

算法设计与分析

Computer Algorithm Design & Analysis

吕志鹏

zhipeng.lv@hust.edu.cn

班级名称：算法设计与分析2025



算法设计与分析2025
群号: 419123077



QQ: 419123077



微助教: QE854

A decorative graphic on the left side of the slide, consisting of a black crosshair with blue, red, and yellow squares at the intersections.

Chapter 24

Single-Source Shortest Paths

单源最短路径

最短路径问题:

给定一个带权重的有向图 $G=(V,E)$ 和权重函数 $\omega:E\rightarrow R$ 。图中一条路径 $p=\langle v_0, v_1, \dots, v_k \rangle$ 的权重 $\omega(p)$ 是构成该路径的所有边的权重之和:

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i) .$$

从结点 u 到结点 v 的最短路径权重 $\delta(u,v)$ 定义如下:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from } u \text{ to } v , \\ \infty & \text{otherwise .} \end{cases}$$



单源最短路径问题：

给定一个图 $G=(V,E)$ ，找出从给定的源点 $s \in V$ 到其它每个结点 $v \in V$ 的最短路径。

■ 最短路径的最优子结构

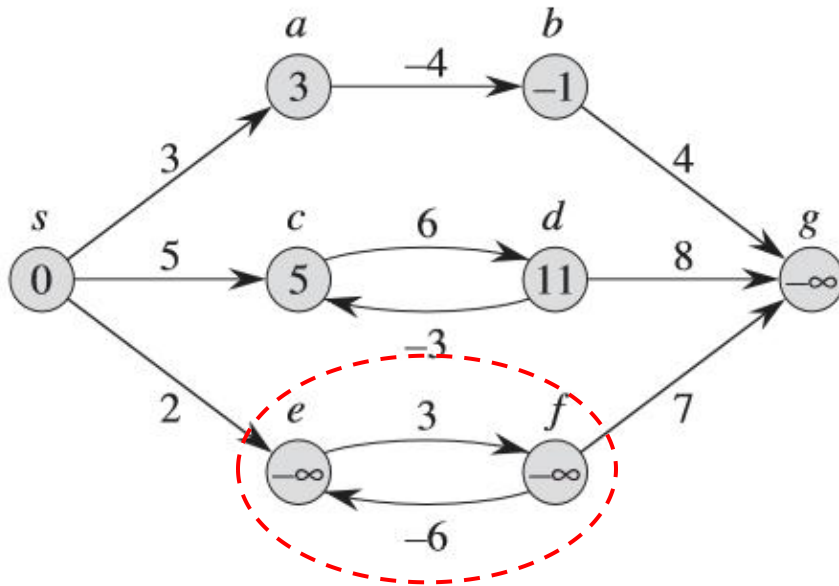
这样最短路径具有最优子结构性：两个结点之间的最短路径的任何子路径都是最短的。

引理24.1 给定一个带权重的有向图 $G=(V,E)$ 和权重函数 $\omega:E \rightarrow \mathbb{R}$ 。设 $p=\langle v_0, v_1, \dots, v_k \rangle$ 为从结点 v_0 到结点 v_k 的一条最短路径，并且对于任意的 i 和 j ， $0 \leq i \leq j \leq k$ ，设 $p_{ij}=\langle v_i, v_{i+1}, \dots, v_j \rangle$ 为路径 p 中从结点 v_i 到结点 v_j 的子路径，则 p_{ij} 是从结点 v_i 到结点 v_j 的一条最短路径。（证明略，见P375）

■ 负权重的边

权重为负值的边称为**负权重的边**。

- 如果存在负权重的边，则有可能存在**权重为负值的环路**，而造成图中**最短路径无定义**（路径的权重为 $-\infty$ ）。



■ 环路

- 最短路不应包含环路。
 - 不包含环路的路径称为简单路径。
 - 对任何简单路径最多包含 $|V|-1$ 条边和 $|V|$ 个结点。
 - 不失一般性，假设后续算法寻找的最短路径都不包含环路。

■ 最短路径的表示

- 一个结点的前驱结点记为： $v.\pi$
 - 前驱结点或者为NIL或者为另一个结点
- 利用 $v.\pi$ 的记录可以搜索出最短路径上的所有结点。

■ 前驱子图

定义前驱子图为 $G_\pi = (V_\pi, E_\pi)$ ，其中，

- 结点集合 $V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$
 - 即 V_π 是图 G 中的前驱结点不为NIL的结点的集合，再加上源点 s 。
- 边集合 $E_\pi = \{(v.\pi, v) \in E : v \in V_\pi - \{s\}\}$
 - 即 E_π 是由 V_π 中的结点的 π 值所“诱导” (induced) 的边的集合。

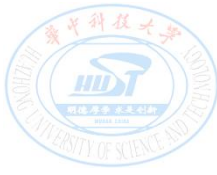
则，算法终止时， **G_π 是一棵最短路径树。**

- 该树包含了从源结点 s 到每个可以从 s 到达的结点的一条最短路径。

设 $G=(V,E)$ 是一条带权重的有向图，其权重函数为 $\omega:E\rightarrow R$ ，假定 G 不包含从 s 可以到达的权重为负值的环路，因此，所有的最短路径都有定义。

一棵根结点为 s 的最短路径树是一个有向子图 $G'=(V', E')$ ，这里 $V' \subseteq V$ ， $E' \subseteq E$ 。且有以下性质：

- (1) V' 是图 G 中从源结点 s 可以到达的所有结点的集合。
- (2) G' 形成一棵根结点为 s 的树。
- (3) 对于所有的结点 $v \in V'$ ，图 G' 中从结点 s 到结点 v 的唯一简单路径是图 G 中从结点 s 到结点 v 的一条最短路径。



■ 松弛操作 (Relax)

对于每个结点 v ，维持一个属性 $v.d$ ，记录从源点 s 到结点 v 的最短路径权重的上界。称 $v.d$ 为 s 到 v 的最短路径估计。

- 过程INITIALIZE-SINGLE-SOURCE实现对结点最短路径估计和前驱结点的初始化：

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
1  for each vertex  $v \in G.V$   
2       $v.d = \infty$   
3       $v.\pi = \text{NIL}$   
4   $s.d = 0$ 
```

初始化后，对所有的结点 $v \in V$ 有，

- $v.\pi = \text{NIL}$;
- $s.d = 0$;
- 对所有的结点 $v \in V - \{s\}$ 有： $v.d = \infty$ 。

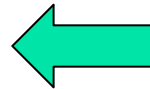
INITIALIZE-SINGLE-SOURCE 的时间： $\Theta(V)$

松弛操作：首先测试一下是否可以对从s到v的最短路径进行改善（即有没有更短的路径）。如果可以改善，则v.d更新为新的最短路径估计值，v的前驱v.π更新为新的前驱结点。

RELAX(u, v, w)

```

1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
    
```



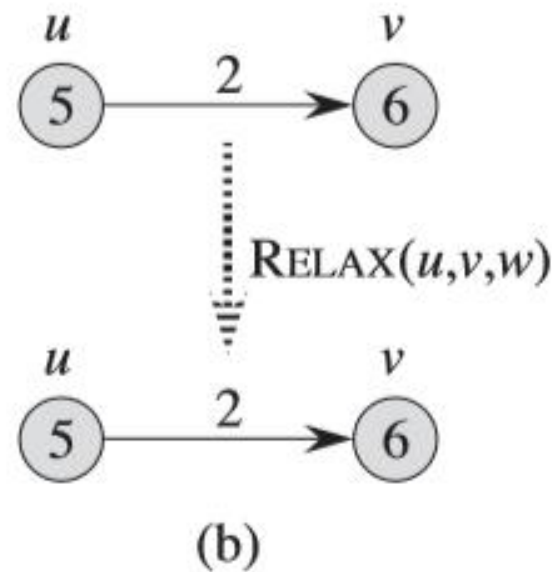
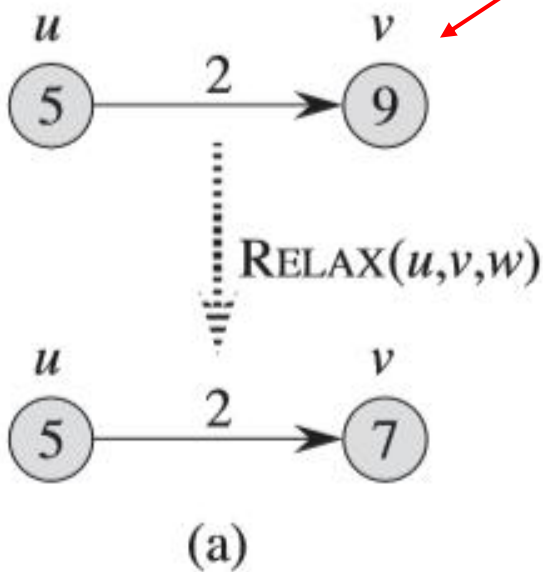
测试：对s到v所经过的**最后一个中间结点u**，按下列方式计算从s出发，经过u而到达v的路径的权重：

将从结点s到结点u之间最短路径加上结点u到v之间的边的权重，然后与当前的s到v的最短路径估计v.d进行比较，看有没有变小。如果变小，则对v.d和v.π进行更新——这一操作就称为“**松弛**”

RELAX 的时间：O(1)

例:

结点中的数字：每个结点的最短路径估计



对边 (u, v) 进行松弛操作，权重： $\omega(u, v) = 2$ ：

(a) : 因为 $v.d > u.d + \omega(u, v)$ ，所以 $v.d$ 的值减小 ($9 > 5 + 2$)。

(b) : 因为 $v.d \leq u.d + \omega(u, v)$ ，所以 $v.d$ 的值没有改变。



最短路径和松弛操作的性质

1. 三角不等式性质

引理24.11 设 $G=(V, E)$ 为一个带权重的有向图，其权重函数为 $\omega:E \rightarrow \mathbb{R}$ ，设其源结点为 s 。那么对于所有的边 $(u, v) \in E$ ，有

$$\delta(s, v) \leq \delta(s, u) + \omega(u, v)$$

证明：

假定 p 是从源结点 s 到结点 v 的一条最短路径，则 p 的权重不会比任何从 s 到 v 的其它路径的权重大，因此路径 p 的权重也不会比这样的一条路径的权重大：从源结点 s 到结点 u 的一条最短路径，再加上边 (u, v) 而到达结点 v 的这条路径。

如果 s 到 v 没有最短路径，则不可能存在到 v 的路径。 ■

2. 上界性质: $v.d$ 是 s 到 v 的最短路径权重 $\delta(s, v)$ 的上界

引理24.11 设 $G=(V,E)$ 为一个带权重的有向图, 其权重函数为 $\omega:E \rightarrow \mathbb{R}$, 设其源结点为 s , 该图由算法 *INITIALIZE-SINGLE-SOURCE*(G, s) 执行初始化。那么对于所有的结点 $v \in V$, $v.d \geq \delta(s, v)$ 。并且该不变式在对图 G 的边进行任何次序的松弛过程中都保持成立, 而一旦 $v.d$ 取得其下界 $\delta(s, v)$ 后, 将不再发生变化。

用数学归纳法证明: 对于所有的结点 $v \in V$, $v.d \geq \delta(s, v)$ 。

➤ 注: 归纳的主体是松弛步骤的数量。

基础步: 在经 *INITIALIZE-SINGLE-SOURCE*(G, s) 初始化之后, 对于所有的结点 $v \in V - \{s\}$, 置 $v.d = \infty$, 而 $s.d = 0$, 显然 $s.d \geq \delta(s, s)$, 而其它的结点 $v.d \geq \delta(s, v)$, 结论成立。



归纳步：考虑对边 (u, v) 的松弛操作。

归纳假设：在对边 (u, v) 进行松弛之前，对所有的结点 $x \in V$,

$$x.d \geq \delta(s, x)。$$

而在**对边 (u, v) 进行松弛**的过程中，唯一可能发生改变的值只有 $v.d$ ，如果该值发生变化，则有：

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &\geq \delta(s, u) + w(u, v) \quad (\text{by the inductive hypothesis}) \\ &\geq \delta(s, v) \quad (\text{by the triangle inequality}) , \end{aligned}$$

同时，根据计算的规则，在 $v.d$ 达到其下界 $\delta(s, v)$ 后，就无法再减小（也不可能增加）。

引理得证。

RELAX (u, v, w)

```
1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
```

3. 非路径性质

推论24.12 给定一个带权重的有向图 $G=(V,E)$ ，其权重函数为 $\omega:E\rightarrow\mathbb{R}$ 。假定从源结点 s 到给定点 v 之间**不存在路径**，则该图在由算法 *INITIALIZE-SINGLE-SOURCE*(G,s) 进行初始化后，有

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

并且该等式作为不变式一直维持到图 G 的所有松弛操作结束。

证明：

因为从源点 s 到给定点 v 之间**不存在路径**，所以 $\delta(s,v) = \infty$ 。

而根据上界性质，总有 $v.d \geq \delta(s,v)$ ，所以， $v.d \geq \delta(s,v) = \infty$ 。

得证。

引理24.13 设 $G=(V,E)$ 为一个带权重的有向图，其权重函数为 $\omega:E \rightarrow \mathbb{R}$ ，并且边 $(u,v) \in E$ 。那么在对边 (u,v) 进行松弛操作RELAX(u,v,ω)后，有 $v.d \leq u.d + \omega(u,v)$ 。

证明：

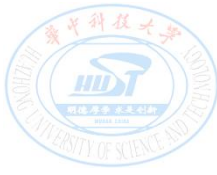
如果在对边 (u,v) 进行松弛操作前，有 $v.d > u.d + \omega(u,v)$ ，则松弛操作时，置 $v.d = u.d + \omega(u,v)$ 。

如果在松弛操作前有 $v.d \leq u.d + \omega(u,v)$ ，则松弛操作不会改变 $v.d$ 和 $u.d$ 的值，因此在松弛操作后仍有 $v.d \leq u.d + \omega(u,v)$ 。

得证。

```

RELAX( $u, v, w$ )
1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
    
```

4. 收敛性质

引理24.14 设 $G=(V,E)$ 为一个带权重的有向图，其权重函数为 $\omega:E\rightarrow\mathbb{R}$ 。设 $s\in V$ 为某个源结点， $s \rightsquigarrow u \rightarrow v$ 为图 G 中的一条最短路径 ($u, v\in V$)。

假定图 G 由算法 *INITIALIZE-SINGLE-SOURCE*(G,s) 进行初始化，并在这之后进行了一系列边的松弛操作，其中包括对边 (u,v) 的松弛操作 *RELAX*(u,v,ω)。如果在对边 (u,v) 进行松弛操作之前的某时刻有 $u.d = \delta(s,u)$ ，则在该松弛操作之后的所有时刻有 $v.d = \delta(s,v)$ 。



证明:

根据上界性质, 如果在对边 (u, v) 进行松弛前的某个时刻有 $u.d = \delta(s, u)$, 则该等式在松弛之后仍然成立。

特别地, 在对边 (u, v) 进行松弛后, 有

$$\begin{aligned} v.d &\leq u.d + w(u, v) && \text{(by Lemma 24.13)} \\ &= \delta(s, u) + w(u, v) \\ &= \delta(s, v) && \text{(by Lemma 24.1) .} \\ &&& \text{最优子结构性} \end{aligned}$$

而根据上界性质, 有 $v.d \geq \delta(s, v)$ 。所以有 $v.d = \delta(s, v)$, 并且该等式在此之后一直保持成立。

得证。

4. 路径松弛性质

引理24.15 设 $G=(V,E)$ 为一个带权重的有向图，其权重函数为 $\omega:E\rightarrow\mathbb{R}$ 。设 $s\in V$ 为某个源结点，考虑从源结点 s 到结点 v_k 的任意一条最短路径 $p=\langle v_0, v_1, \dots, v_k \rangle$ ， $v_0=s$ 。

如果图 G 由算法 *INITIALIZE-SINGLE-SOURCE*(G,s) 进行初始化，并在这之后进行了一系列边的松弛操作，其中包括对边 (v_0, v_1) 、 (v_1, v_2) 、 \dots 、 (v_{k-1}, v_k) 按照所列次序而进行的松弛操作，则在所有这些松弛操作之后，有 $v_k.d = \delta(s, v_k)$ ，并且在此之后该等式一直保持。

该性质的成立与其他边的松弛操作及次序无关，即使这些松弛操作是与对 p 上的边所进行的松弛操作穿插进行的。

归纳法证明： 在最短路径 p 的第 i 条边被松弛之后，有 $v_i.d = \delta(s, v_i)$

基础步： 在对路径 p 的任何一条边进行松弛操作之前，从初始化算法可以得出： $v_0.d = s.d = \delta(s, s)$ 。结论成立，且 $s.d$ 的取值在此之后不再发生变化。

归纳步： 假定依次经过 (v_0, v_1) 、 (v_1, v_2) 、 \dots 、 (v_{i-2}, v_{i-1}) 松弛操作之后， $v_{i-1}.d = \delta(s, v_{i-1})$ 。

则在对边 (v_{i-1}, v_i) 进行松弛操作时，根据**收敛性质**，必有在对该边进行松弛后 $v_i.d = \delta(s, v_i)$ ，并且该等式在此之后一直保持成立。

得证。

24.1 Bellman-ford算法

Bellman-ford算法可以求解一般情况下的单源最短路径问题
——可以有负权重的边，但不能有负权重的环。

设 $G=(V, E)$ 为一个带权重的有向图，其权重函数为 $\omega : E \rightarrow \mathbb{R}$ 。

$s \in V$ 为源结点。

BELLMAN-FORD(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```

算法返回TRUE当且仅当图G中不包含从源结点可达的权重为负值的环路。

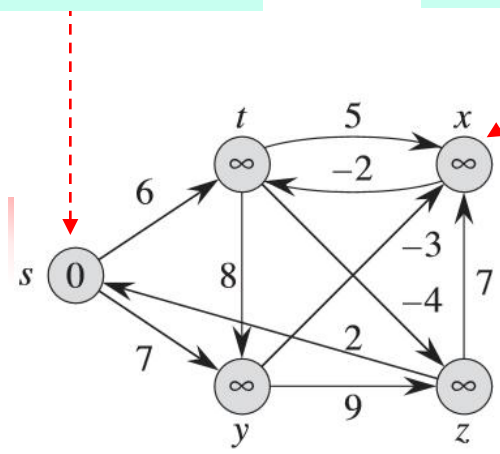
```

RELAX( $u, v, w$ )
1  if  $v.d > u.d + w(u, v)$ 
2      $v.d = u.d + w(u, v)$ 
3      $v.\pi = u$ 

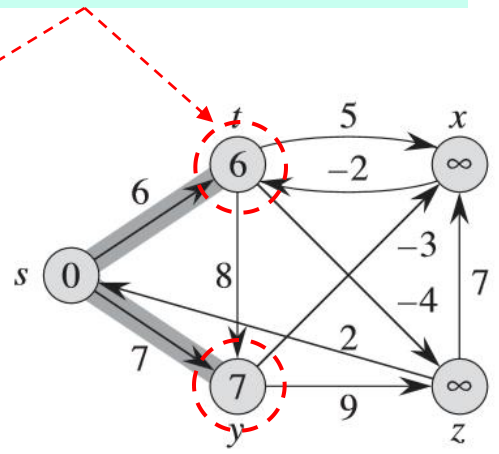
```

源结点: s

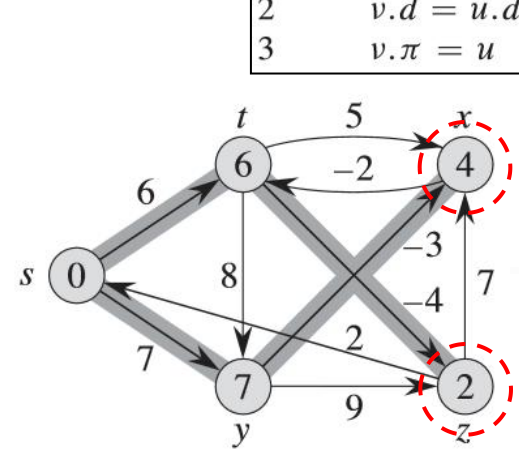
结点中的数值是结点的 d 值



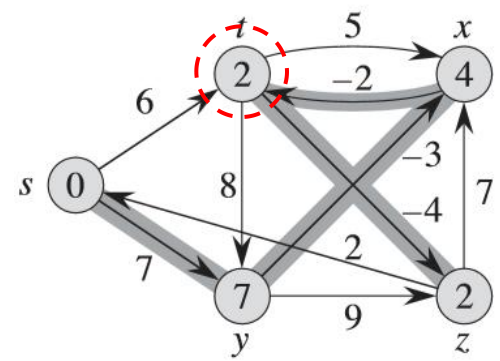
(a) 初始化后



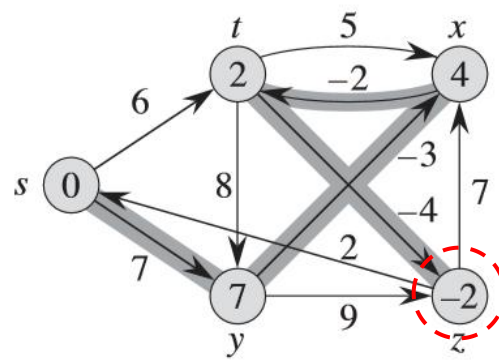
(b) 第一次松弛操作后



(c) 第二次松弛操作后



(d) 第三次松弛操作后



(e) 第四次松弛操作后

初始化后，算法将对图中每条边进行 $|V|-1$ 次松弛处理：
for循环执行 $|V|-1$ 次，
每次对所有的边进行进行一次松弛处理。

例，Bellman-ford算法的执行过程

- 加了阴影的边表示前驱值：如果边 (u, v) 加了阴影，则 $v.\pi = u$ 。
- 本例中Bellman-ford算法执行4次松弛操作后返回TRUE。

Bellman-ford算法的运行时间

初始化: $\Theta(V)$ →

松弛处理: for循环执行 $|V|-1$ 次,
每次的时间是 $\Theta(E)$

BELLMAN-FORD(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```

$O(E)$

- Bellman-ford算法总的运行时间 $O(VE)$ 。

INITIALIZE-SINGLE-SOURCE(G, s)

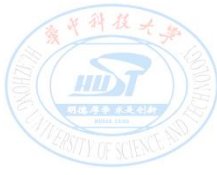
```
1  for each vertex  $v \in G.V$ 
2       $v.d = \infty$ 
3       $v.\pi = \text{NIL}$ 
4   $s.d = 0$ 
```

$\Theta(V)$

RELAX(u, v, w)

```
1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
```

$\Theta(1)$

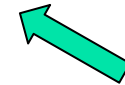


Bellman-ford算法的证明

设 $G=(V, E)$ 为一个带权重的源点为 s 的有向图，其权重函数为 $\omega : E \rightarrow \mathbb{R}$ ，并假定图 G 中不包含从源结点 s 可以到达的权重为负值的环路。

引理24.2 Bellman-ford算法的第2~4行的for循环在执行 $|V|-1$ 次之后，对于所有从源结点 s 可以到达的结点 v 有

$$v.d = \delta(s, v)。$$



v.d到达下界

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```


引理24.2 假定图G中不包含从源结点s可以到达的权重为负值的环路。则Bellman-ford算法的第2~4行的for循环在执行 $|V|-1$ 之后，对于所有从源结点s可以到达的结点v有 $v.d = \delta(s, v)$ 。

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```

证明：（使用**路径松弛性质**证明）

考虑任意从源结点s可以到达的结点v。设 $p = \langle v_0, v_1, \dots, v_k \rangle$ 是从结点s到结点v之间的任意一条最短路径，这里 $v_0 = s$ ， $v_k = v$ 。

- 因为最短路径都是简单路径，所以p中最多包含 $|V|-1$ 条边，故 $k \leq |V|-1$ 。
- 同时，算法第2~4行的for循环每次松弛所有的 $|E|$ 条边，**每一次为p最多扩展一条边**（想想为什么？）。所以对序列 $\langle v_0, v_1, \dots, v_k \rangle$ 在其第i次松弛操作时，被松弛的边中包含边 (v_{i-1}, v_i) ，这里 $i = 1, 2, \dots, k$ 。
- 根据路径松弛性质有： $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$ 。得证。

引理24.3 对所有结点 $v \in V$ ，存在一条从源结点 s 到结点 v 的路径当且仅当Bellman-ford算法终止时有 $v.d < \infty$ 。

证明：（略）

定理24.4（Bellman-ford算法的正确性） 设Bellman-ford算法运行在一个带权重的源点为 s 的有向图 $G=(V,E)$ 上，其权重函数为 $\omega:E \rightarrow \mathbb{R}$ 。

- 如果图 G 中不包含从源结点 s 可以到达的权重为负值的环路，则算法将返回TRUE，且对于所有结点 $v \in V$ ，前驱子图 G_π 是一个根结点为 s 的最短路径树。
- 而如果图 G 中包含一条从源结点 s 可以到达的权重为负值的环路，则算法将返回FALSE。



定理24.4 (Bellman-ford算法的正确性) 的证明 (略) :

1) **首先证明**: 如果图 G 中不包含从源结点 s 可以到达的权重为负值的环路, 则算法将返回TRUE, 且对于所有结点 $v \in V$, 前驱子图 G_π 是一个根结点为 s 的最短路径树。

证明:

(1) 证明: 对于所有结点 $v \in V$, 在算法终止时, 有 $v.d = \delta(s, v)$ 。

- 如果结点 v 是从 s 可以到达的, 则论断可以从引理24.2得到证明。
- 如果结点 v 不能从 s 可达, 则论断可以从非路径性质获得。

因此, 对于所有结点 $v \in V$, 在算法终止时, 有 $v.d = \delta(s, v)$ 。

(2) 综合前驱子图性质和本论断, 可以推导出 G_π 是一棵最短路径树



(3) 终止时, 算法是否返回TRUE?

算法终止时, 对所有的边 $(u,v) \in E$, 有

$$\begin{aligned} v.d &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \quad (\text{by the triangle inequality}) \\ &= u.d + w(u, v), \end{aligned}$$

因此, 算法第6行中没有任何测试可以让算法返回FALSE (G 中不包含从源结点 s 可以到达的权重为负值的环路), 因此一定返回TRUE值。

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```

2) 然后证明：如果图G中包含一条从源结点s可以到达的权重为负值的环路，则算法将返回FALSE。

证明：

假定图G包含一个权重为负值的环路，并且该环路可以从源结点s到达。设该环路为 $c = \langle v_0, v_1, \dots, v_k \rangle$ ，这里 $v_0 = v_k$ 。

因为环路的权重为负值，所以有：

$$\sum_{i=1}^k w(v_{i-1}, v_i) < 0 . \quad (24.1)$$

反证法证明： 假设此种情况下Bellman-ford算法返回TRUE值，
则有： 对所有的 $i=1,2,\dots,k$,

$$v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$$

将环路c上的所有这种不等式都加起来，有：

$$\begin{aligned} \sum_{i=1}^k v_i.d &\leq \sum_{i=1}^k (v_{i-1}.d + w(v_{i-1}, v_i)) \\ &= \sum_{i=1}^k v_{i-1}.d + \sum_{i=1}^k w(v_{i-1}, v_i) \end{aligned}$$

由于 $v_0 = v_k$, 环路c上面的每个结点在上述求和表达式 $\sum_{i=1}^k v_i \cdot d$ 和 $\sum_{i=1}^k v_{i-1} \cdot d$ 中都刚好各出现一次。因此有

$$\sum_{i=1}^k v_i \cdot d = \sum_{i=1}^k v_{i-1} \cdot d .$$

因此有 $0 \leq \sum_{i=1}^k w(v_{i-1}, v_i)$

与 $\sum_{i=1}^k w(v_{i-1}, v_i) < 0$. 相矛盾。

因此, 如果图G中不包含从源结点s可以到达的权重为负值的环路, 则算法将返回TRUE, 否则返回FALSE。 得证。

24.3 Dijkstra算法

- Dijkstra算法解决带权重的有向图上单源最短路径问题。
- 该算法要求所有边的权重均为非负值，即对于所有的边 $(u,v) \in E$, $\omega(u,v) \geq 0$, —— 不能有负权重的边和环。

DIJKSTRA(G, w, s)

```
1 INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  $S = \emptyset$ 
3  $Q = G.V$ 
4 while  $Q \neq \emptyset$ 
5      $u = \text{EXTRACT-MIN}(Q)$ 
6      $S = S \cup \{u\}$ 
7     for each vertex  $v \in G.Adj[u]$ 
8         RELAX( $u, v, w$ )
```

算法从结点集 $V-S$ 中选择当前最短路径估计最小的结点 u ，将 u 从 Q 中删除，并加入到 S 中， $u.d$ 就是源结点 s 到 u 的最短路径的长度。这里 Q 是一个最小优先队列，保存结点集 $V-S$ 。

然后对所有从 u 出发的边进行松弛。然后重复上述过程，直到 $Q = \emptyset$ 。

Dijkstra算法是一个贪心算法：每次总是选择V-S集合中最短路径估计值最小的结点加入S中。

DIJKSTRA(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = \text{EXTRACT-MIN}(Q)$ 
6       $S = S \cup \{u\}$ 
7      for each vertex  $v \in G.Adj[u]$ 
8          RELAX( $u, v, w$ )
```

每个结点有且仅有一次机会被从Q中抽取并加入S中。一旦u被从Q中抽取出来，u.d就是s到u的最短路径长度（不再改变，上界性质）。

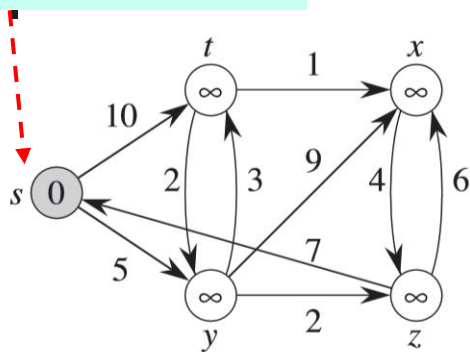
u加入S后，对从u出发的边的(u,v)进行松弛。而如果v.d变小，则是因为存在从s经过u到达v的更短路径所致。此时，修改 $v.d = u.d + \omega(u, v)$ ， $v.\pi = u$ ，即最短路径上v结点的新前驱为u。

while循环总共执行了 $|V|$ 次

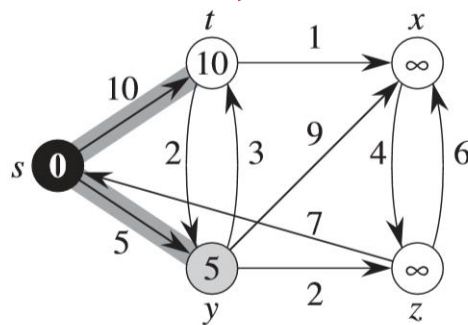
源结点: s

开始的时候 $s \in S$

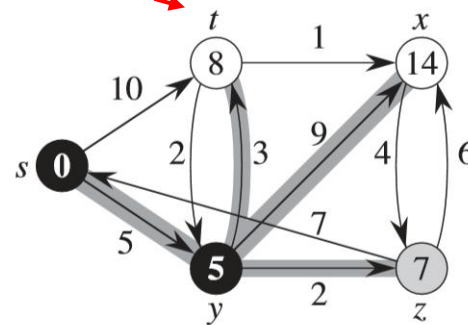
结点中的数值是 s 到该结点的最短路径的估计值



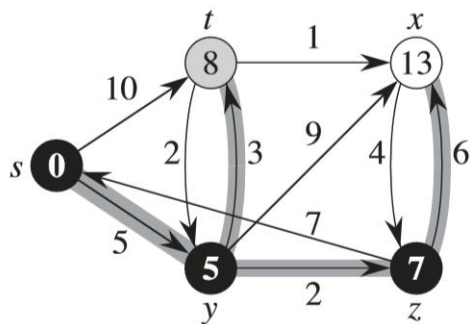
(a)



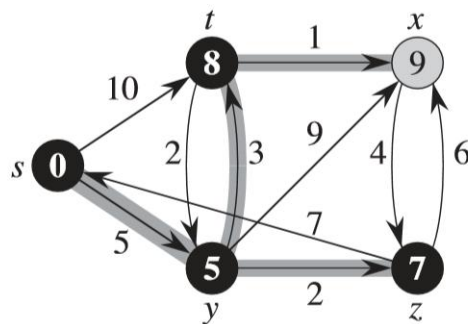
(b)



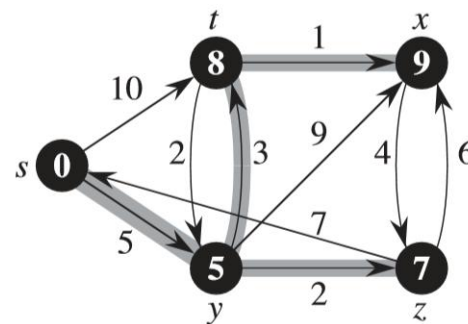
(c)



(d)



(e)



(f)

例, Dijkstra算法的执行过程

- 加了阴影的边表明前驱值 (当前 u 出发的边)。
- 黑色的结点属于 S , 白色的结点属于 $V-S$ 。加阴影的结点是算法下一次循环将选择加入 S 的点。

定理24.6 (Dijkstra算法的正确性) 设Dijkstra算法运行在带权重的有向图 $G=(V,E)$ 上。如果所有边的权重为非负值, 则在算法终止时, 对于所有结点 $u \in V$, 有 $u.d = \delta(s,u)$ 。

证明: 利用循环不变式证明

循环不变式: 算法在while语句的每次循环开始前, 对于每个结点 $u \in S$, 有 $u.d = \delta(s,u)$

只需证明: 对于每个结点 $u \in V$, 当 u 被加入到 S 时, 有 $u.d = \delta(s,u)$ 。

注: 一旦 u 加入 S , 就不会再修正 $u.d$ 。且根据上界性质, 该等式将一直保持。

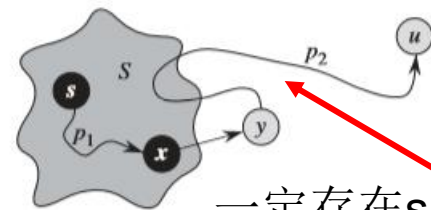
证明过程:

- (1) 初始化: 初始时, $S = \emptyset$, 因此循环不变式直接成立。
- (2) 保持: 在每次循环中, 对于加入到集合 S 中的结点 u 而言,
 $u.d = \delta(s, u)$ 。

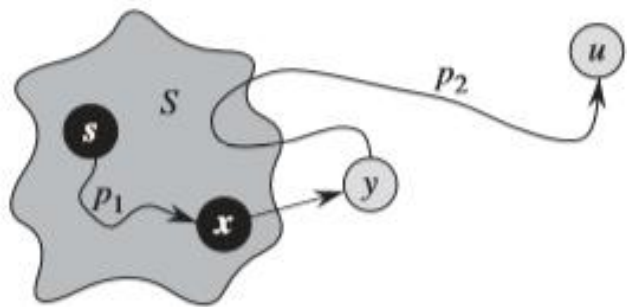
用反证法证明: 设结点 u 是第一个在加入到集合 S 时 $u.d \neq \delta(s, u)$ 的结点。

由于 s 是第一个加入到集合 S 中的结点, 并且 $s.d = \delta(s, s) = 0$, 所以 $u \neq s$, 并且在 u 即将加入 S 时, $S \neq \emptyset$, 因为 S 中至少包含了 s 。

故, 此时必存在至少一条从 s 到 u 的路径 (否则, 根据非路径性质将有 $u.d = \delta(s, u) = \infty$, 与假设的 $u.d \neq \delta(s, u)$ 相矛盾, 故这样路径一定存在), 这样也必存在一条从 s 到 u 的最短路径, 记为 p 。



一定存在 s 到 u 的最短路径 p



考虑路径 p 上第一个满足 $y \in V-S$ 的结点 y ，并设 y 的前驱是结点 x ， $x \in S$ ，如图所示。路径分为：

$$s \xrightarrow{p_1} x \rightarrow y \xrightarrow{p_2} u$$

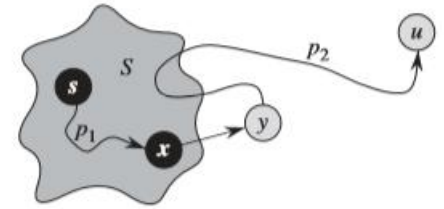
注： x 有可能是 s 本身， y 也有可能是 u 本身（事实上也只能是 u 本身，除非 $\delta(y,u)=0$ ）。

则有：**在结点 u 加入到集合 S 时，应有 $y.d = \delta(s,y)$ 。**

- 这是因为 $x \in S$ ， u 是第一个 $u.d \neq \delta(s,u)$ 的结点，在将 x 加入到集合 S 时，有 $x.d = \delta(s,x)$ ， y 是 x 的邻接点，所以此时边 (x,y) 将被松弛。由于 y 是最短路径 p 上的结点，根据最短路径的最优子结构性和收敛性质，此时应有 $y.d = \delta(s,y)$ 。

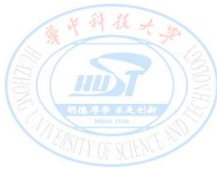
因为结点 y 是从结点 s 到结点 u 的一条最短路径上位于 u 前面的一个结点，所以应有 $\delta(s,y) \leq \delta(s,u)$ ，因此

$$\begin{aligned} y.d &= \delta(s, y) \\ &\leq \delta(s, u) \\ &\leq u.d \quad (\text{by the upper-bound property}) \end{aligned}$$



而在算法第5行选择结点 u 时，结点 u 和 y 都还在集合 $V-S$ 里，所以有 $u.d \leq y.d$ （思考为什么）。因此上式的不等式事实上只能是等式，即： $y.d = \delta(s, y) = \delta(s, u) = u.d$ 。

这与假设的 $u.d \neq \delta(s,u)$ 相矛盾。因此假设不成立。所以， u 在加入 S 时，将有 $u.d = \delta(s,u)$ ，该等式在随后的循环中一直保持。



终止：在算法终止时， $Q=\emptyset$ ， $S=V$ 。

根据前面保持性的证明，终止时对于所有的结点 $u \in V$ ，有 $u.d = \delta(s, u)$ 。

证毕。

推论24.7 如果在带权重的有向图 $G=(V, E)$ 上运行Dijkstra算法，其中的权重皆为非负值，源结点为 s ，则在算法终止时，前驱子图 G_π 是一棵根结点为 s 的最短路径树。

从定理24.6和前驱子图性质可证（证明略）。

```
1 INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  $S = \emptyset$ 
3  $Q = G.V$ 
4 while  $Q \neq \emptyset$ 
5      $u = \text{EXTRACT-MIN}(Q)$ 
6      $S = S \cup \{u\}$ 
7     for each vertex  $v \in G.Adj[u]$ 
8         RELAX( $u, v, w$ )
```

Dijkstra算法运行时间分析

- 根据算法的处理规则，每个结点 u 仅被加入集合 S 一次，邻接链表 $Adj[u]$ 中的每条边在整个运行期间也只被检查一次。因此**算法第7-8行的for循环执行次数总共为 $|E|$ 次**（即松弛判定总次数）。
- Dijkstra算法的总运行时间依赖于**最小优先队列 Q** 的实现。
 - 如果用**线性数组(无序或者按序插入)**实现，每次找 d 最小的结点 u 需要 $O(V)$ 的时间，所以算法的总运行时间为 $O(V^2 + E) = O(V^2)$ 。
 - 如果用**二叉堆**实现，每次找 d 最小的结点 u 需要 $O(\lg V)$ 的时间，所以算法的总运行时间为 $O((V + E) \lg V)$ 。
 - 如果用**斐波那契堆**实现，算法的总运行时间可以改善至 **$O(V \lg V + E)$** 。

24.4 差分约束和最短路径

线性规划：给定一个 $m \times n$ 的矩阵 A 、一个 m 维的向量 b 和一个 n 维的向量 c 。试找一 n 维向量 x ，使得在 $Ax \leq b$ 的约束下，目标函数 $\sum_{i=1}^n c_i x_i$ 最大。

- 求解线性规划问题：单纯形法等
- 本节讨论线性规划的一个特例：差分约束系统。

差分约束系统：在一个差分约束系统中，线性规划矩阵A的每一行包括一个1和一个-1，其它所有项皆为0。由 $Ax \leq b$ 给出的约束条件形式上是**m个涉及n个变量的差额限制条件**(difference constraints)，每个约束条件是以下简单的线性不等关系：

$$x_j - x_i \leq b_k$$

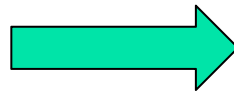
这里 $1 \leq i, j \leq n$, $i \neq j$, 并且 $1 \leq k \leq m$ 。

例：一个满足下列条件的5维向量 $x=(x_i)$ 的问题：

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

上述问题的一般形式：

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



$$x_1 - x_2 \leq 0,$$

$$x_1 - x_5 \leq -1,$$

$$x_2 - x_5 \leq 1,$$

$$x_3 - x_1 \leq 5,$$

$$x_4 - x_1 \leq 4,$$

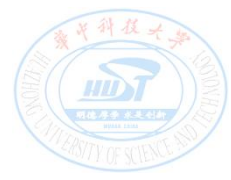
$$x_4 - x_3 \leq -1,$$

$$x_5 - x_3 \leq -3,$$

$$x_5 - x_4 \leq -3.$$

- 找一组满足上述约束条件的解。
- 问题可能的答案有： $x=(-5,-3,0,-1,-4)$ 、

$x'=(0,2,5,4,1)$ 等多组可行解。



这些解之间的一个基本关系是：

引理24.8 设向量 $x=(x_1, x_2, \dots, x_n)$ 为差分约束系统 $Ax \leq b$ 的一个解，设 d 为任意常数，则 $x+d=(x_1+d, x_2+d, \dots, x_n+d)$ 也是个该差分约束系统的一个解。

证明：

根据约束条件，对每对 x_i 和 x_j ， $(x_i+d)-(x_j+d)=x_i-x_j$ 。

因此若向量 x 满足 $Ax \leq b$ ，则向量 $x+d$ 也满足 $Ax \leq b$ 。 ■



差分约束系统的应用举例

未知变量 x_i 代表事件发生的时间，每个约束条件给出的是在两个时间之间必须间隔的最短时间。

比如，设这些事件是产品装配过程中的步骤：

如果在时刻 x_1 使用一种需要两个小时才能风干的粘贴剂材料，则下一个步骤需要2个小时后等粘贴剂干了之后才能在时刻 x_2 安装部件。这样就有约束条件 $x_2 \geq x_1 + 2$ ，亦即 $x_1 - x_2 \leq -2$ 等。



约束图:

在一个 $Ax \leq b$ 的差分约束系统中，将 $m \times n$ 的矩阵 A 看成是一张有 n 个结点和 m 条边构成的图的邻接矩阵的转置。

约束图定义如下:

对给定的差分约束系统 $Ax \leq b$ ，其对应的约束图是一个带权重的有向图 $G=(V,E)$ ，这里，

$$V = \{v_0, v_1, \dots, v_n\}$$

$$E = \{(v_i, v_j) : x_j - x_i \leq b_k \text{ 是一个约束条件}\}$$

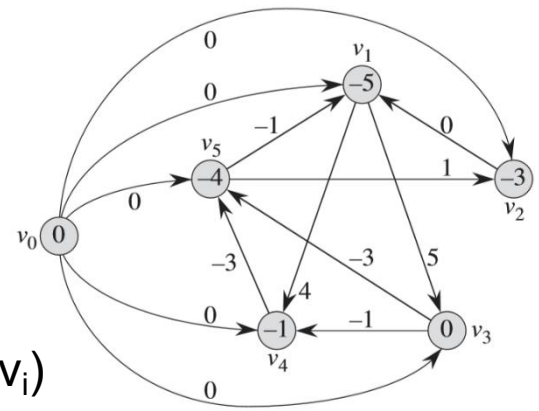
$$\cup \{(v_0, v_1), (v_0, v_2), (v_0, v_3), \dots, (v_0, v_n)\}$$

每条有向边对应一个不等式， (v_0, v_i) 是从新增的 v_0 到其他所有结点的边

说明:

- (1) **结点集合**: 约束图中引入一个额外的结点 v_0 , 从其出发可以达到其他所有结点。因此**结点集合 V** 由代表每个变量 x_i 的结点 v_i 和额外的结点 v_0 组成。
- (2) **边集合**: 边集合 E 包含代表每个差分约束的边, 同时包含 v_0 到其他所有结点的边 (v_0, v_i) , $i=1, 2, \dots, n$ 。
- (3) **边的权重**: 如果 $x_j - x_i \leq b_k$ 是一个差分约束条件, 则**边 (v_i, v_j)** 的权重记为 $\omega(v_i, v_j) = b_k$, 而从 v_0 出发到其他结点的边的权重 $\omega(v_0, v_j) = 0$ 。

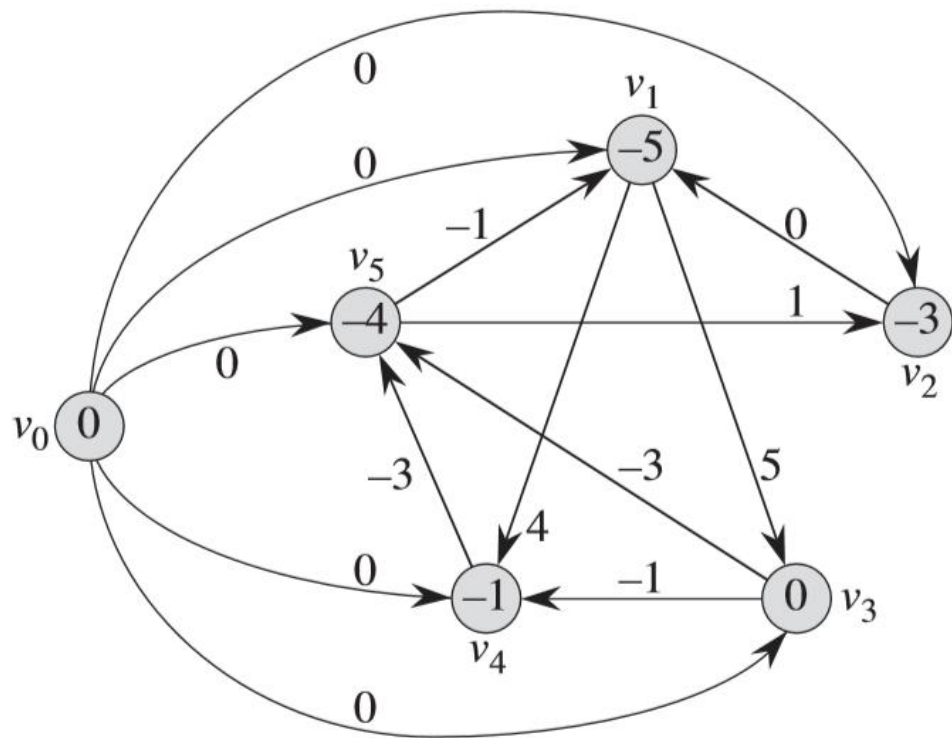
上例的差分约束系统的约束图如下:



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

结点中的数值是 $\delta(v_0, v_i)$

上例的差分约束系统的约束图如下：



$$x_1 - x_2 \leq 0,$$

$$x_1 - x_5 \leq -1,$$

$$x_2 - x_5 \leq 1,$$

$$x_3 - x_1 \leq 5,$$

$$x_4 - x_1 \leq 4,$$

$$x_4 - x_3 \leq -1,$$

$$x_5 - x_3 \leq -3,$$

$$x_5 - x_4 \leq -3.$$

- 结点集合V由代表每个变量 x_i 的结点 v_i 和额外的结点 v_0 组成
- 边集合E包含代表每个差分约束的边，同时包含 v_0 到其他所有结点的边 (v_0, v_i)
- 边 (v_i, v_j) 的权重记为 $\omega(v_i, v_j) = b_k$, $\omega(v_0, v_j) = 0$

定理24.9 给定差分约束系统 $Ax \leq b$, 设 $G=(V,E)$ 是该差分约束系统所对应的约束图。

(1) 如果图 G 不包含权重为负值的回路, 则

$$x = (\delta(v_0, v_1), \delta(v_0, v_2), \delta(v_0, v_3), \dots, \delta(v_0, v_n))$$

是该系统的一个可行解。

(2) 如果图 G 包含权重为负值的回路, 则该系统没有可行解。

证明: 考虑任意一条边 $(v_i, v_j) \in E$, 根据三角不等式有:

$$\delta(v_0, v_j) \leq \delta(v_0, v_i) + w(v_i, v_j)$$

即:

$$\delta(v_0, v_j) - \delta(v_0, v_i) \leq w(v_i, v_j)$$

因此, 令 $x_i = \delta(v_0, v_i)$, $x_j = \delta(v_0, v_j)$ 则 x_i 和 x_j 满足对应边 (v_i, v_j) 的差分约束条件 $x_j - x_i \leq w(v_i, v_j)$ 。

$$w(v_i, v_j) = b_k$$



因此, $x = (\delta(v_0, v_1), \delta(v_0, v_2), \delta(v_0, v_3), \dots, \delta(v_0, v_n))$

是问题的一个可行解。

(前提: 不包含权重为负的环路) 。

而如果约束图包含权重为负值的环路，不失一般性，设权重为负值的环路为 $c = \langle v_1, v_2, \dots, v_k \rangle$ ，这里 $v_1 = v_k$ 。

环路 c 对应下面的差分约束条件组：

$$\begin{aligned} x_2 - x_1 &\leq w(v_1, v_2), \\ x_3 - x_2 &\leq w(v_2, v_3), \\ &\vdots \\ x_{k-1} - x_{k-2} &\leq w(v_{k-2}, v_{k-1}), \\ x_k - x_{k-1} &\leq w(v_{k-1}, v_k). \end{aligned}$$

- 不等式左侧求和，等于0。（ $v_1 = v_k$ ，所有 v_i 相互抵消）
- 不等式右侧求和，等于环路 c 的权重 $\omega(c)$ ，且有： $0 \leq \omega(c)$
- 这与 c 是权重为负值的环路相矛盾。故该组不等式无解。

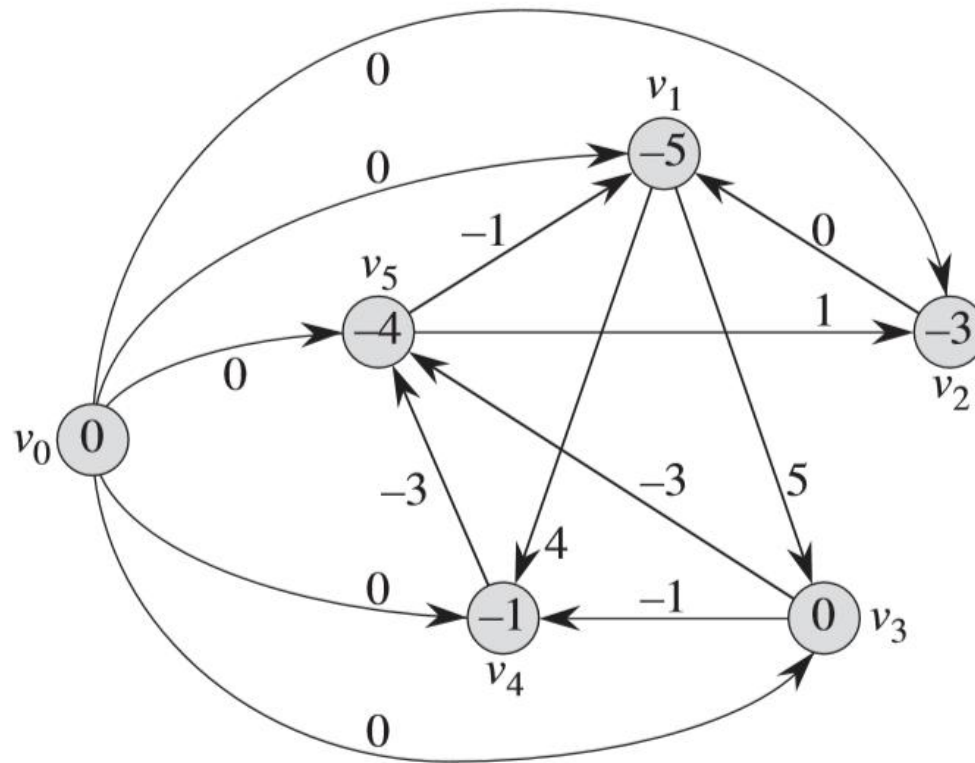


求解差分约束系统：

由定理24.9可得，可以使用Bellman-Ford算法来求解差分约束系统（思考为什么是Bellman-Ford算法？）。

约束图中含有从源结点 v_0 到其他所有结点的边,若存在权重为负值的环路，则都可以从结点 v_0 到达。则，

- 如果Bellman-Ford算法返回TRUE，则最短路径权重 $\delta(v_0, v_i), i=1, 2, \dots, n$, 给出该系统的一个可行解。
- 如果算法返回FALSE，则该系统无解。



- 结点中的数值是 $\delta(v_0, v_i)$;
- 该例的一个最短路径权重提供的可行解是: $x=(-5, -3, 0, -1, -4)$;
- 同时, 对于任意常数 d , $(d-5, d-3, d, d-1, d-4)$ 也是问题的解 (引理24.8)。

课堂练习

- 1、请画出下面的差分约束系统的约束图。并回答如何利用约束图求一个差分约束系统的可行解或判定该系统没有可行解。

$$x_1 - x_2 \leq 1$$

$$x_1 - x_3 \leq -4$$

$$x_2 - x_4 \leq 7$$

$$x_3 - x_2 \leq 2$$

$$x_4 - x_1 \leq -1$$

$$x_4 - x_3 \leq 3$$



课堂练习

- 1、请画出下面的差分约束系统的约束图。并回答如何利用约束图求一个差分约束系统的可行解或判定该系统没有可行解。

$$x_1 - x_2 \leq 1$$

$$x_1 - x_3 \leq -4$$

$$x_2 - x_4 \leq 7$$

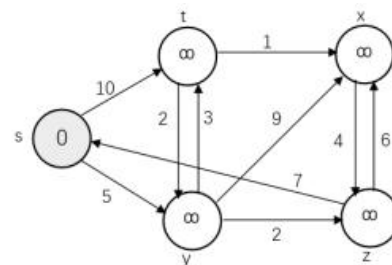
$$x_3 - x_2 \leq 2$$

$$x_4 - x_1 \leq -1$$

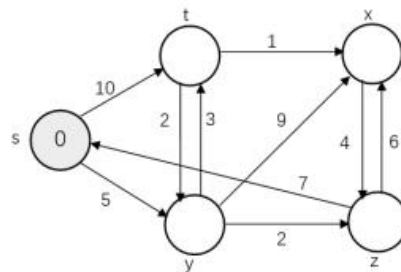
$$x_4 - x_3 \leq 3$$

课堂练习

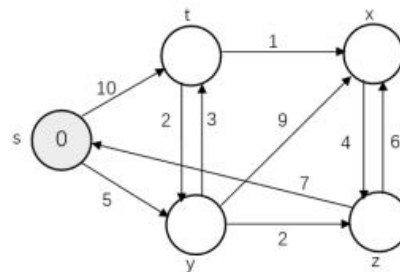
2、已知有向图如下面图(a)所示。(1) 执行Dijkstra算法求从结点s到其它各个结点的最短路径。请在图(b)~图(e)中的各个结点内填写算法第一次至第四次松弛操作后各结点的d值，并推导各结点的前驱结点。(2) 如果重置 $\omega(x,z)=-4$ ，Dijkstra算法还能正确执行吗？为什么？这种情况下用什么算法可以正确求出s到其它各个结点的最短路径？至少写出一个经典算法的名字或其基本思想。



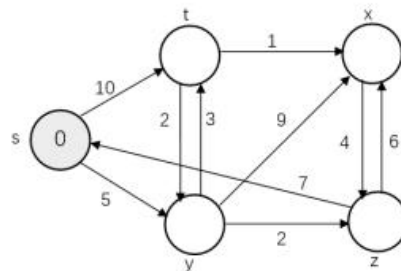
图(a)



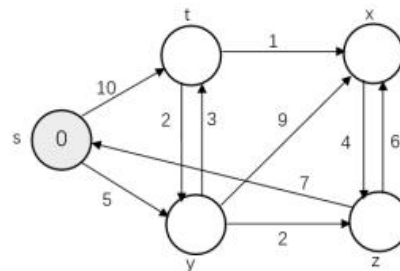
图(b)



图(c)



图(d)



图(e)

前驱结点：t. π =_____，x. π =_____，
y. π =_____，z. π =_____。