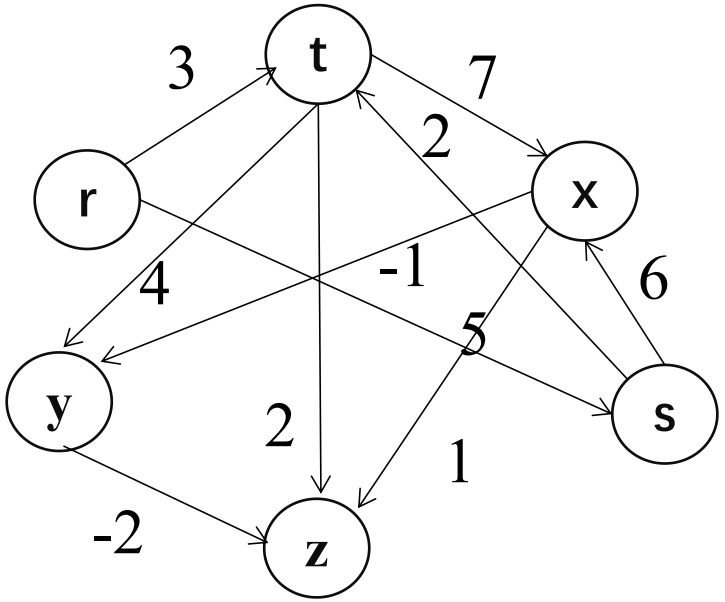




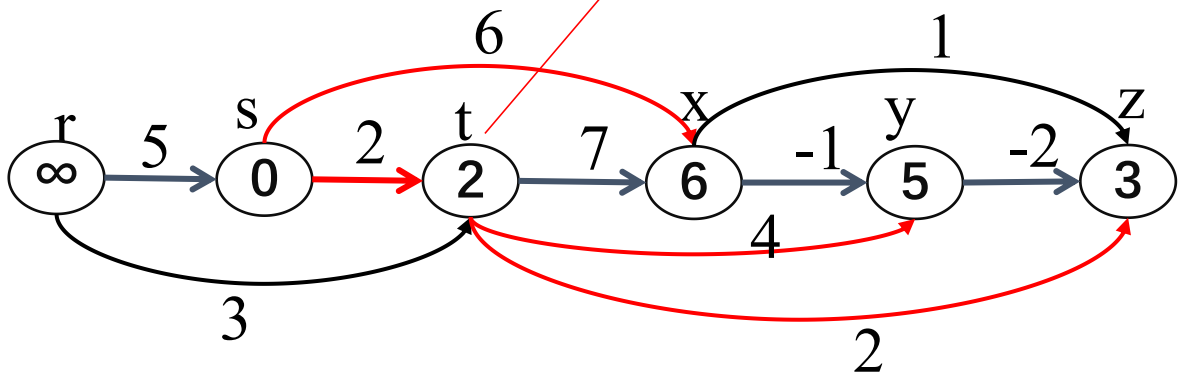

```
1 if(d[v]>d[u]+w(u,v))
2 then d[v]←d[u]+w(u,v)
3 π[v]←u    /*解释松弛操作*/
```

s到t的最短路径长度为2, 因此修改d[t]=2, s到x的目前最短路径为6，因此修改d[x]=6, 一次松弛操作完成，结果如上图所示。

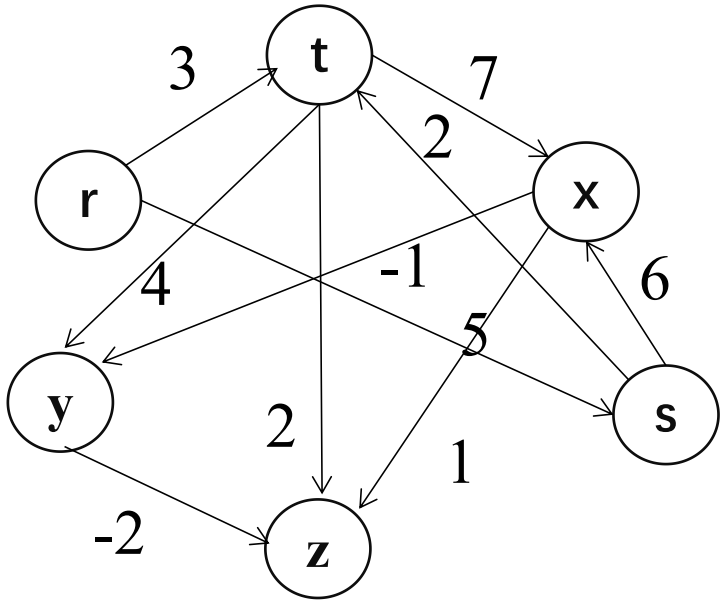
有向无环图上的最短路径计算

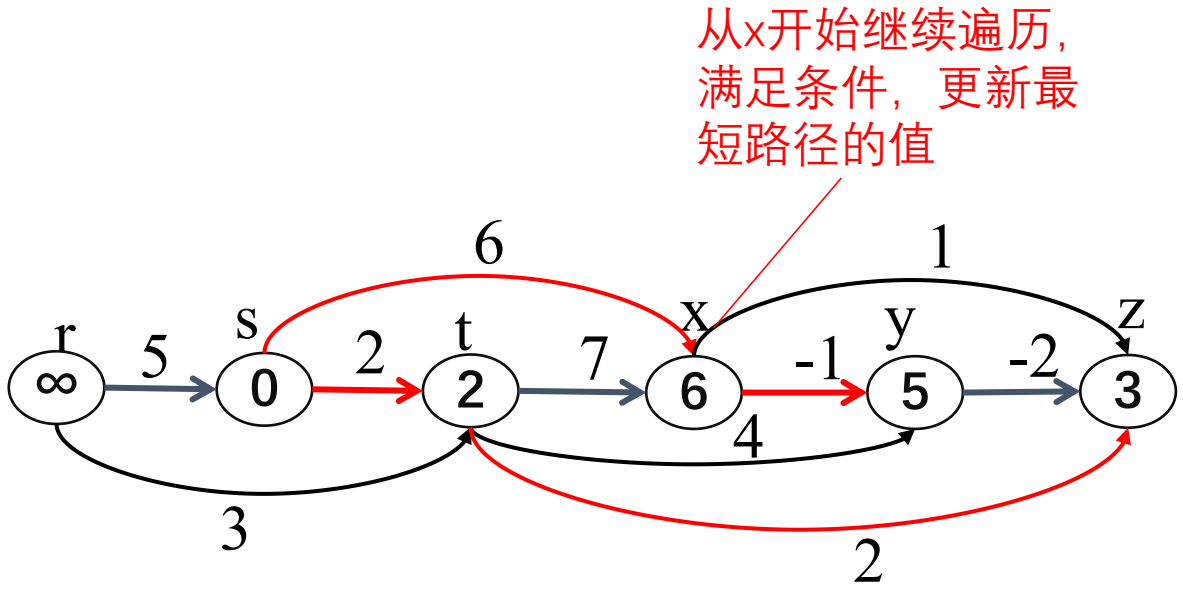


从t开始继续遍历，满足条件，更新最短路径

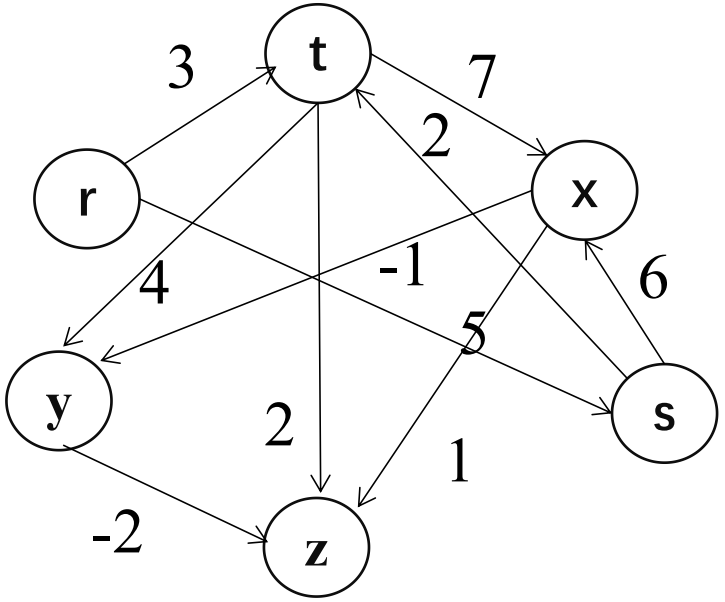


有向无环图上的最短路径计算

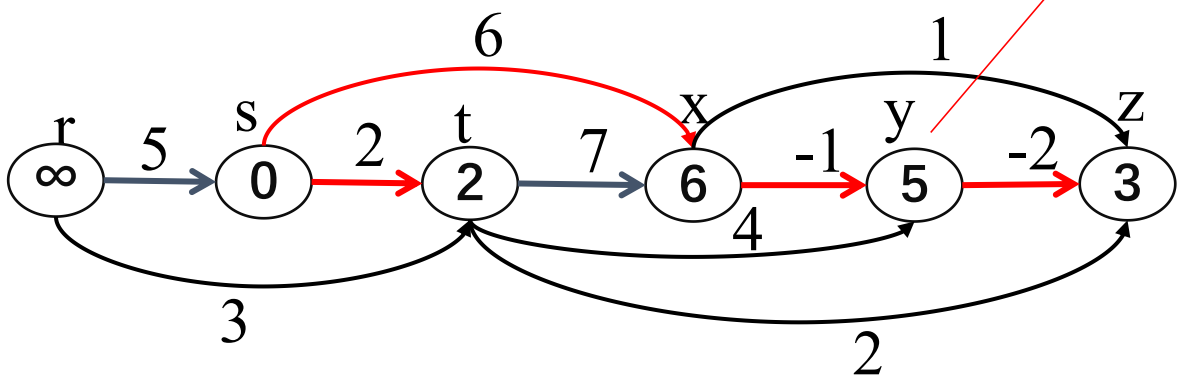




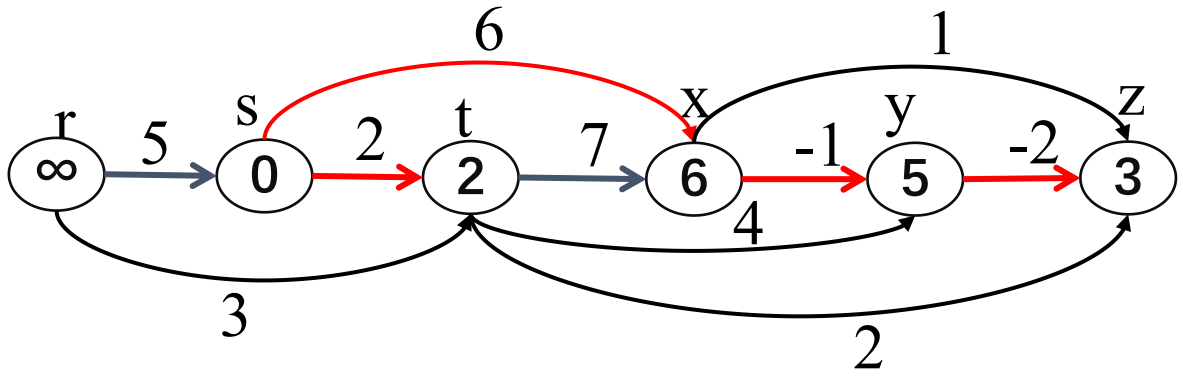
有向无环图上的最短路径计算



从y开始



最终结果



湖南大学 HUNAN UNIVERSITY

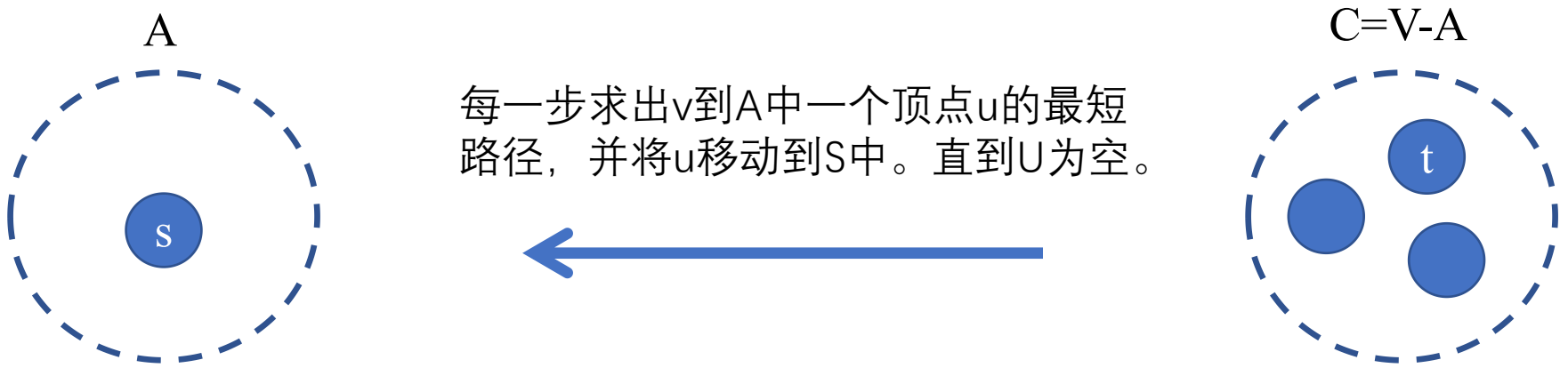
目录

Dijkstra算法

狄克斯特拉（Dijkstra）求解思路：

设G=(V，E)是一个带权有向图，把图中顶点集合V分成两组：

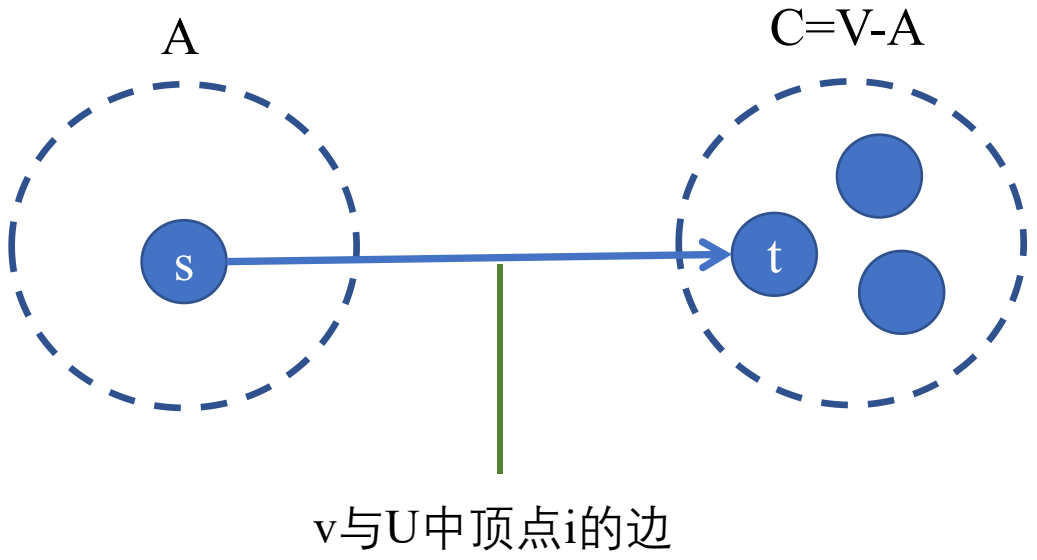
- 第1组为已求出最短路径的顶点集合（用A表示，初始时A中只有一个源点s，以后每求得一条最短路径s，…，t，就将t加入到集合A中，直到全部顶点都加入到A中，算法就结束了）。
- 第2组为其余未求出最短路径的顶点集合（用C表示）。



Dijkstra算法

Dijkstra算法的过程：

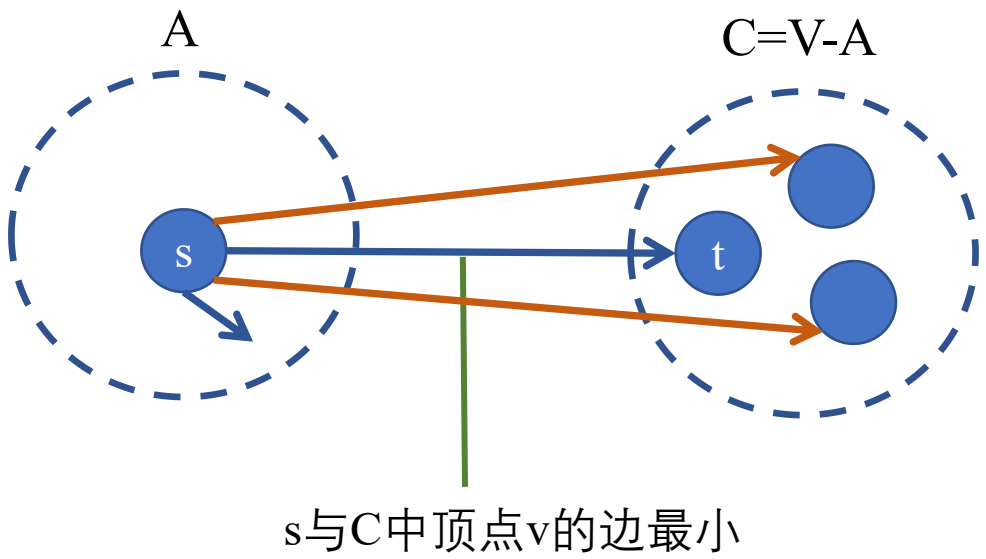
（1）初始化：A只包含源点即A={s}，s的最短路径为0。C包含除s外的其他顶点，C中顶点t距离为边上的权值（若s与t有边<s，t>）或∞（若t不是s的出边邻接点）。



Dijkstra算法

Dijkstra算法的过程：

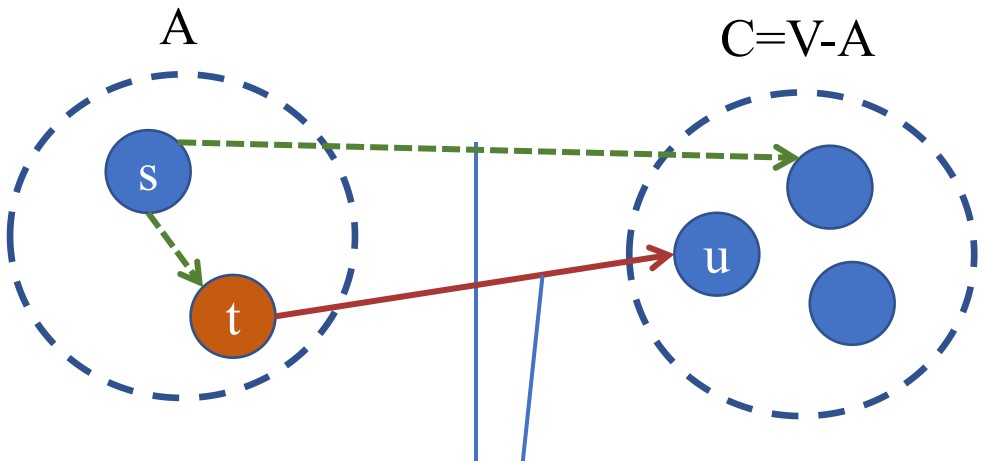
（2）从C中选取一个距离s最小的顶点t，把t加入A中（该选定的距离就是s ⇨ t的最短路径长度）。



Dijkstra算法

Dijkstra算法的过程：

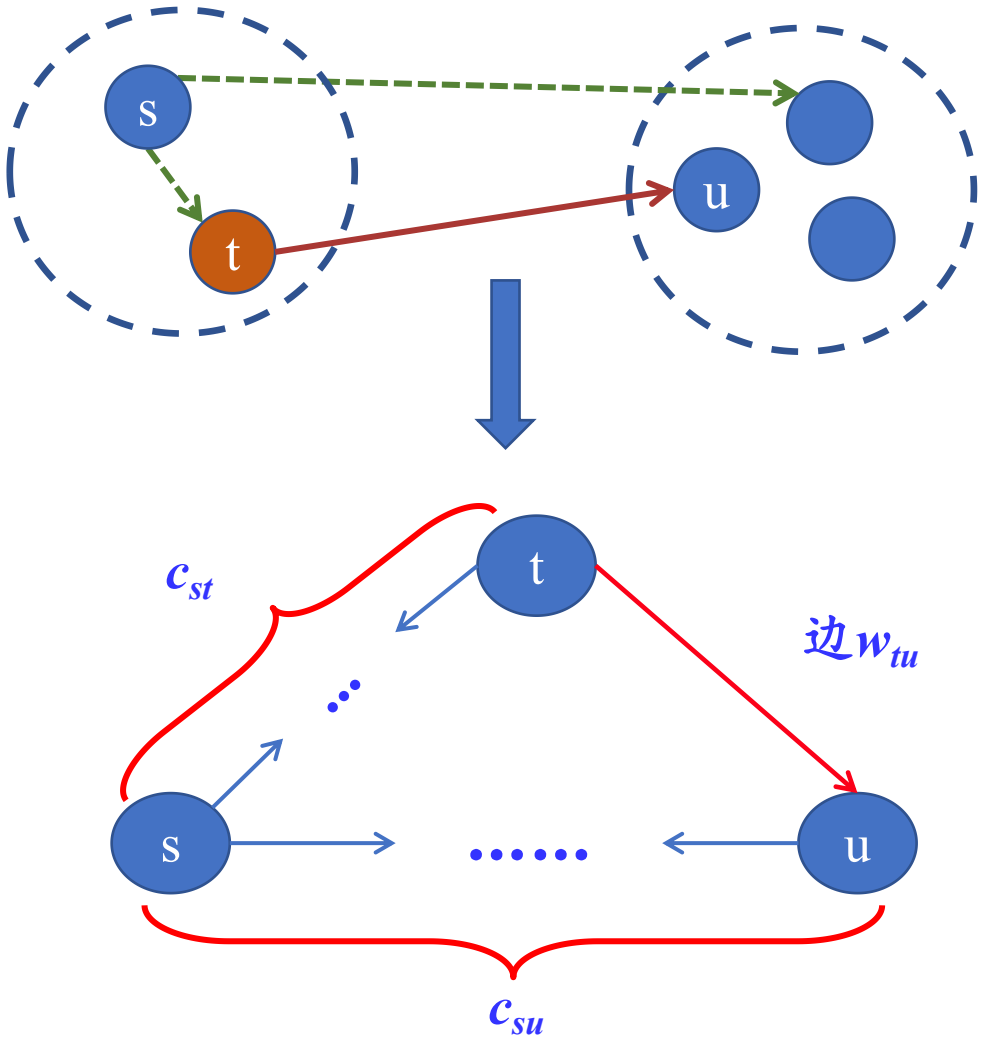
（3）以t为新考虑的中间点，修改U中各顶点u的最短路径长度：若从源点s到顶点u（u∈C）的最短路径长度（经过顶点t）比原来最短路径长度（不经过顶点t）短，则修改顶点u的最短路径长度。



两条路径进行比较：
若经过t的最短路径长度更短，则修正

Dijkstra算法

修改方式：



s ⇨ u的路径：

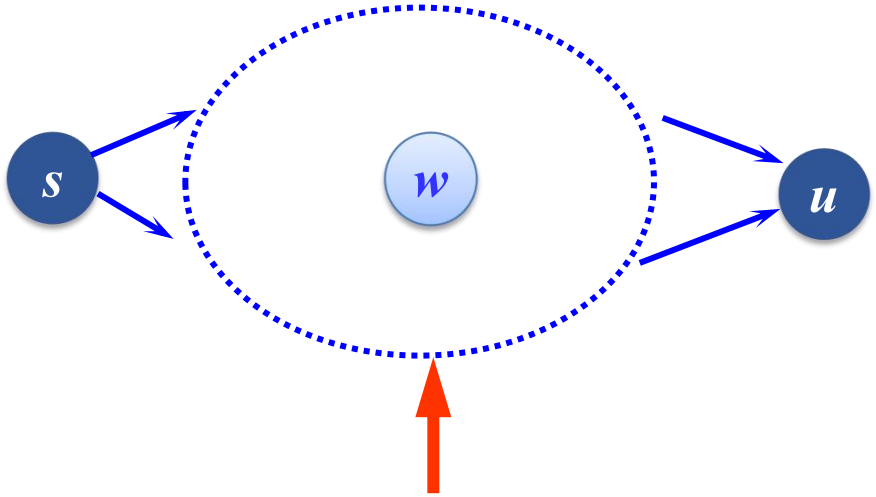
- 不经过顶点t
- 经过顶点t

顶点v ⇨ j的最短路径长度 = MIN(c_{sk}+w_{ku}，c_{tj})

Dijkstra算法

Dijkstra算法的过程：

（4）重复步骤（2）和（3）直到所有顶点都包含在S中。



考虑中间其他所有顶点w，通过比较得到s ⇨ u的最短路径

Dijkstra算法

算法设计（解决2个问题）

- 如何存放最短路径长度:

用一维数组d[j]存储!

源点v默认， d[j]表示源点 ⇨ 顶点j的最短路径长度。如d[2]=12表示源点 ⇨ 顶点2的最短路径长度为12。

- 如何存放最短路径（删掉）

从源点到其他顶点的最短路径有n-1条，一条最短路径用一个一维数组表示，如从顶点0 ⇨ 5的最短路径为0、2、3、5，表示为

p[5]={0,2,3,5}。

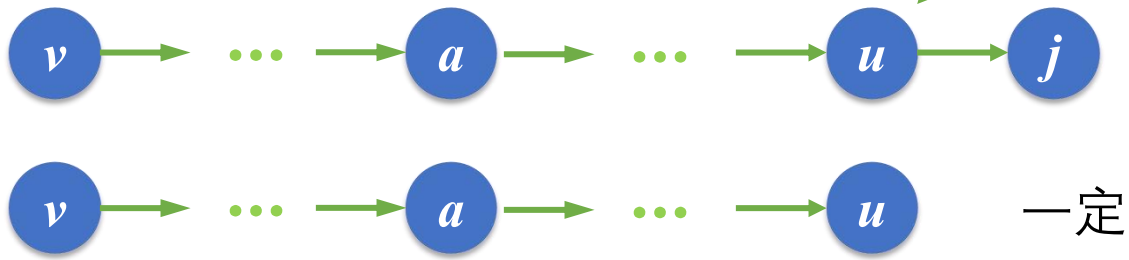
所有$n-1$条最短路径可以用二维数组p[][]存储。

Dijkstra算法

改进的方法是采用一维数组path来保存：

若从源点v ⇨ j的最短路径如下：

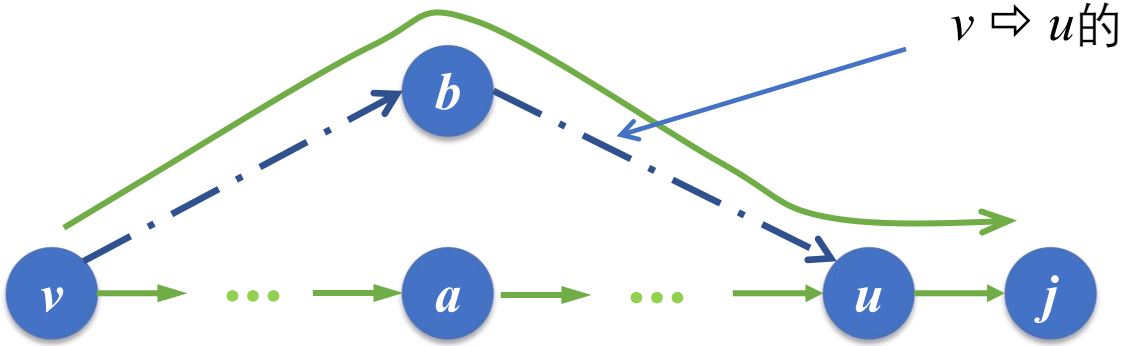
v ⇨ j最短路径中j的前一个顶点



则：

一定是从源点v ⇨ u的最短路径

v ⇨ u的最短路径

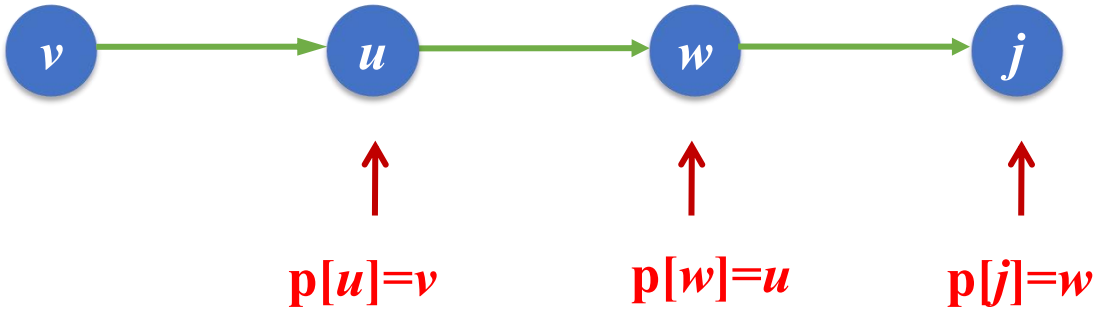


而通过b的路径更短，则$v \rightarrow \cdots a \rightarrow \cdots \quad u \rightarrow j$不是最短路径

与假设矛盾，问题得到证明。

Dijkstra算法

v ⇨ j的最短路径:

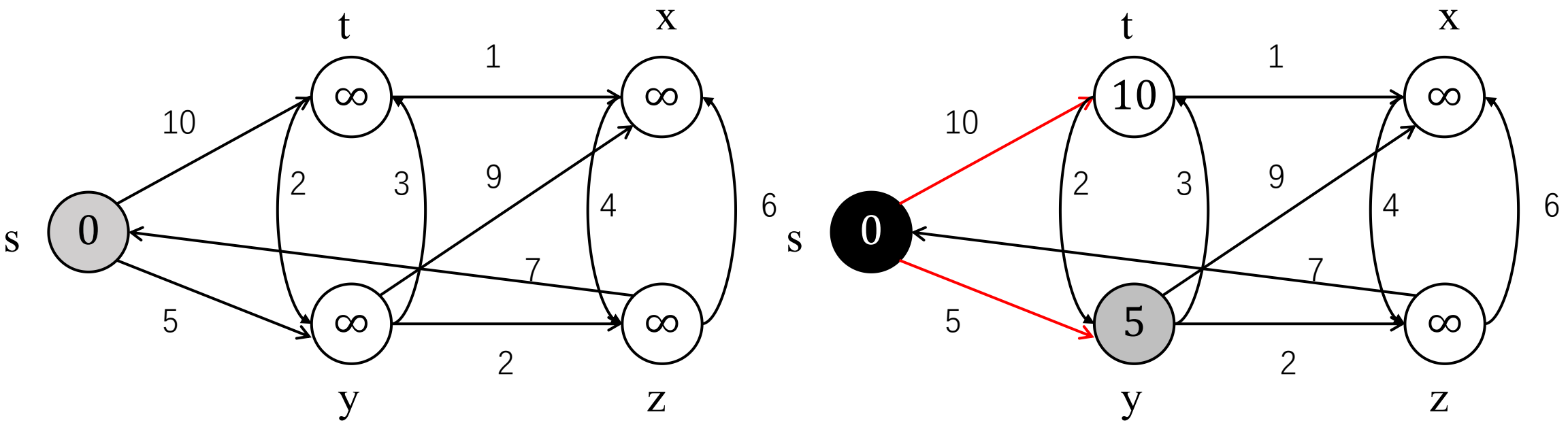


从p[j]推出的逆路径: j，w，u，v

对应的最短路径为: $v \rightarrow u \rightarrow w \rightarrow j$

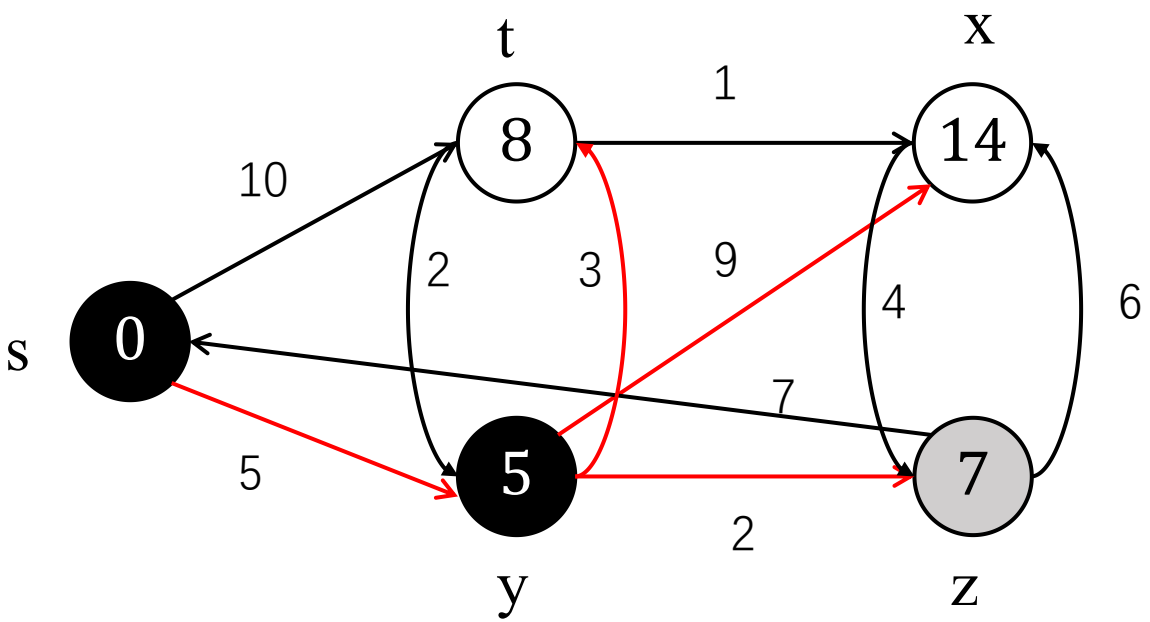
Dijkstra算法

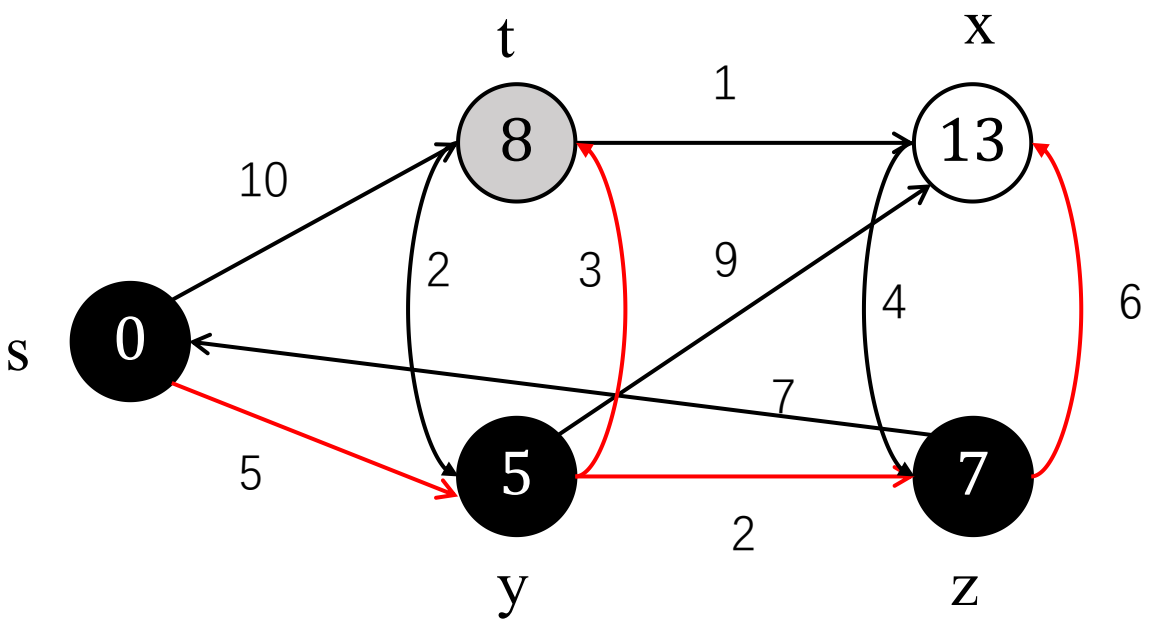
Dijkstra算法
示例演示



Dijkstra算法

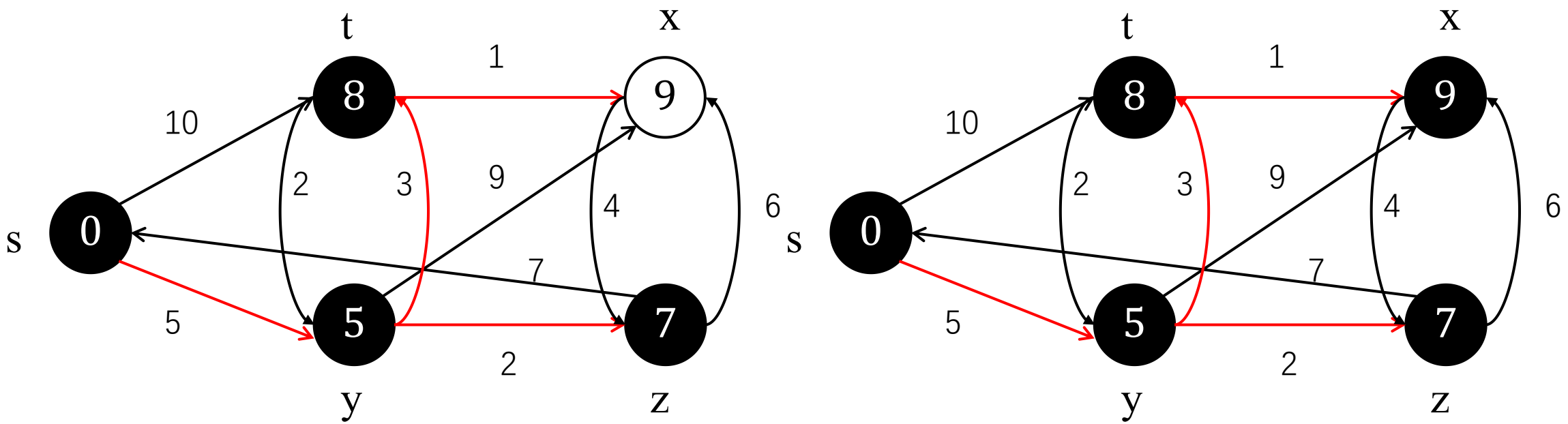
Dijkstra算法
示例演示





Dijkstra算法

Dijkstra算法
示例演示



4 Dijkstra算法

伪代码：

```
Dijkstra(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2   S = ∅
3   Q = G.V
4   while Q ≠ ∅
5         u = EXTRACT-MIN(Q)
6         S = S ∪ {u}
7         for each vertex v ∈ G.Adj[u]
8             RELAX(u,v,w)
```

4 Dijkstra算法

算法正确性：

给定带权图G=(V, E, W)，算法终止的时候d[v]=δ(s, v)

证明：反证法

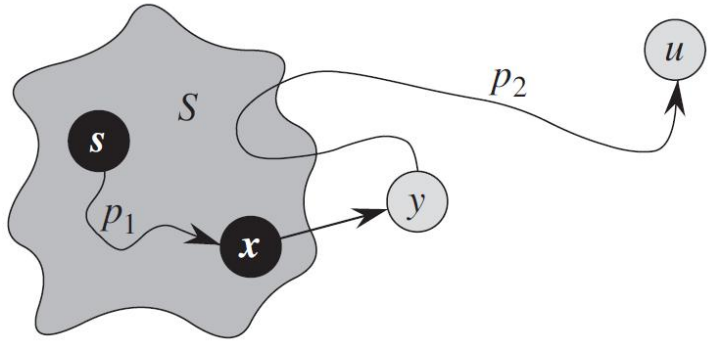
假设u是第一个出优先队列时$d[u] \neq \delta(s,u)$的点，

必然存在一条最短路径p是s到u，

x是p中u出优先队列时最后一个进S的点，

y是x在p中的后继，显然d[y]小于等于d[u]，

所以y肯定要在u之前出队列，与假设矛盾



4 Dijkstra算法

时间复杂度：

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

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(E + V \lg V)$ worst case

4 Dijkstra算法

思考题：

Dijkstra算法为什么不适合负权值的情况？

思考题：

Dijkstra算法可以用于带权无向图求最短路径吗？

思考题：

Dijkstra算法是否一定可以停止吗？

4 Bellman-Ford算法

Dijkstra算法的局限性：

- 如果边权为负值，Dijkstra算法还正确吗？
- 求解右图A 至其他点的最短距离
- 算法步骤：

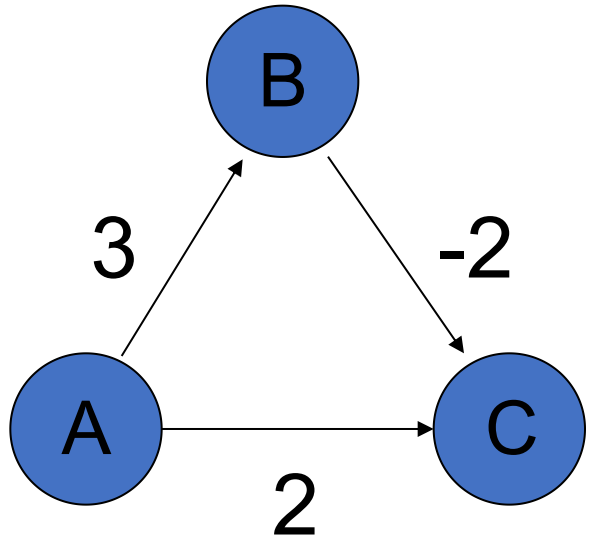
  1）标记点A

  2）d[C]=2最小，标记点C

  3）d[B]=3最小，标记点B

  结束

- 但是δ(A, C) =1



湖南大学 HUNAN UNIVERSITY

目录

5 Bellman-Ford算法

Dijkstra算法的局限性：

- 如果边权为负值，Dijkstra算法还正确吗？
- 求解右图A 至其他点的最短距离
- 算法步骤：

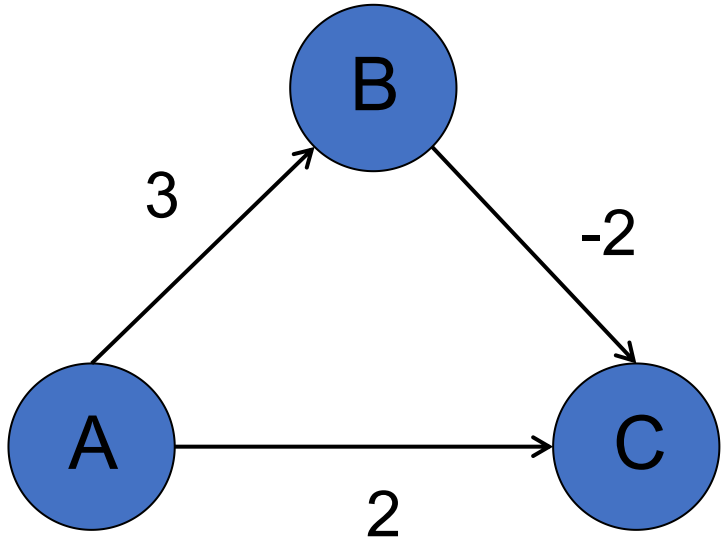
  1）标记点A

  2）d[C]=2最小，标记点C

  3）d[B]=3最小，标记点B

  结束

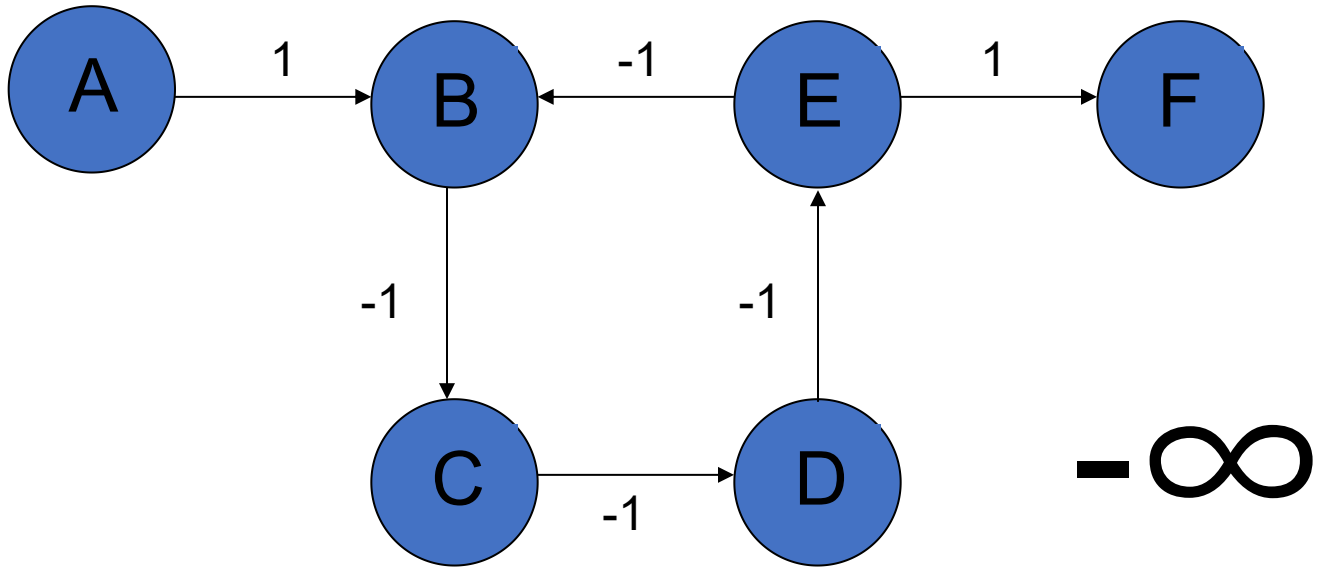
- 但是δ(A, C) =1



5 Bellman-Ford算法

Dijkstra算法的局限性：

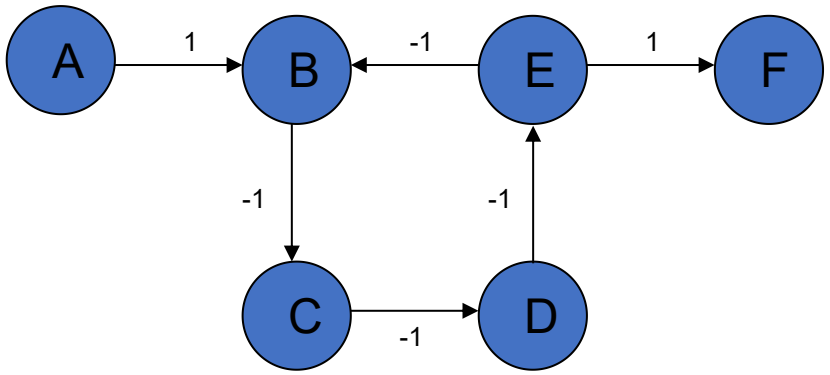
下图中，A至F的最短路径长度是多少？



5 Bellman-Ford算法

Dijkstra算法的局限性：

- 如果利用Dijkstra算法求解，结果为……
- 标记点A，d[B]=1，标记点B
- d[C]=0，标记点C
- d[D]=-1，标记点D
- d[E]=-2，标记点E
- d[F]=-1，标记点F
- 所求得的距离
  并不是最短的



5 Bellman-Ford算法

Dijkstra算法的局限性：错误结果的原因

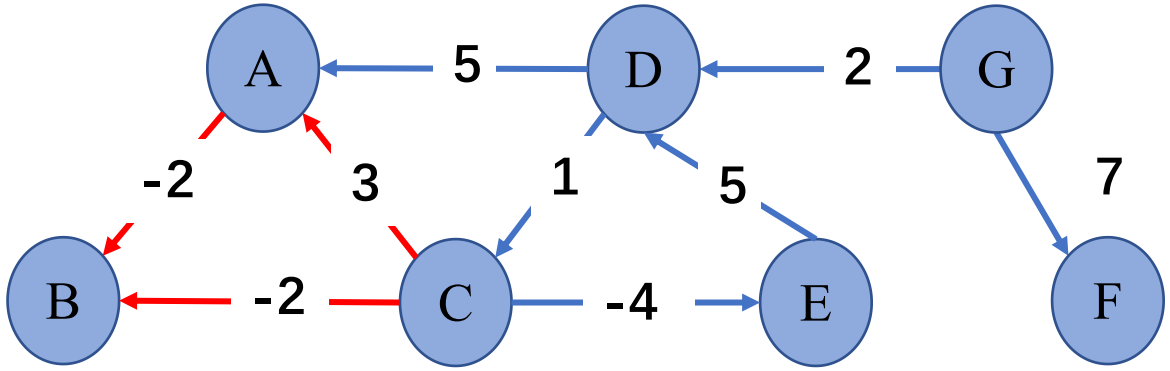
- Dijkstra的缺陷就在于它不能处理负权回路：Dijkstra对于标记过的点就不再进行更新了，所以即使有负权导致最短距离的改变也不会重新计算已经计算过的结果。
- 我们需要新的算法——Bellman-Ford

5 Bellman-Ford算法

负权环路：

给定一条路径p = $v_0e_1v_1e_2 \ldots e_lv_l$，如果若$v_0$和点$v_l$是同一个点，则p被称为一条环路。

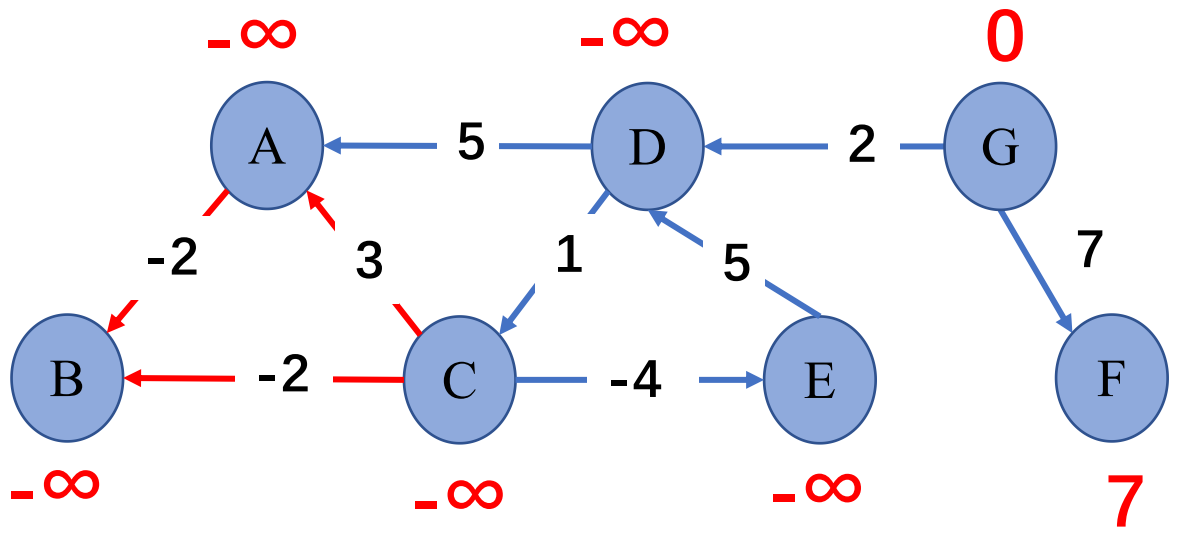
如果p中各条边权重之和$\sum_{k=1}^{k=l} w(e_k)$是负数，那么p称为负权环路。



5 Bellman-Ford算法

基于负权环路的距离定义：

当图中有负权环路的时候，部分点的最短路径就未定义了，记为-∞



5 Bellman-Ford算法

算法伪代码：

```
1  Bellman-Ford(G,w,s)
2    for v in V:
3      d[v]= ∞; p[v]=NULL;
4       d[s]=0;
5   for i from 1 to |V|-1 :
6       for each e (u, v) ∈E:
7           Relax(u,v,w)
8   for each e (u, v) ∈E:
9       if d[v] > d[u] + w(u, v)
10          return FALSE
11   return True
```

Bellman－Ford算法可以大致分为三个部分：

第一，初始化所有点。每一个点保存一个值，表示从原点到达这个点的距离，将原点的值设为0，其它的点的值设为无穷大（表示不可达）。

第二，进行循环，循环下标为从1到n－1（n等于图中点的个数）。在循环内部，遍历所有的边，进行松弛计算。

第三，遍历途中所有的边（e（u，v）），判断是否存在这样情况：

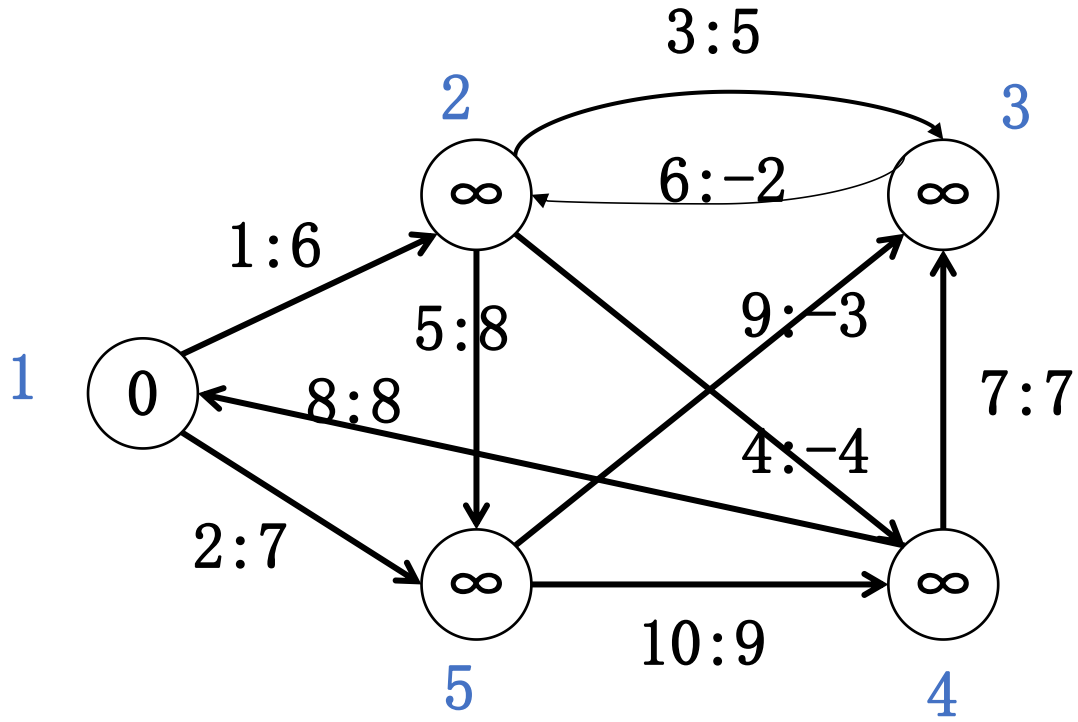
d[v] > d[u] + w(u,v)

则返回false，表示途中存在从源点可达的权为负的回路。

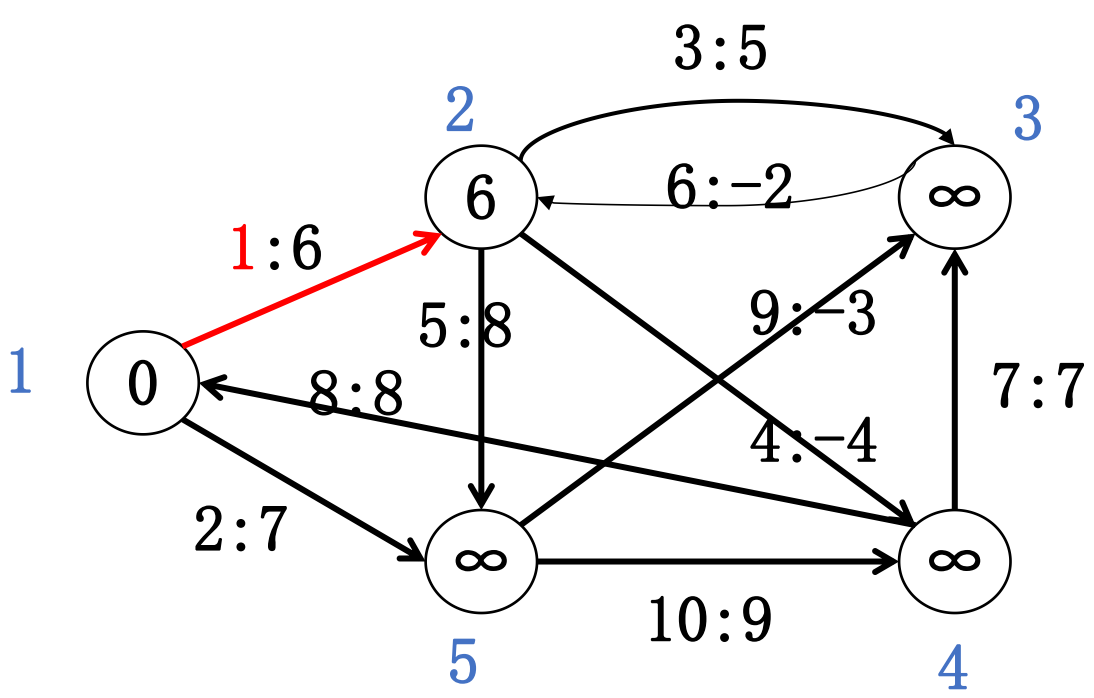
5 Bellman-Ford算法

算法伪代码：

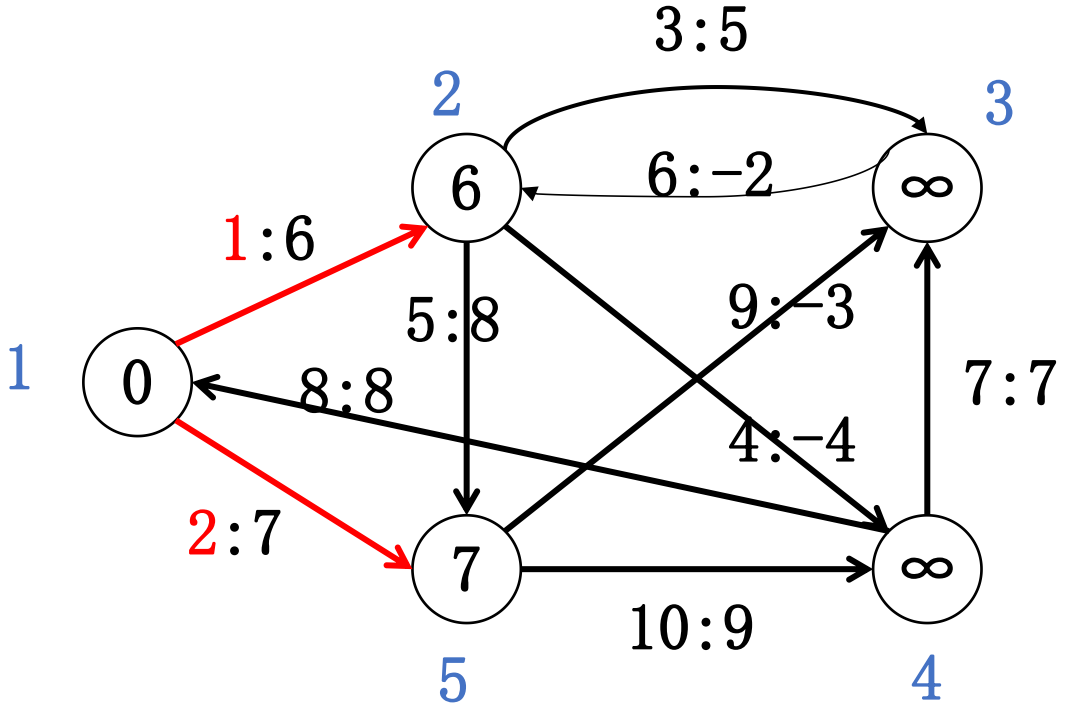
```
1  Bellman-Ford(G,w,s)
2    for v in V:
3      d[v]= ∞; p[v]=NULL;
4      d[s]=0;
5   for i from 1 to |V|-1 :
6      for each e (u, v) ∈E:
7          Relax(u,v,w)
8   for each e (u, v) ∈E:
9      if d[v] > d[u] + w(u, v)
10         return FALSE
11   return True
```



5 Bellman-Ford算法

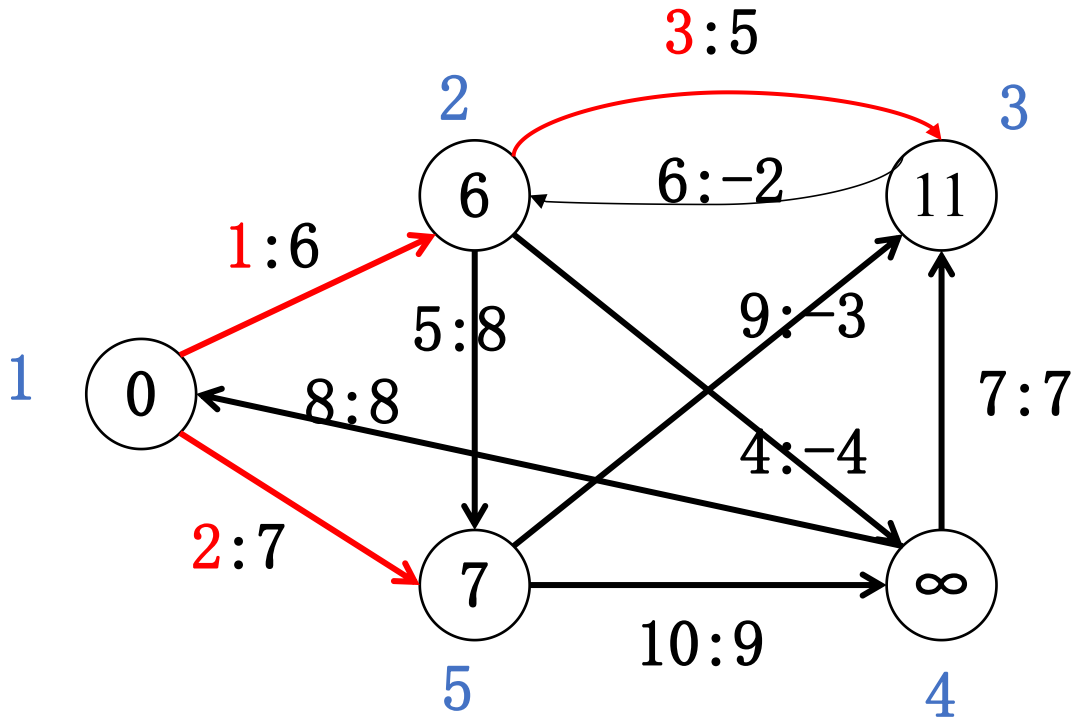


i=1 j=1

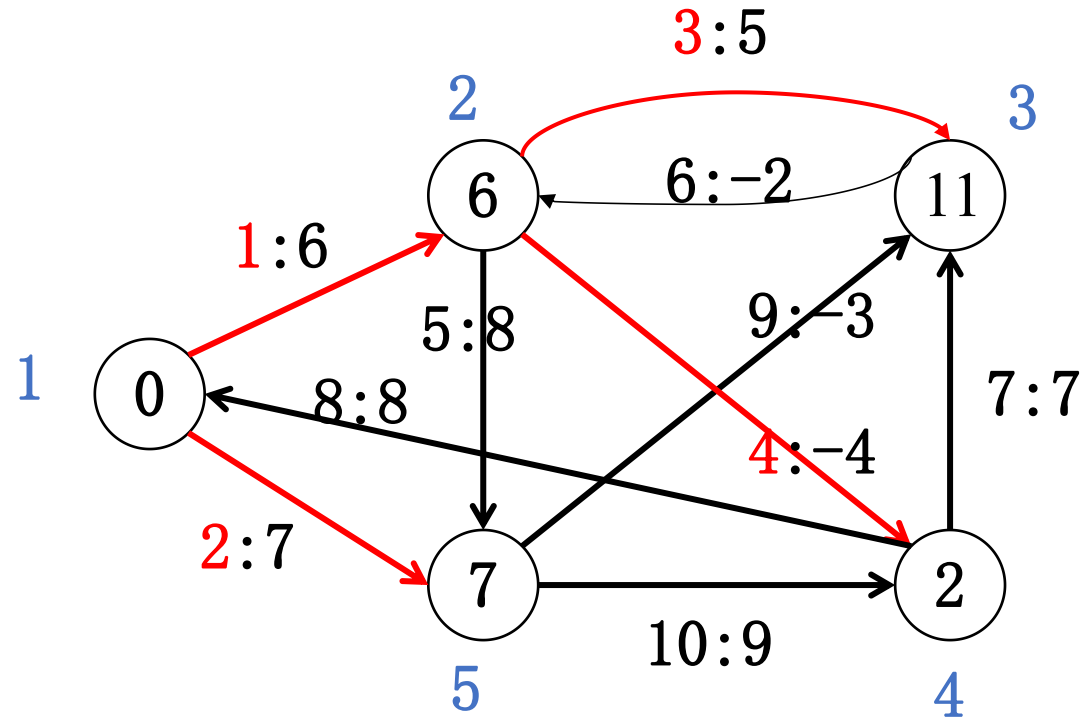


i=1 j=2

5 Bellman-Ford算法

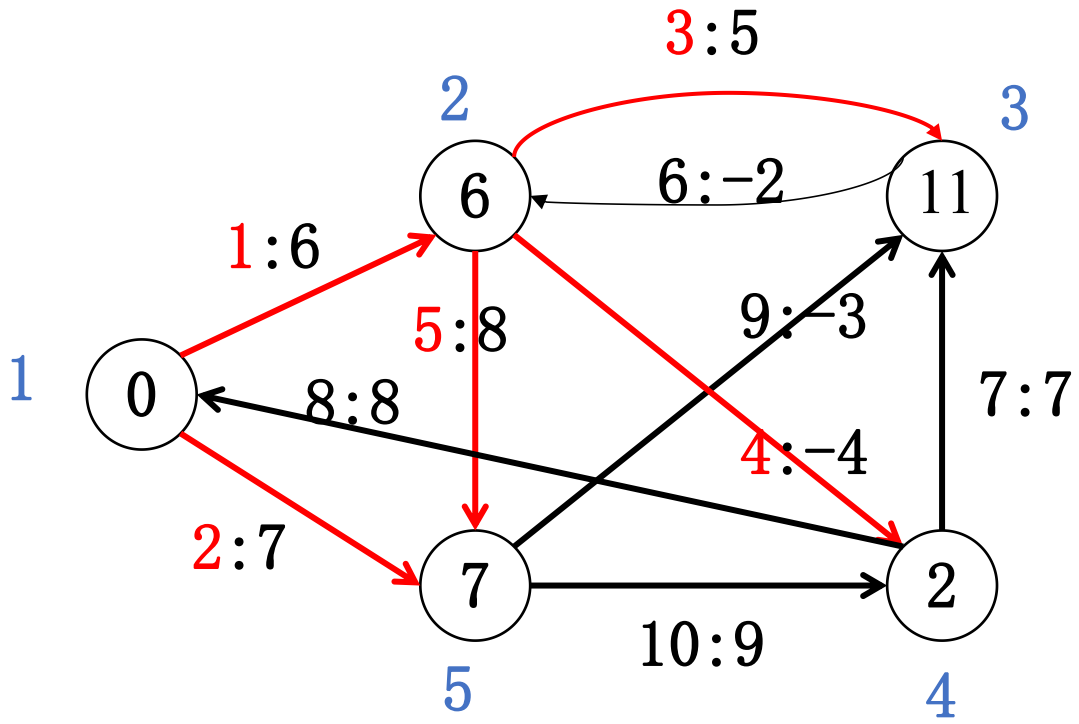


i=1 j=3

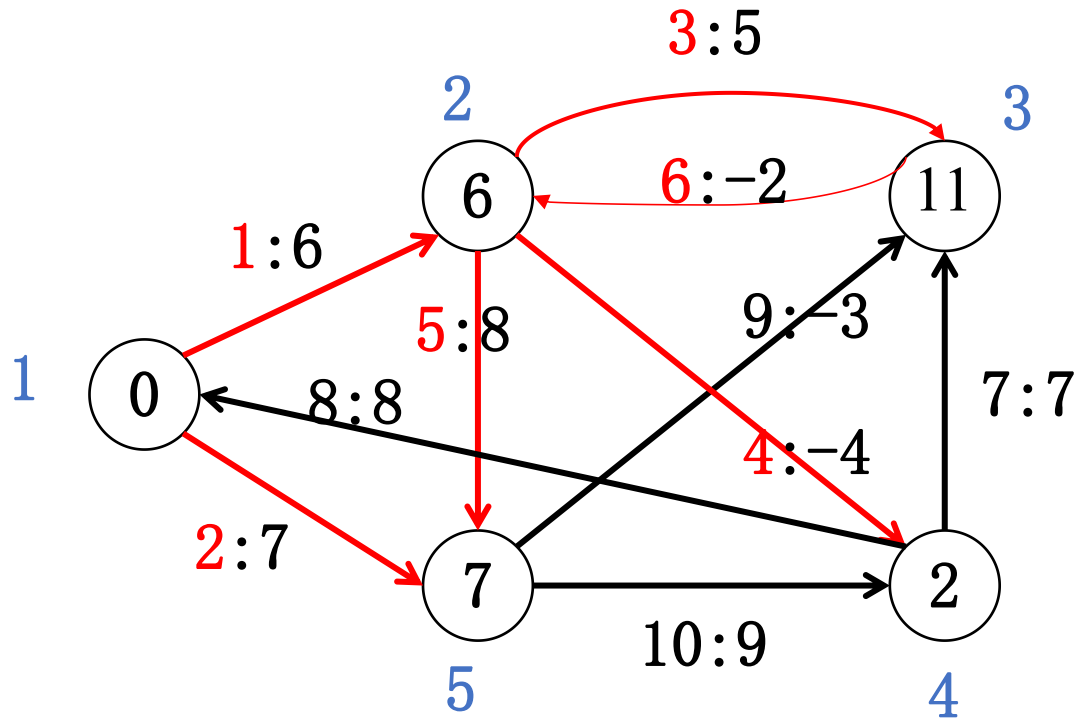


i=1 j=4

5 Bellman-Ford算法



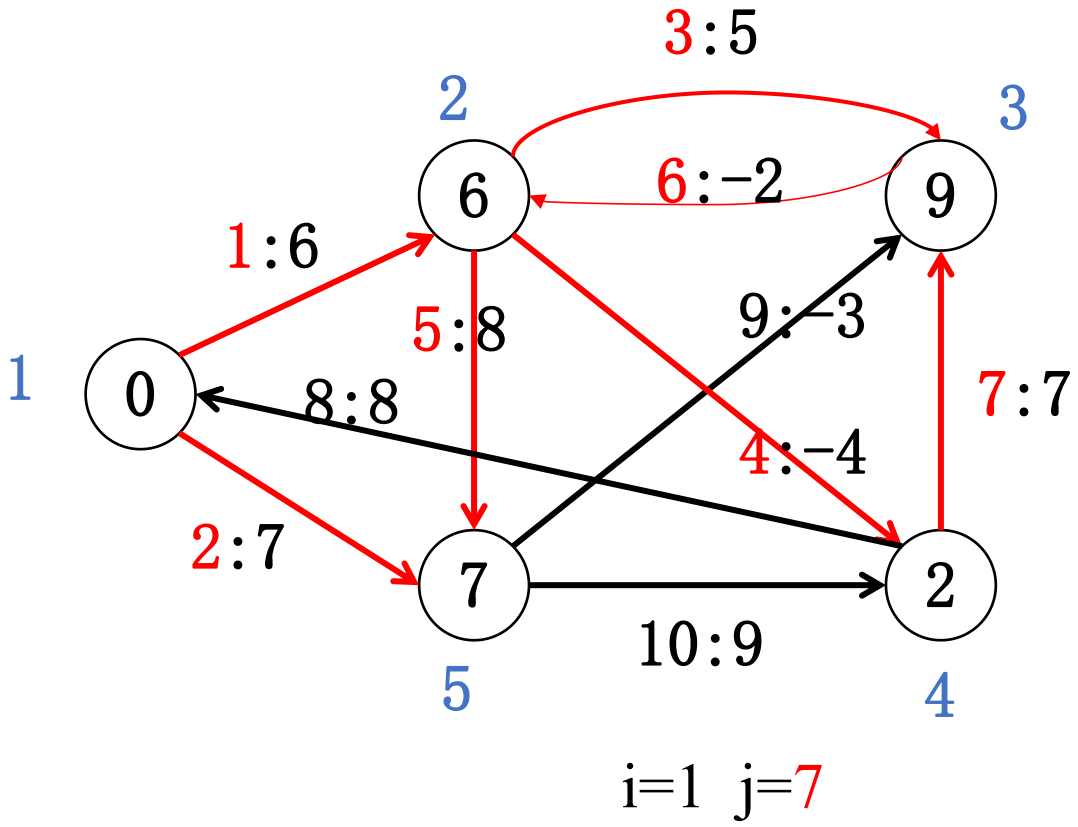
i=1 j=5



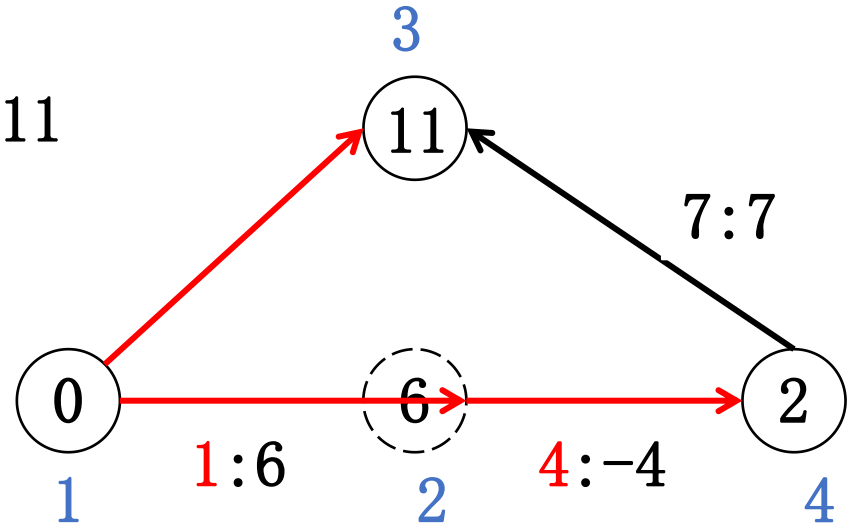
i=1 j=6

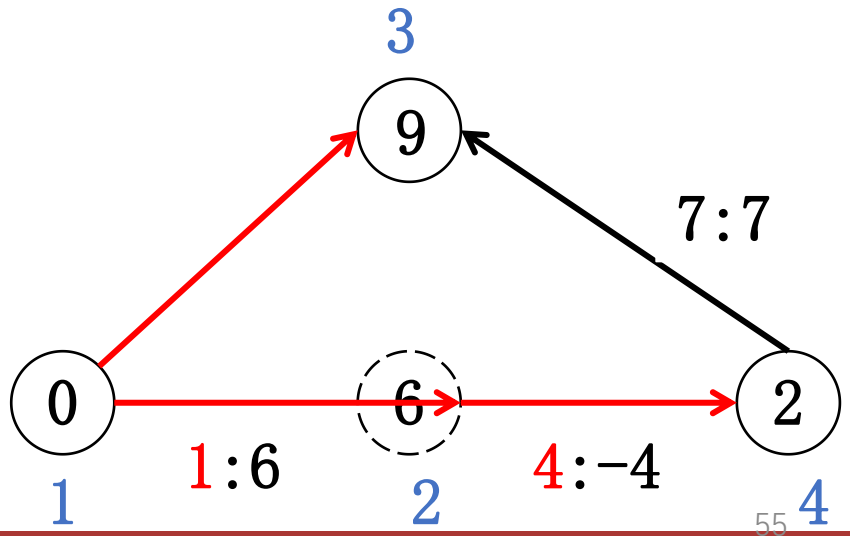
5 Bellman-Ford算法

1->3目前最短路径：11

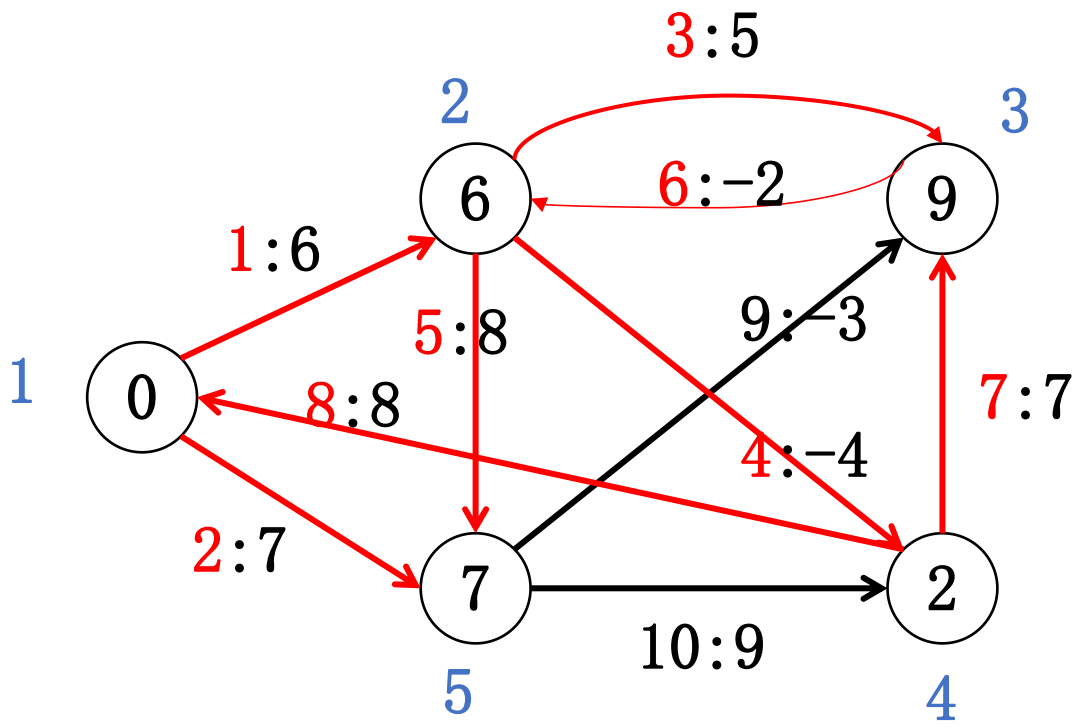


i=1 j=7

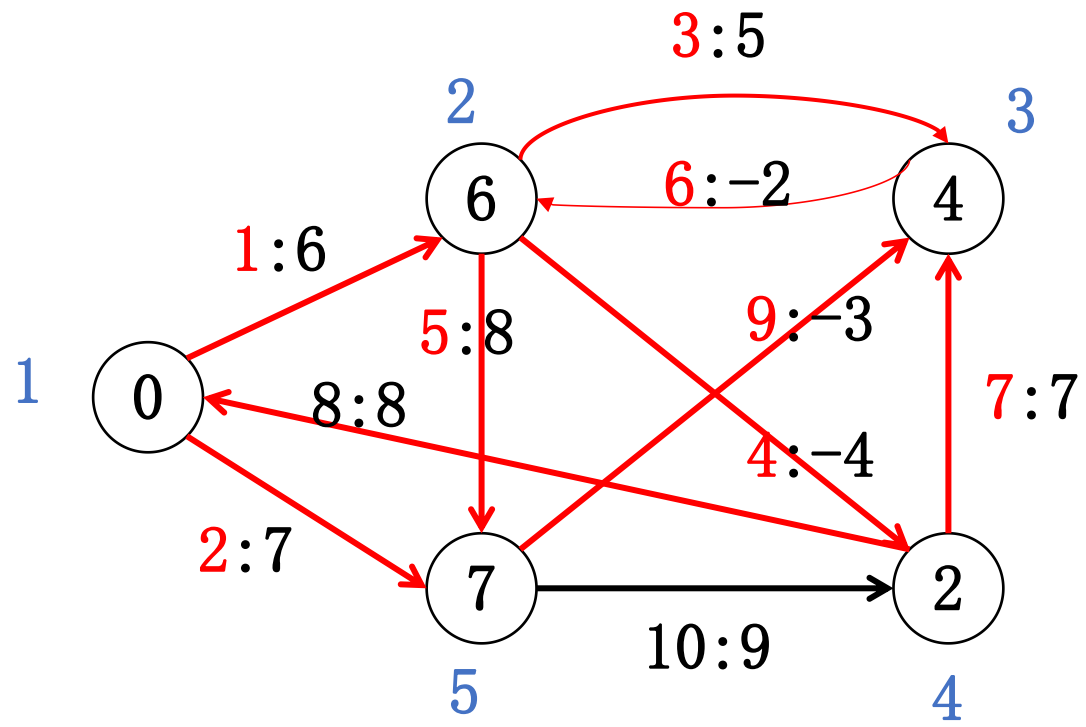




5 Bellman-Ford算法



i=1 j=8

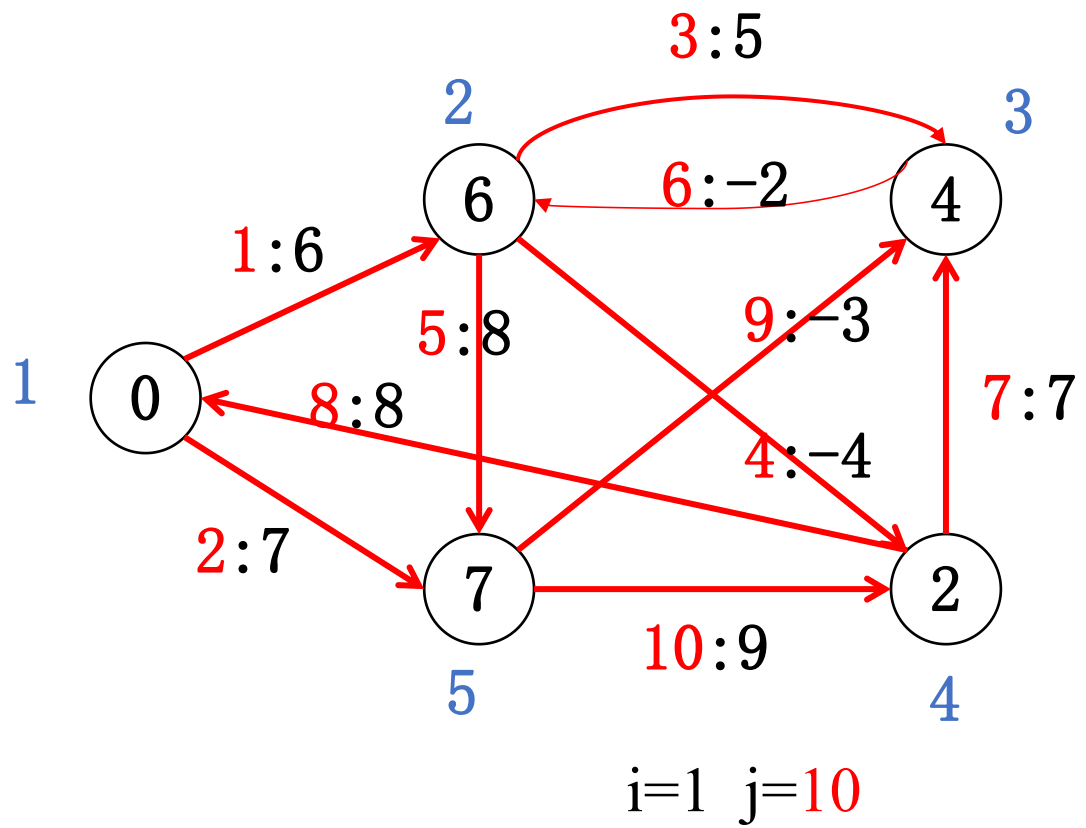


i=1 j=9

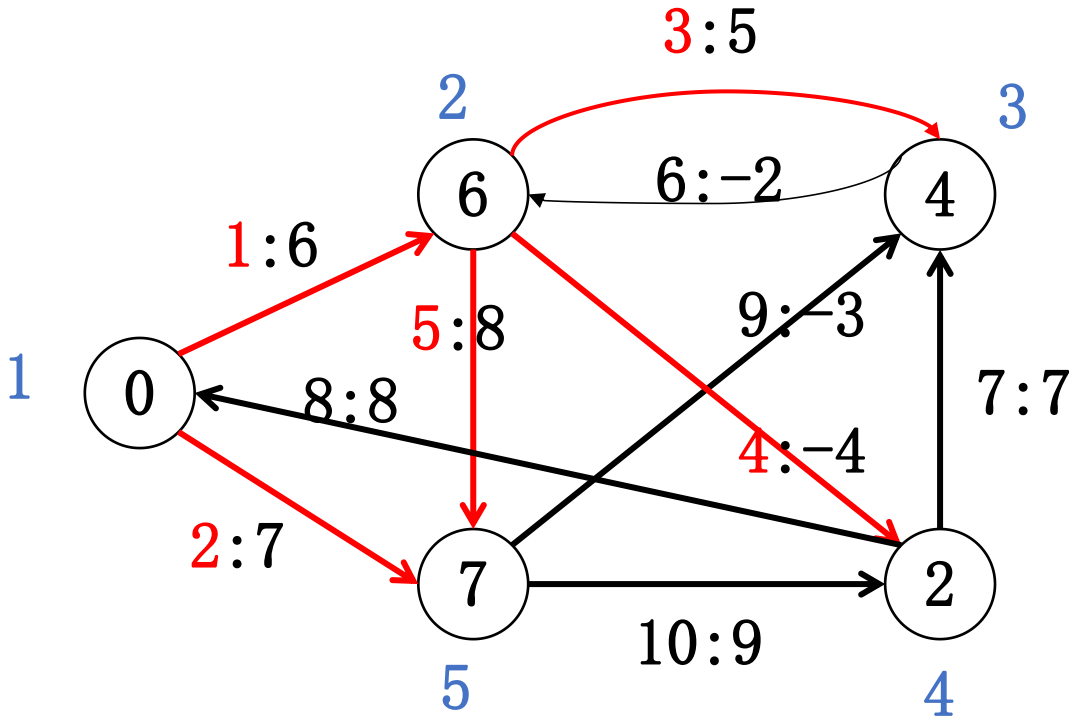
5 Bellman-Ford算法

算法伪代码：

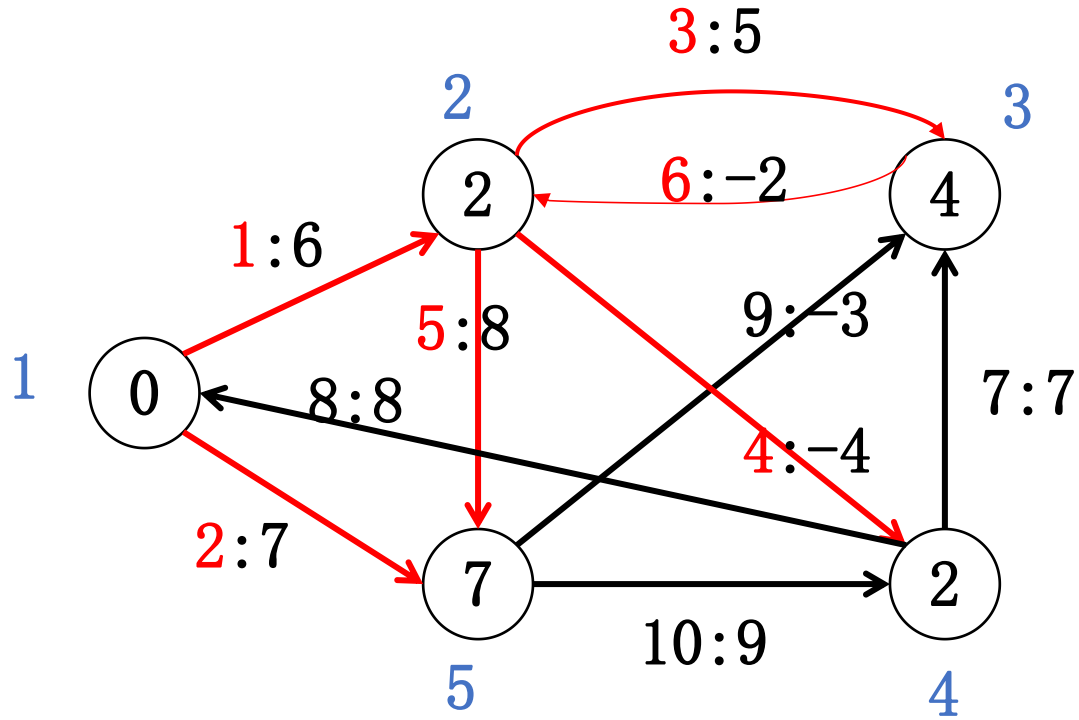
```
1  Bellman-Ford(G,w,s)
2     for v in V:
3        d[v]= ∞; p[v]=NULL;
4        d[s]=0;
5    for i from 1 to |V|-1 :
6        for each e (u, v) ∈E:
7            Relax(u,v,w)
8   for each e (u, v) ∈E:
9        if d[v] > d[u] + w(u, v)
10           return FALSE
11   return True
```



5 Bellman-Ford算法

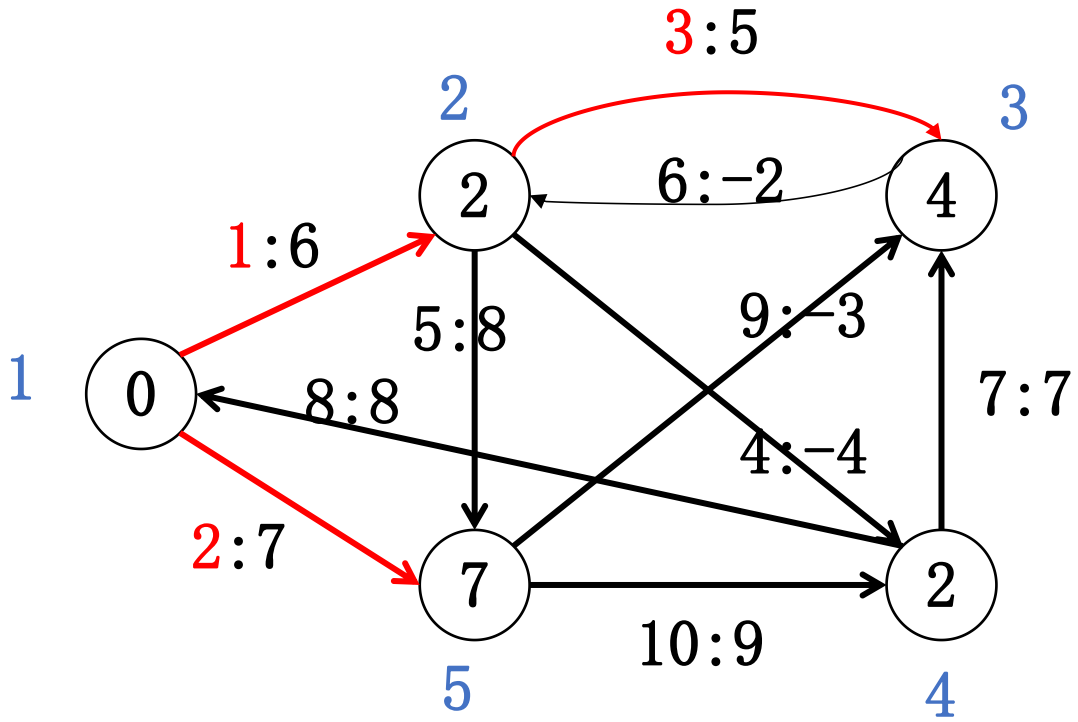


i=2 j=5

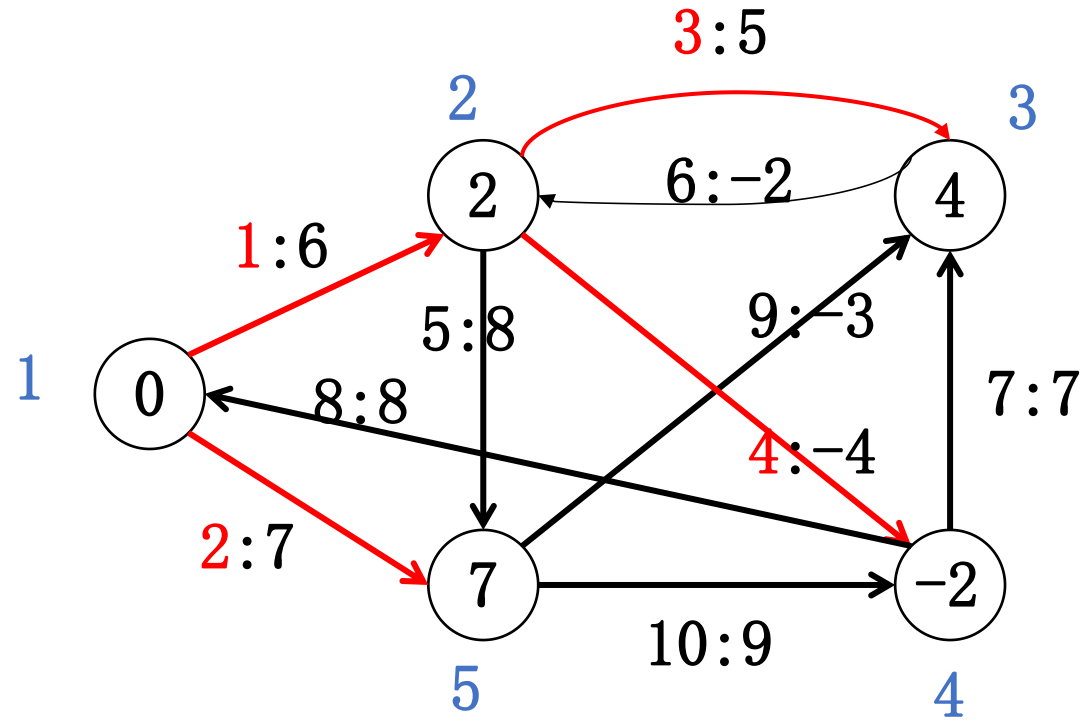


i=2 j=6

5 Bellman-Ford算法



i=3 j=3

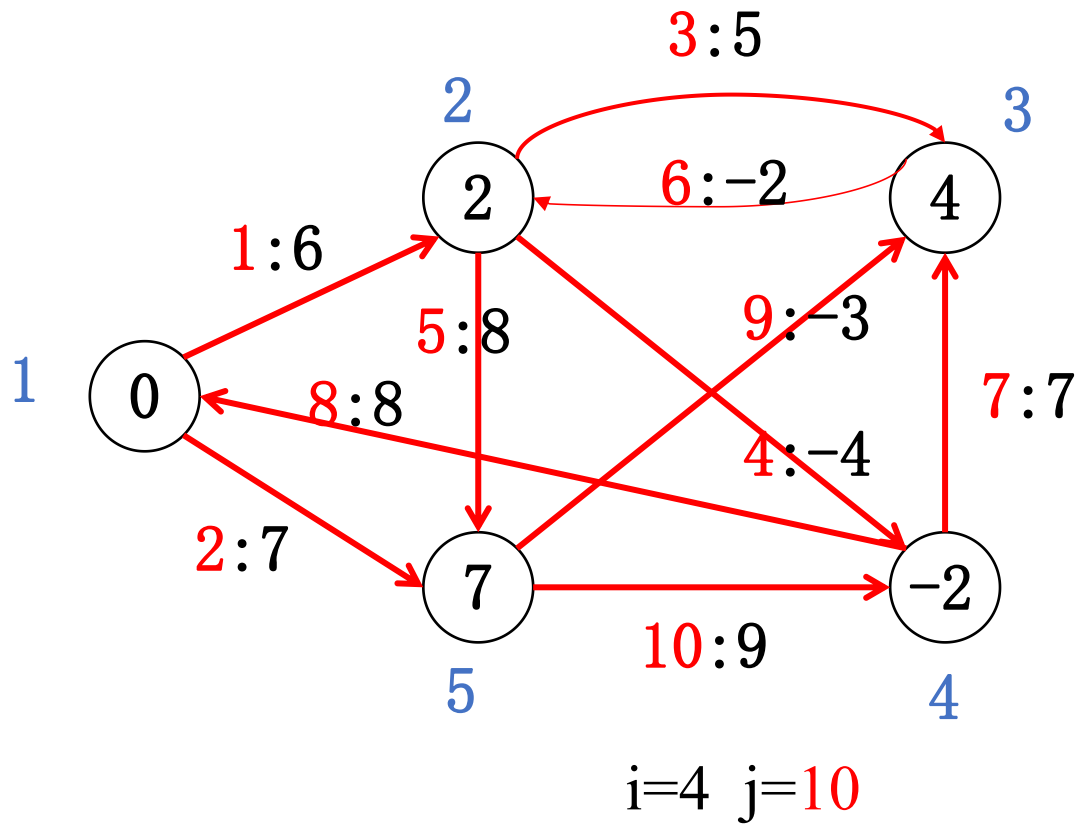


i=3 j=4

5 Bellman-Ford算法

算法伪代码：

```
1  Bellman-Ford(G,w,s)
2     for v in V:
3        d[v]= ∞; p[v]=NULL;
4        d[s]=0;
5    for i from 1 to |V|-1 :
6        for each e (u, v) ∈E:
7            Relax(u,v,w)
8   for each e (u, v) ∈E:
9        if d[v] > d[u] + w(u, v)
10           return FALSE
11   return True
```



5 Bellman-Ford算法

算法时间复杂度分析：

```
1  Bellman-Ford(G,w,s)
2     for v in V:
3        d[v]= ∞; p[v]=NULL;
4        d[s]=0;
5    for i from 1 to |V|-1 :
6        for each e (u, v) ∈E:
7            Relax(u,v,w)
8    for each e (u, v) ∈E:
9        if d[v] > d[u] + w(u, v)
10           return FALSE
11   return True
```

O(|E|)

O(|E|)

O(|V|*|E|)

5 Bellman-Ford算法

正确性证明：

给定带权图G=(V, E, W)，如果图G中不包含从源点s可达的负权环路，那么上述算法在|V|-1次迭代之后对所有s可达的点v都有d[v]=$\delta(s, v)$。

证明：数学归纳法，对于路径长度进行归纳，证明长度为n的最短路径在第n次循环就已经得到

基础步：长度为0的最短路径在第0次循环就已经得到，由于没有负权环路，初始化阶段就有d[s]=$\delta(s, s)$=0；

归纳步：长度小于n的最短路径在第n次循环就已经得到，长度为n+1最短路径有两部分组成：长度为n的最短路径和只有一条边e_{n+1}的v_n.d=$\delta(s, v_n)$一条最短路径。

长度为n的最短路径按归纳假设有，于是按照松弛操作有v_{n+1}.d=v_n.d+w(e_{n+1})=$\delta(s, v_{n+1})$

5 Bellman-Ford算法

正确性证明：

如果d[v] 在Bellman-Ford算法的|V|-1次迭代之后还能继续松弛，那么图中有负权环路。

证明：如果在|V|-1次循环之后点v还能继续松弛，那么从s到v的最短路径比|V|-1还要长，那么其中必须有个环路。

这个环路必然权重是负数，否则不会松弛

湖南大学 HUNAN UNIVERSITY

目录

6 差分约束和最短路径

线性规划问题：

研究线性约束条件下线性目标函数的极值问题的数学理论和方法。

定义：给定一个m*n维矩阵A、n维向量c和m维向量b，求一个n维变量矩阵，在$Ax \leq b$的情况下，最大化$\sum_{i=0}^{n} c_i x_i$

6 差分约束和最短路径

差分约束系统：

对于线性规划问题而言，如果矩阵A中每一行只有一个1和-1且其它值为0。因此，由$Ax \leq b$所给出的约束条件变为m个涉及n个变量的差额限制条件，其中每个约束条件是如下所示的简单线性不等式：

$$x_i - x_j \leq b_k$$

这里$1 \leq i \quad j \leq n \quad 1 \leq k \leq m$

6 差分约束和最短路径

差分约束系统：

For example,consider the problem of finding a 5-vector $x = (x_i)$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

6 差分约束和最短路径

差分约束系统：

$$x_1 - x_2 \leq 0$$

$$x_1 - x_5 \leq -1$$

$$x_2 - x_5 \leq 1$$

$$x_3 - x_1 \leq 5$$

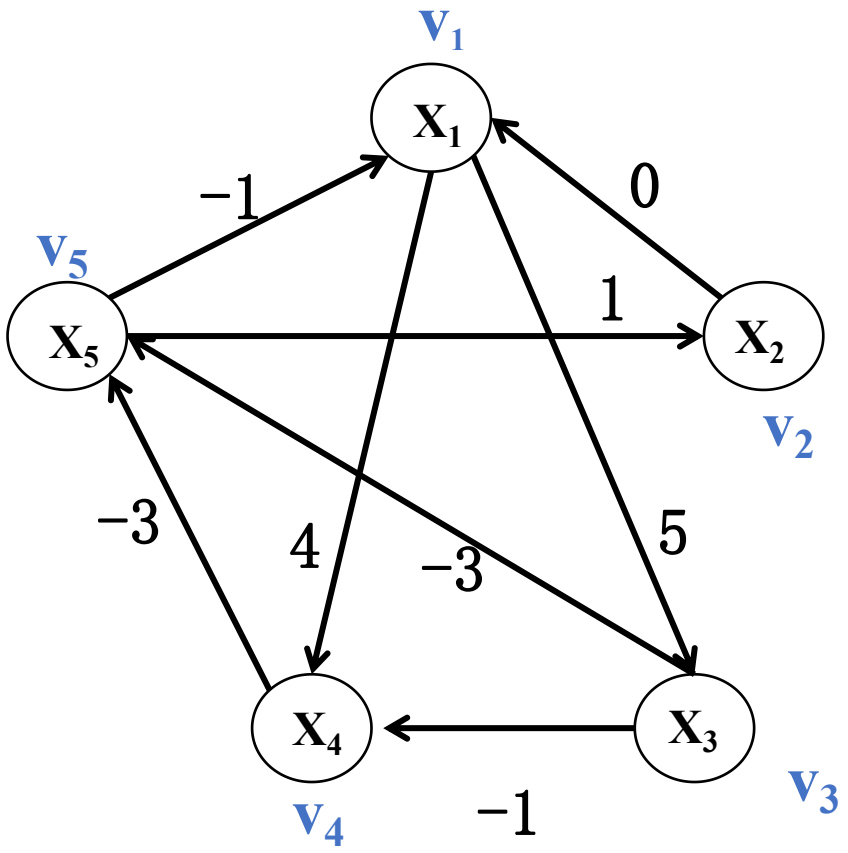
$$x_4 - x_1 \leq 4$$

$$x_4 - x_3 \leq -1$$

$$x_5 - x_3 \leq -3$$

$$x_5 - x_4 \leq 3$$

6 差分约束和最短路径



6 差分约束和最短路径

限制图：

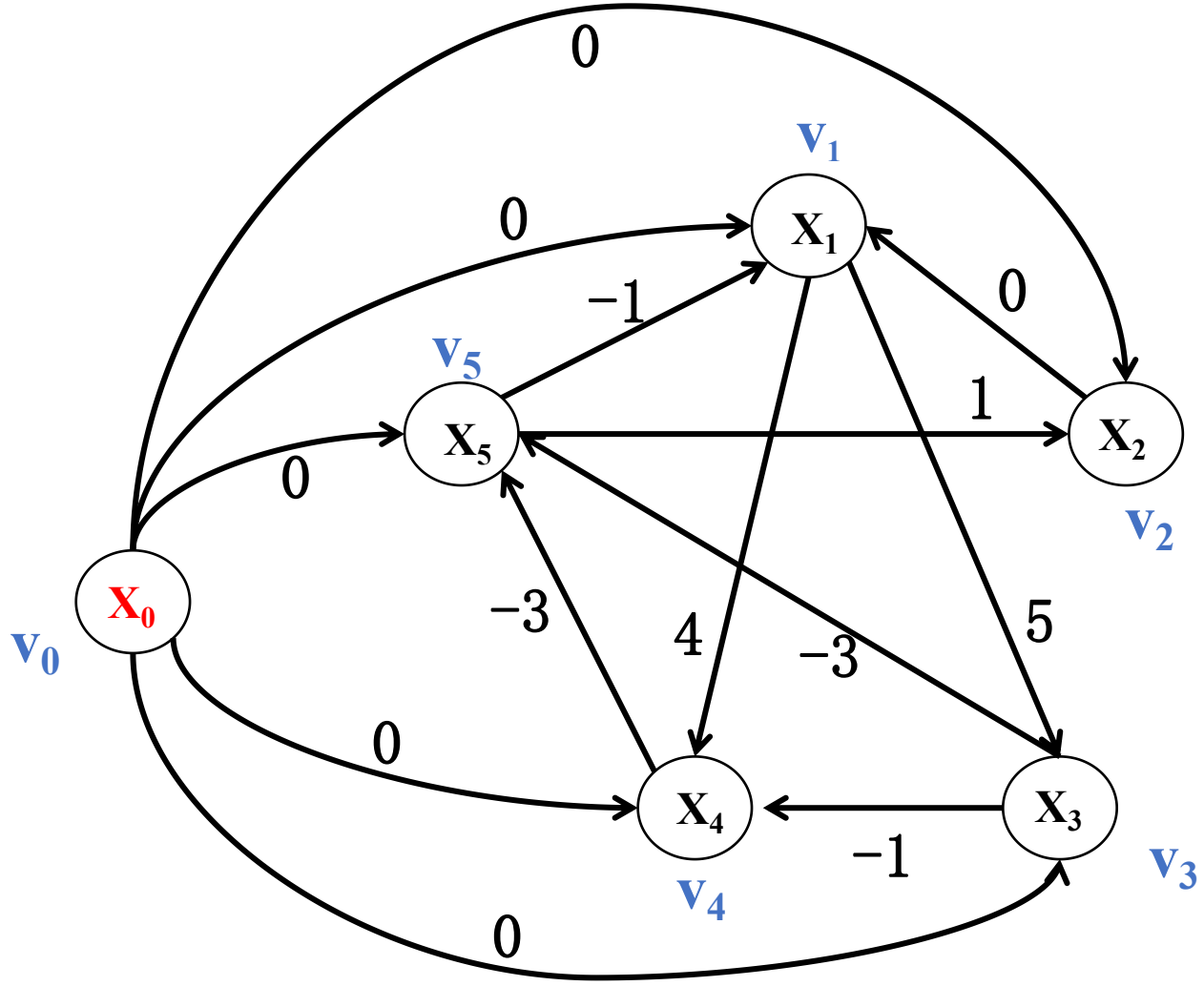
$$x_1 - x_0 \leq 0$$

$$x_2 - x_0 \leq 0$$

$$x_3 - x_0 \leq 0$$

$$x_4 - x_0 \leq 0$$

$$x_5 - x_0 \leq 0$$



6 差分约束和最短路径

差分约束系统：

$$x_1 - x_2 \leq 0$$

$$x_1 - x_5 \leq -1$$

$$x_2 - x_5 \leq 1$$

$$x_3 - x_1 \leq 5$$

$$x_4 - x_1 \leq 4$$

$$x_4 - x_3 \leq -1$$

$$x_5 - x_3 \leq -3$$

$$x_5 - x_4 \leq 3$$

可能解：

x=(-5,-3,0,-1,-4)

x’=(0,2,5,4,1)

6 差分约束和最短路径

引理1：

如果存在一组解 $\{x_1,x_2,..,x_n\}$的话，那么对于任何一个常数 d 有 $\{x_1+d,x_2+d,..,x_n+d\}$也肯定是一组解，因为任何两个数加上一个数以后，它们之间的关系（差）是不变的，这个差分约束系统中的所有不等式都不会被破坏。

证明见算法导论P388

6 差分约束和最短路径

定理（算法导论P389）：

如果限制图中没有负权环路，那么上述算法求出的最短路径距离就是一个差分约束系统的解。

证明：对于任意一条限制图上的边(x_i, x_j)，根据三角不等式，我们有$\delta(s, x_i) + w_{ij} \geq \delta(s, x_j)$，于是$\delta(s, x_i) - \delta(s, x_j) \leq w_{ij}$，即差分约束

6 差分约束和最短路径

定理（算法导论P389）：

如果限制图G中有负权环路，那么这个差分约束系统没有结果

证明：假设存在一条负权环路 $v_0e_1v_1e_2 \dots e_nv_n\ e_{n+1}v_0$且$x_i$对应$v_i$，那么

$$x_0 - x_1 \leq w_{01}$$

$$x_1 - x_2 \leq w_{12}$$

……

$$x_n - x_0 \leq w_{n0}$$

上面的不等式左右分别求和，就有$0 \leq w_{01} + w_{12} + \dots + w_{n0}$，由于$v_0e_1v_1e_2 \dots e_nv_n\ e_{n+1}v_0$是负权环路，那么$w_{01} + w_{12} + \dots + w_{n0}<0$。

6 差分约束和最短路径

求解差分约束系统：

基于Bellman-Ford算法的算法。首先，在限制图G的基础上，构造一个图G′，引入一个源点s，s到所有点都有一条边，长度都为0

然后，对G’进行Bellman-Ford算法得到s到所有点的最短路径距离

这些距离就是x_i.d就是一个解

复杂度为O(m*n)

湖南大學
HUNAN UNIVERSITY

谢谢观赏

实事求是　敢为人先