

# 算法设计与分析

## 第八章 图论算法

哈尔滨工业大学（深圳）

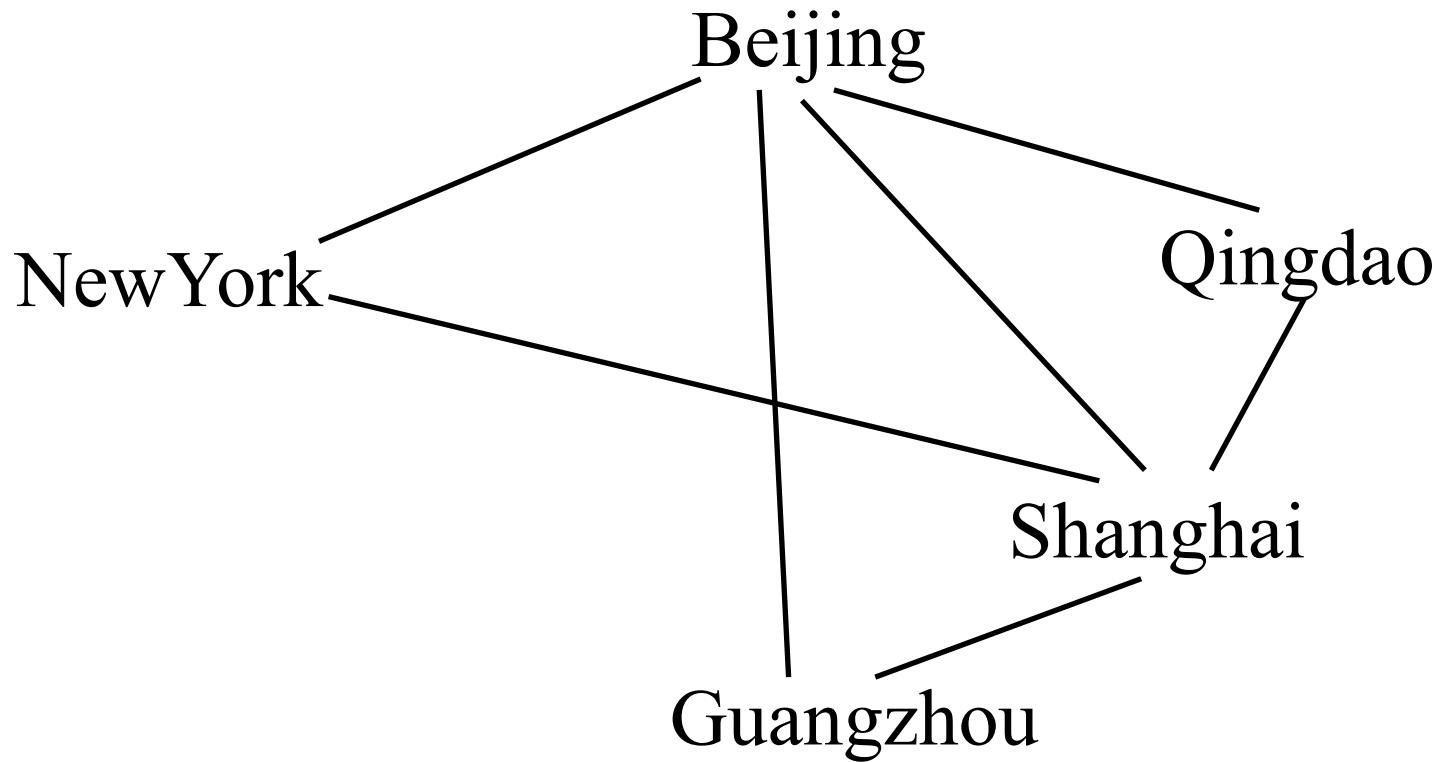
李穆

# 本讲内容

8.1 最短路径问题

8.2 网络流问题

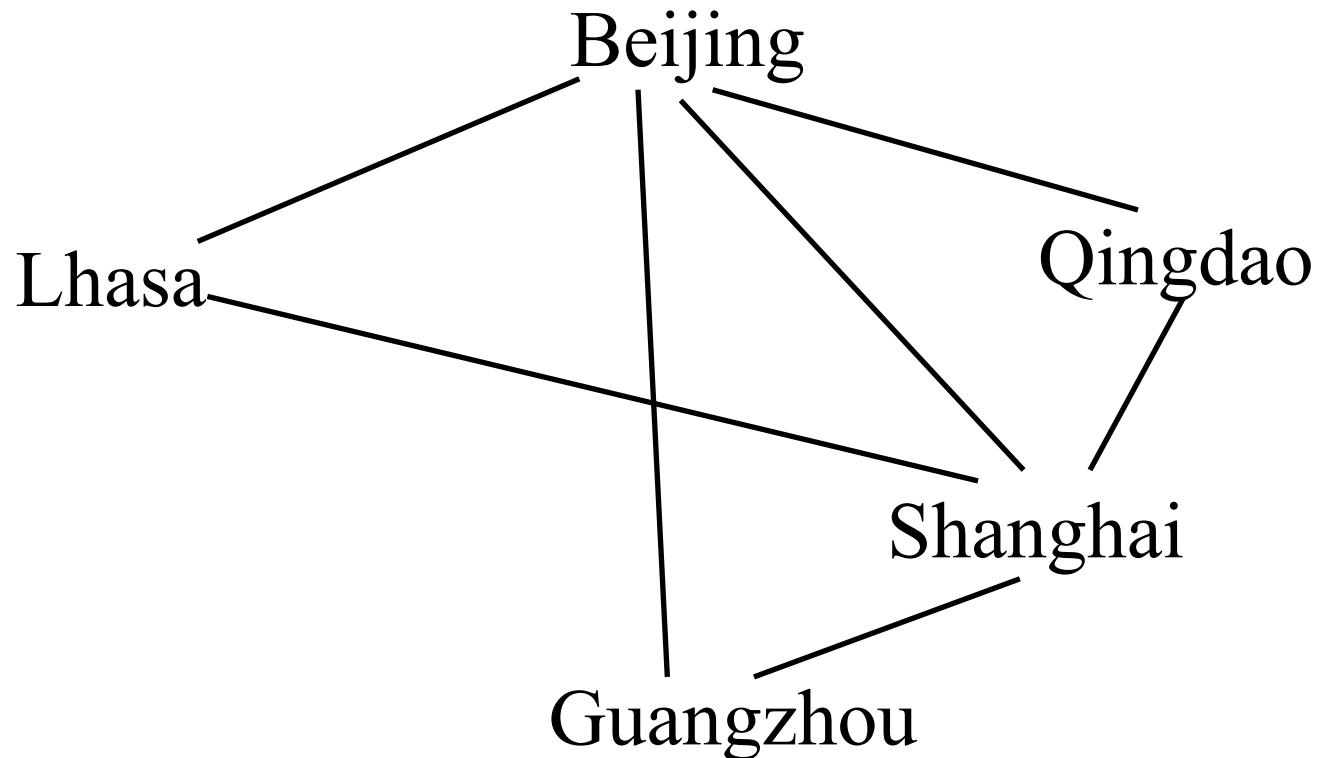
8.3 匹配问题



小明住在广州，他希望跟住在纽约的朋友一块玩游戏。假如他的网络供应商能够让他自己选择线路与他的朋友建立连接，他该如何选择让网络延迟最低。如图是供应商的节点和线路。

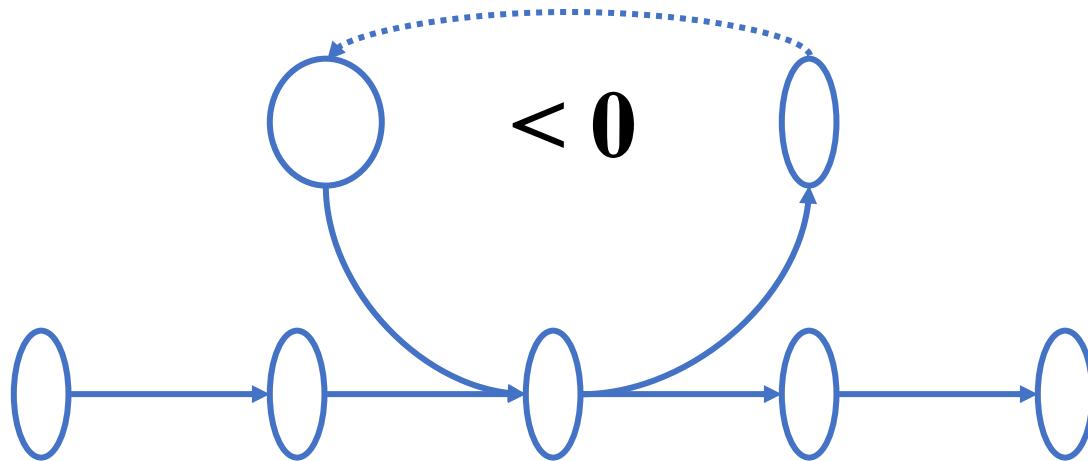
# 单源最短路径

- 问题：给定一个边加权的有向图 $G = (V, E)$ ，找到从给定源结点 $u$ 到另一个结点 $v$ 的最短路径。



# 有没有最短路径

- 如果图中包含负圈，某些最短路径可能不存在 (为什么？)：

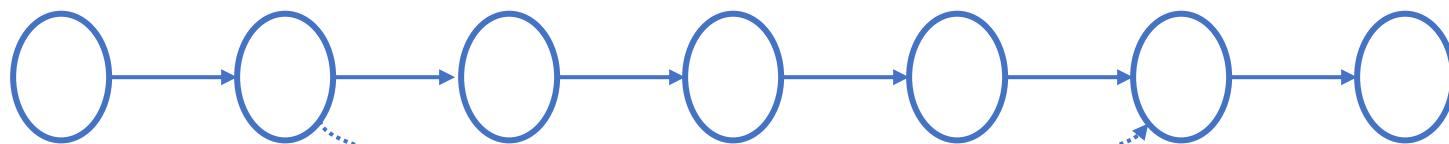


# 动态规划算法设计步骤

- 分析优化解的结构
- 递归地定义最优值的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优值的信息
- 根据构造最优值的信息构造优化解

# 最短路径的性质

- **优化子结构:** 两个结点之间的一条最短路径包含着其他的最短子路径



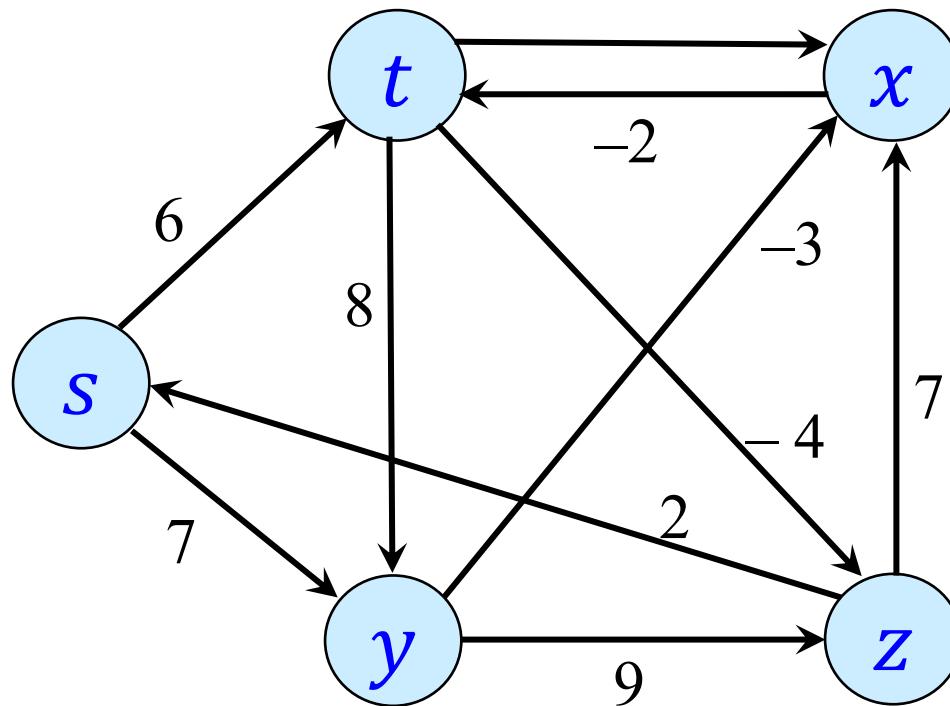
- 证明: (反证法) 如果某条子路径不是最短子路径
  - 必然存在最短子路径
  - 用最短子路径替换当前子路径
  - 当前路径不是最短路径, 矛盾!

# 动态规划

- 重叠子问题:

1)  $x \rightarrow t \rightarrow y$

2)  $x \rightarrow y$



# 动态规划

- 递归定义最优解代价:
- $d[k][v]$  表示有  $k$  条边的最短路径的代价
- $d[k][v] = \min_{(u,v) \in E} \{d[k-1][u] + w(u, v)\}$
- 扫描顺序:  $k = 1$  to  $|V|-1$

# 动态规划算法

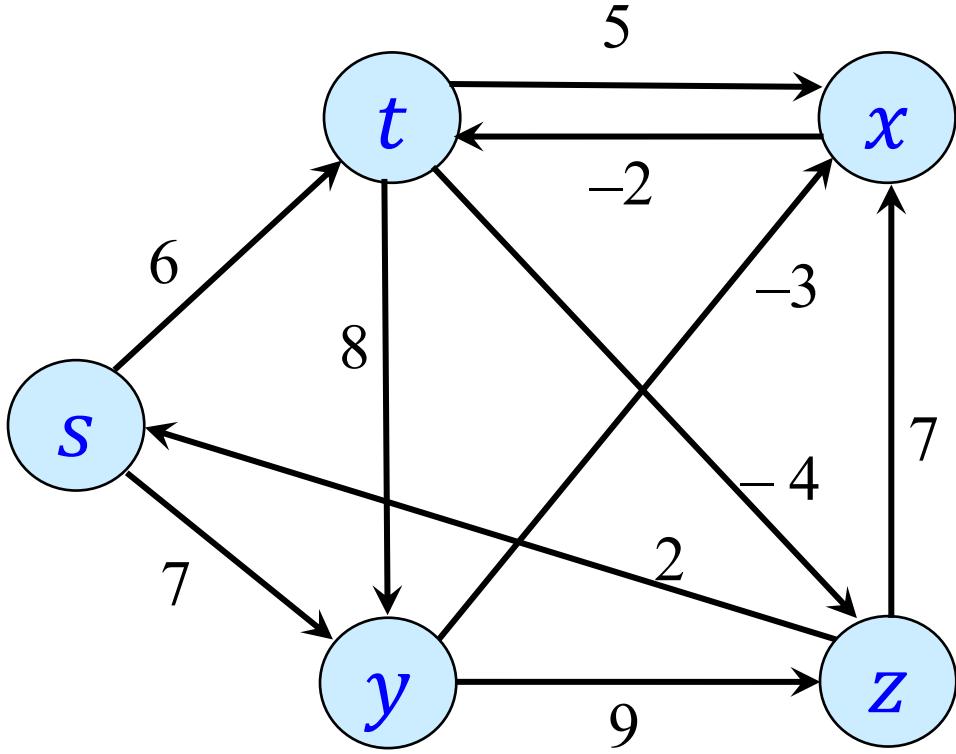
ShortestPath(G,s,w)

1. For each  $v \in V$  Do
2. For  $k = 1$  To  $|V|-1$  Do
3.      $d[k][v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
4.      $d[0][s] \leftarrow 0$ ;
5. For  $k = 1$  To  $|V|-1$  Do
6.     For each node  $v \in V$  Do
7.         For each edge  $(u,v) \in v.\text{adj}()$  Do
8.              $d[k][v] = \min(d[k][v], d[k-1][u] + w(u,v))$ ;

# 动态规划算法

ShortestPath(G,s,w)

1. For each  $v \in V$  Do
2. For  $k = 1$  To  $|V|-1$  Do
3.      $d[k][v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
4.      $d[0][s] \leftarrow 0$ ;
5. For  $k = 1$  To  $|V|-1$  Do
6.     For each edge  $(u,v) \in E$  Do
7.          $d[k][v] = \min(d[k][v], d[k-1][u] + w(u, v))$ ;
8.     For each  $v \in V$  Do
9.          $dt[v] \leftarrow \infty$ ;
10.    For  $k = 1$  To  $|V|-1$  Do
11.      If  $d[k][v] < dt[v]$  Then  $dt[v] \leftarrow d[k][v]$ ;



$$d[s] = d[z] + 2$$

$$d[t] = \begin{cases} d[s] + 6 \\ d[x] - 2 \end{cases}$$

$$d[x] = \begin{cases} d[t] + 5 \\ d[y] - 3 \\ d[z] + 7 \end{cases}$$

$$d[y] = \begin{cases} d[s] + 7 \\ d[t] + 8 \end{cases}$$

$$d[z] = \begin{cases} d[y] + 9 \\ d[t] - 4 \end{cases}$$

$d[s]$	$d[t]$	$d[x]$	$d[y]$	$d[z]$
0	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	6	$\infty$	7	$\infty$
$\infty$	$\infty$	4	14	2
4	2	9	$\infty$	23
25	7	7	10	-2

# 动态规划

- 递归定义最优解代价:
- $d[k][v]$  表示至多有  $k$  条边的最短路径的代价

$$\bullet d[k][v] = \min \begin{cases} \min_{(u,v) \in E} \{d[k-1][u] + w(u,v)\} \\ d[k-1][v] \end{cases}$$

- 扫描顺序:  $k = 1$  to  $|V|-1$

# 动态规划算法

ShortestPath(G,s,w)

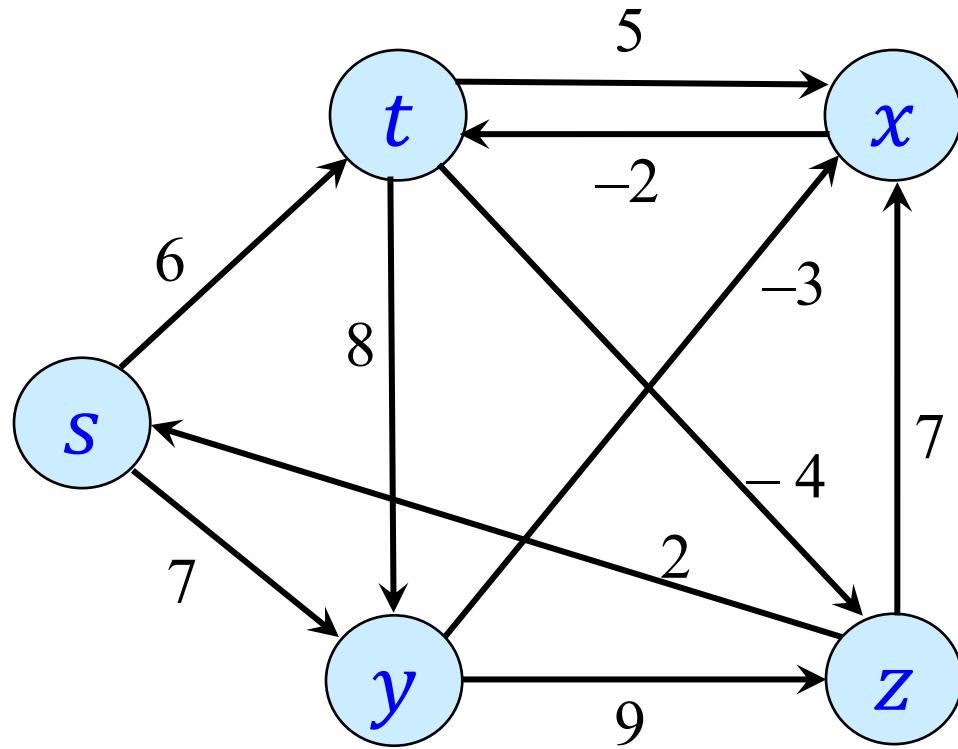
1. For each  $v \in V$  Do
2.      $d[0][v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
3.      $d[0][s] \leftarrow 0$ ;
4. For  $k = 1$  To  $|V|-1$  Do
5.     For each  $v \in V$  Do  $d[k][v] \leftarrow d[k-1][v]$ ;
6.     For each edge  $(u,v) \in E$  Do
7.          $d[k][v] = \min(d[k][v], d[k-1][u] + w(u,v))$ ;

有负环怎么办？

# 动态规划算法

ShortestPath(G,s,w)

1. For each  $v \in V$  Do
2.      $d[0][v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
3.      $d[0][s] \leftarrow 0$ ;
4. For  $k = 1$  To  $|V|$  Do
5.     For each  $v \in V$  Do  $d[k][v] \leftarrow d[k-1][v]$ ;
6.     For each edge  $(u,v) \in E$  Do
7.          $d[k][v] = \min(d[k][v], d[k-1][u] + w(u, v))$ ;
8. For each  $v \in V$  Do
9.     If ( $d[|V|][v] < d[|V|-1][v]$ ) Then
10.         Return “no solution”;
11.         Return True



$$d[s] = d[z] + 2$$

$$d[t] = \begin{cases} d[s] + 6 \\ d[x] - 2 \end{cases}$$

$$d[x] = \begin{cases} d[t] + 5 \\ d[y] - 3 \\ d[z] + 7 \end{cases}$$

$$d[y] = \begin{cases} d[s] + 7 \\ d[t] + 8 \end{cases}$$

$$d[z] = \begin{cases} d[y] + 9 \\ d[t] - 4 \end{cases}$$

$d[s]$	$d[t]$	$d[x]$	$d[y]$	$d[z]$
0	$\infty$	$\infty$	$\infty$	$\infty$
0	$6(s)$	$\infty$	$7(s)$	$\infty$
0	$6(-)$	$4(y)$	$7(-)$	$2(t)$
0	$2(x)$	$4(-)$	$7(-)$	$2(-)$
0	$2(-)$	$4(-)$	$7(-)$	$-2(t)$
0	2	4	7	-2

# 动态规划算法的正确性

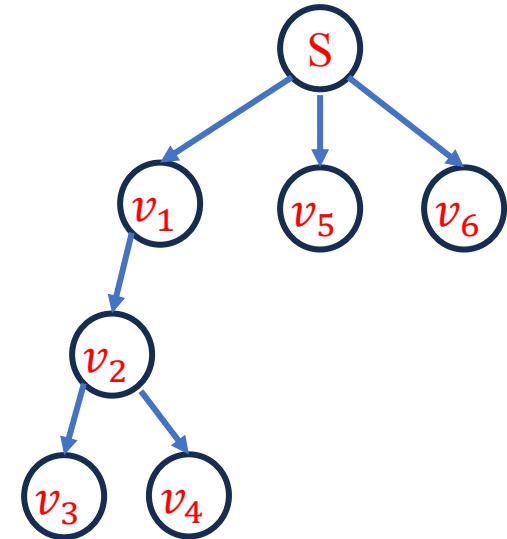
**定理.** 给定一个边加权的有向图  $G = (V, E)$ , 其源结点  $s$  且不包含权重为负值的环路, 那么 ShortestPath 算法的 for 循环执行了  $|V| - 1$  次后, 有  $d$  的值都是正确的。

**证明:**

- 考虑从源结点  $s$  到结点  $t$  的最短路径:

$$s \rightarrow v_1 \rightarrow v_2 \cdots \rightarrow t$$

- 开始,  $d[0][s] = 0$  正确, 不发生变化
- 1轮之后,  $d[1][v_1]$  正确, 不发生变化
- 2轮之后,  $d[2][v_2]$  正确, 不发生变化
- ...
- $|V| - 1$  轮之后停下。



还能不能优化?

- 必须执行  $|V|$  次吗?
- 能不能压缩一下存储空间?

# 动态规划算法

ShortestPath(G,s,w)

1. For each  $v \in V$  Do  $d[0][v] \leftarrow \infty$ ;
2.  $d[0][s] \leftarrow 0$ ;
3. For  $k = 1$  To  $|V|$  Do
4.      $\text{count} \leftarrow 0$ ;
5.     For each  $v \in V$  Do  $d[k][v] \leftarrow d[k-1][v]$ ;
6.     For each edge  $(u,v) \in E$  Do
7.         If  $d[k-1][u] + w(u,v) < d[k][v]$  Then
8.              $d[k][v] = d[k-1][u] + w(u,v)$ ;
9.              $\text{count} \leftarrow \text{count} + 1$
10.     If  $\text{count} = 0$  Then break
11.     If ( $\text{count} > 0$ ) Then     Return “no solution”;
12.     Return True

# ShortestPath<sup>算法</sup>

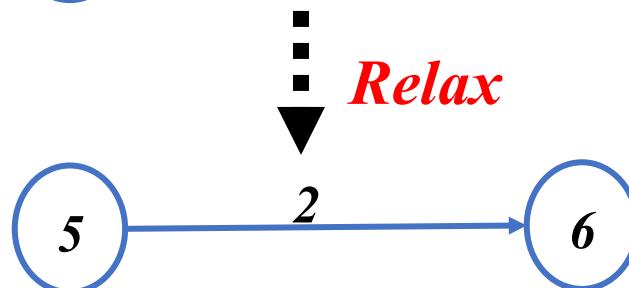
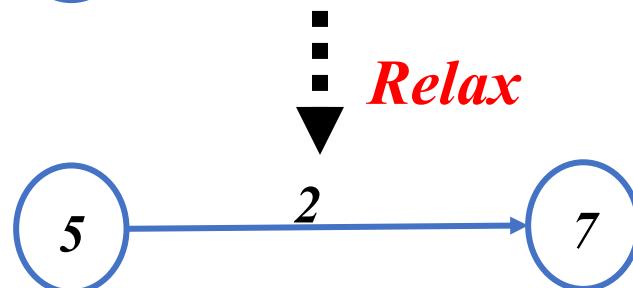
ShortestPath(G,s,w)

1. For each  $v \in V$  Do  $d[0][v] \leftarrow \infty;$
2.  $d[0][s] \leftarrow 0;$
3. For  $i = 1$  To  $|V|$  Do  $d[v] = \min(d[v], d[u] + w(u, v));$
4. For each  $v \in V$  Do  $d[k][v] \leftarrow d[k-1][v];$
5. For each edge  $(u, v) \in E$  Do  $d[k][v] = \min(d[k][v], d[k - 1][u] + w(u, v));$
7. For each  $v \in V$  Do
8. If ( $d[|V|][v] < d[|V|-1][v]$ ) Then
9.     Return “no solution”;
10. Return True

# 松弛技术

- 最短路径算法的核心技术是松弛
- 以  $w(u, v)$  表示顶点  $u$  出发到顶点  $v$  的边的权值，以  $d[v]$  表示当前从起点  $s$  到顶点  $v$  的路径权值

```
1. Relax(u,v,w(u,v)) {  
2.     If (d[v] > d[u] + w(u,v)) Then  
3.         d[v] = d[u] + w(u,v);  
4. }
```



# Bellman-Ford 算法

Bellman-Ford( $G, s, w$ )

1. For each  $v \in V$  Do
2.      $d[v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
3.      $d[s] \leftarrow 0$ ;
4. For  $i = 1$  To  $|V|-1$  Do
5.     For each edge  $(u, v) \in E$  Do
6.         Relax( $u, v, w(u, v)$ );
7.     For each edge  $(u, v) \in E$  Do
8.         If ( $d[v] > d[u] + w(u, v)$ ) Then
9.             Return “no solution”;
10.    Return True

# Bellman-Ford 算法的正确性

**定理.** 给定一个边加权的有向图  $G = (V, E)$ ，其源结点  $s$  且不包含权重为负值的环路，那么 Bellman-Ford 算法的 for 循环，在执行了  $|V| - 1$  次后，有  $d$  的值都是正确的。

**证明：**要证明 Bellman-Ford 算法每次迭代后得到的  $d$  值都不大于 ShortestPath 算法。也即， $d^k[v] \leq d[k][v]$ 。

经过  $|V| - 1$  迭代，改进前的 ShortestPath 算法收敛到最小值， $d[|V| - 1][v]$  即是  $s$  到  $v$  的最短路径。

对于 Bellman-Ford 算法，有  $d^{|V|-1}[v] \leq d[|V| - 1][v]$ ，那么， $d^{|V|-1}[v]$  必然也是  $s$  到  $v$  的最短路径。

- $n=0$  的时候显然成立， $d^0[v] = d[0][v]$ 。

# Bellman-Ford 算法的正确性

**定理.** 给定一个边加权的有向图  $G = (V, E)$ , 其源结点  $s$  且不包含权重为负值的环路, 那么改进的Bellman-Ford算法的for循环, 在执行了  $|V| - 1$  次后, 有  $d$  的值都是正确的。

**证明:**

- 假设  $n < k$  时成立 ( $d^n[v] \leq d[n][v]$ ), 要证  $n = k$  时, 结论也成立。
- 考虑节点  $v$ , 和所有与  $v$  相连的边  $uv$
- 改进后  $d[v] = \min_{(u,v) \in E} \{d[u] + w(u, v), d[v]\}$
- 改进前  $d[k][v] = \min_{(u,v) \in E} \{d[k-1][u] + w(u, v), d[k][v]\}$
- 观察  $d[u]$ , 它要么还未更新, 是第  $k-1$  次迭代后的值  $d^{k-1}[u]$ , 要么已经更新, 根据更新规则, 它比  $d^{k-1}[u]$  要小。总之,  $d[u] \leq d^{k-1}[u] \leq d[k-1][u]$
- 所以, 第  $k$  次更新后, 依然有  $d[v] \leq d[k][v]$ , 也即  $d^k[v] \leq d[k][v]$

# Bellman-Ford 算法

Bellman-Ford( $G, s, w$ )

1. For each  $v \in V$  Do } 初始化  $d$   $O(|V||E|)$
2.    $d[v] \leftarrow \infty;$
3.    $d[s] \leftarrow 0;$
4. For  $i = 1$  To  $|V|-1$  Do } 松弛:  
5.   For each edge  $(u, v) \in E$  Do } 进行  $|V| - 1$  轮,  
6.     Relax( $u, v, w(u, v)$ ); } 松弛每条边  
7.   For each edge  $(u, v) \in E$  Do } 检验结果  
8.     If ( $d[v] > d[u] + w(u, v)$ ) Then

**$Relax(u, v, w)$ :**

**$if (d[v] > d[u] + w(u, v)) then d[v] = d[u] + w(u, v)$**

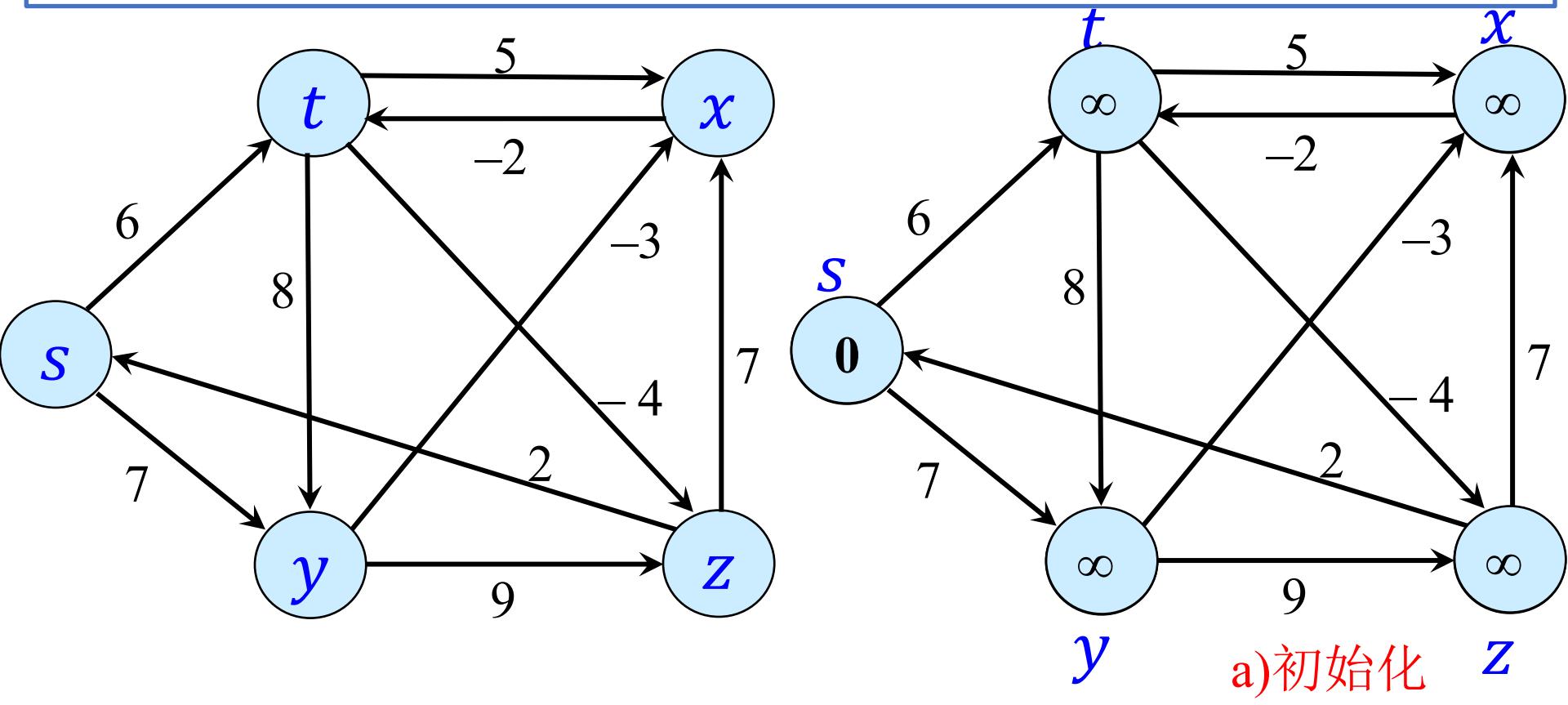
# Bellman-Ford 算法

Bellman-ford算法的执行过程。

源点是顶点  $s$ 。  $s$  到各个顶点  $v$  的值  $d[v]$  被标记在各自顶点内，加粗的边指示了前趋边，每一趟按照如下顺序对边进行松弛：

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y),$

- a) 对边进行第一趟操作的结果。
- b) 至 e) 每一趟连续对边操作的结果，
- e) 是最终结果。

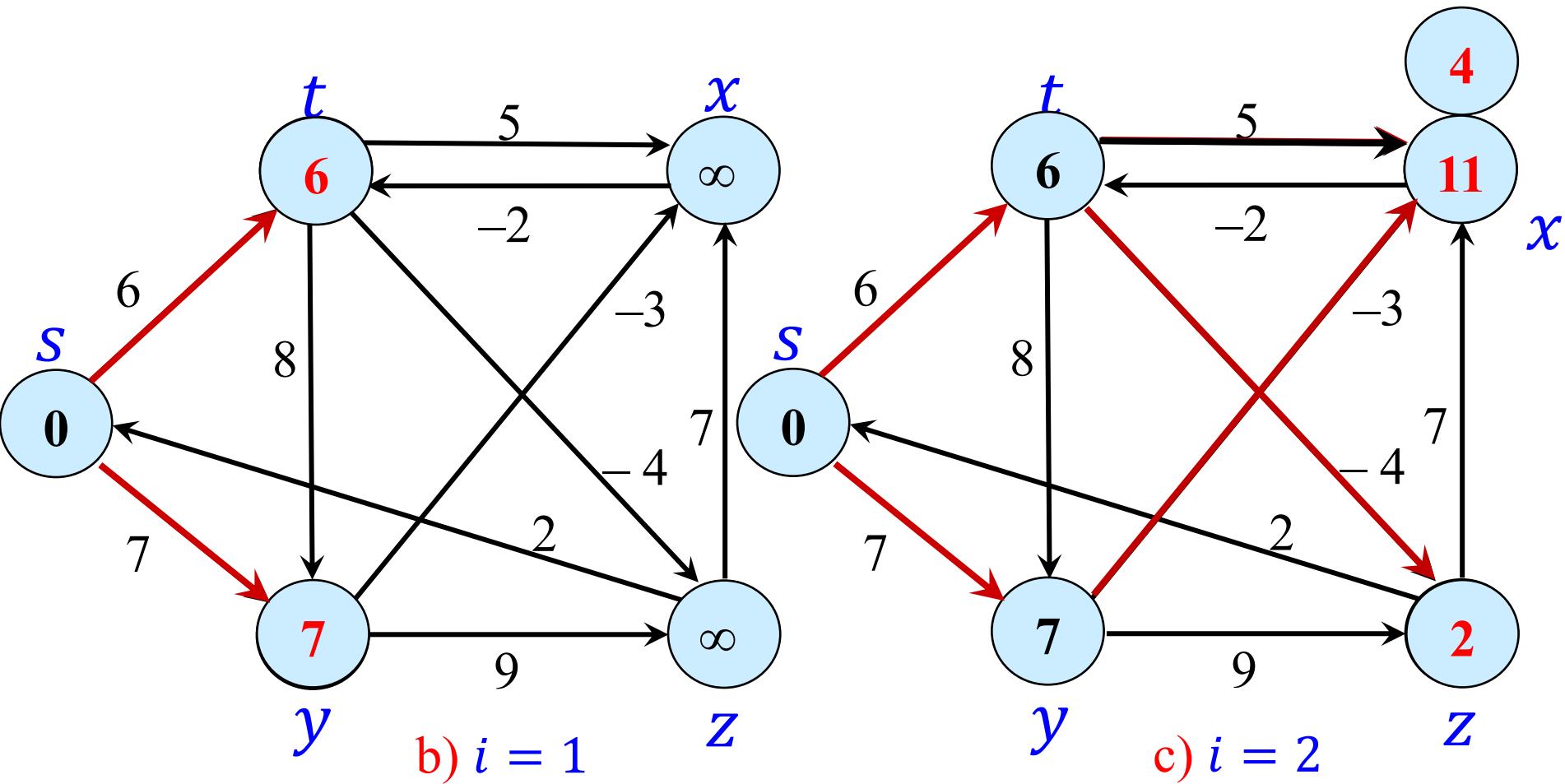


# Bellman-Ford 算法

源点是顶点  $s$ 。  $s$  到各个顶点  $v$  的值  $d[v]$  被标记在各自顶点内，加粗的边指示了前趋边，每一趟按照如下顺序对边进行松弛：

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y),$

a) 对边进行第一趟操作的结果。b) 至e) 每一趟连续对边操作的结果，e) 是最终结果。



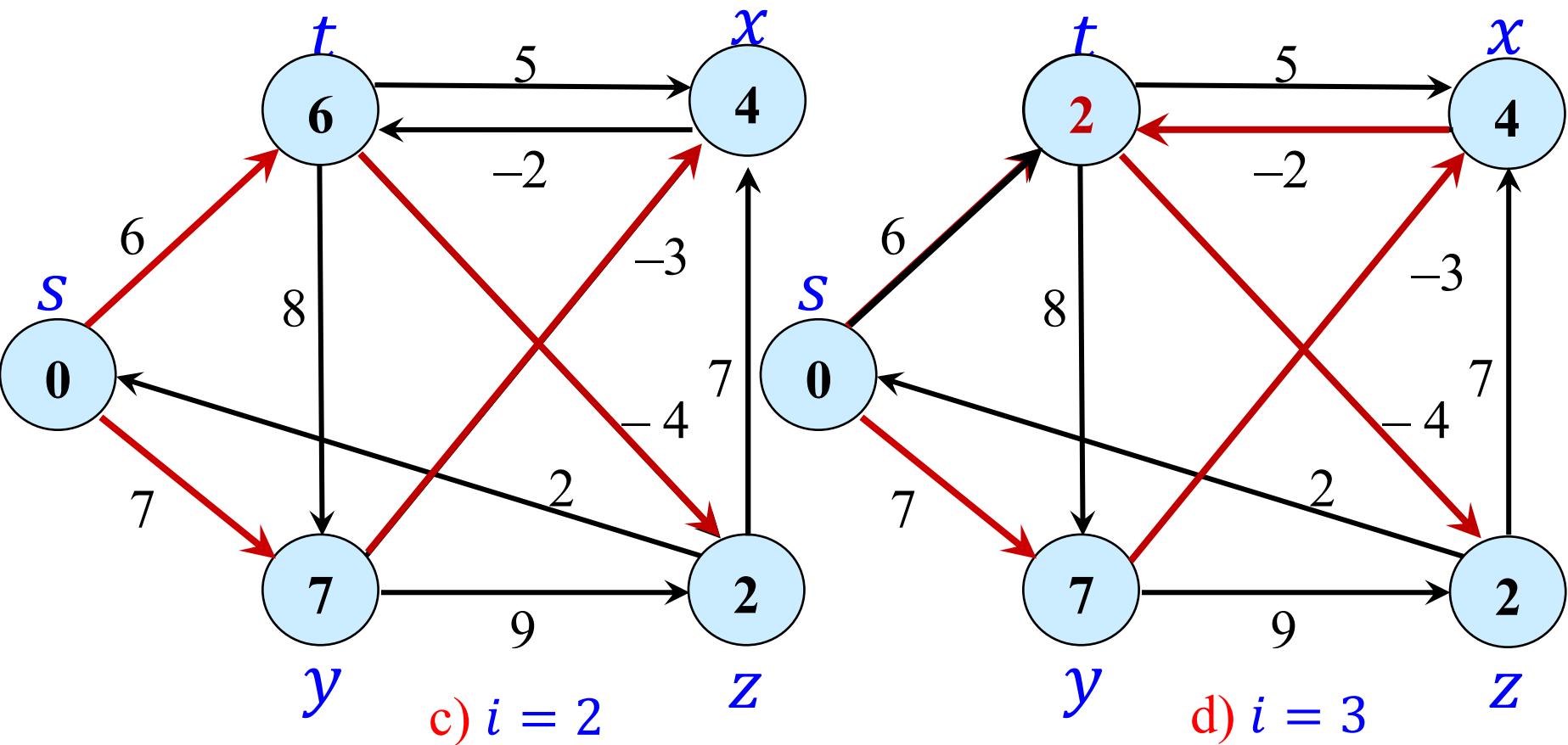
$Relax(u, v, w)$ : if  $(d[v] > d[u] + w(u, v))$  then  $d[v] = d[u] + w(u, v)$

# Bellman-Ford 算法

源点是顶点  $s$ 。  $s$  到各个顶点  $v$  的值  $d[v]$  被标记在各自顶点内，加粗的边指示了前趋边，每一趟按照如下顺序对边进行松弛：

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$ ,

a) 对边进行第一趟操作的结果。b) 至e) 每一趟连续对边操作的结果，e) 是最终结果。



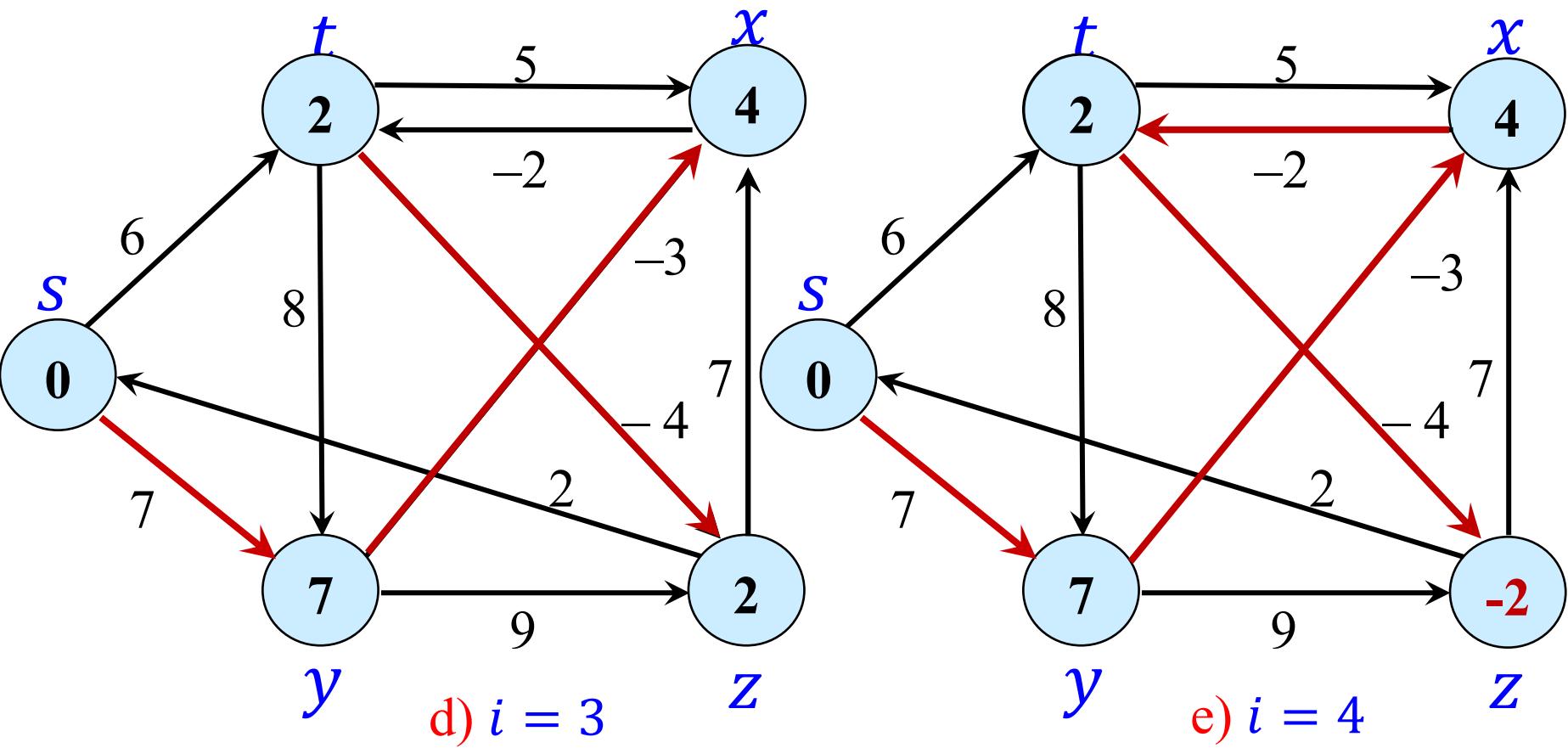
$\text{Relax}(u, v, w)$ : if  $(d[v] > d[u] + w(u, v))$  then  $d[v] = d[u] + w(u, v)$

# Bellman-Ford 算法

源点是顶点  $s$ 。  $s$  到各个顶点  $v$  的值  $d[v]$  被标记在各自顶点内，加粗的边指示了前趋边，每一趟按照如下顺序对边进行松弛：

$(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$ ,

a) 对边进行第一趟操作的结果。b) 至e) 每一趟连续对边操作的结果，e) 是最终结果。



$\text{Relax}(u, v, w)$ : if  $(d[v] > d[u] + w(u, v))$  then  $d[v] = d[u] + w(u, v)$

# 有向无环图中最短路径

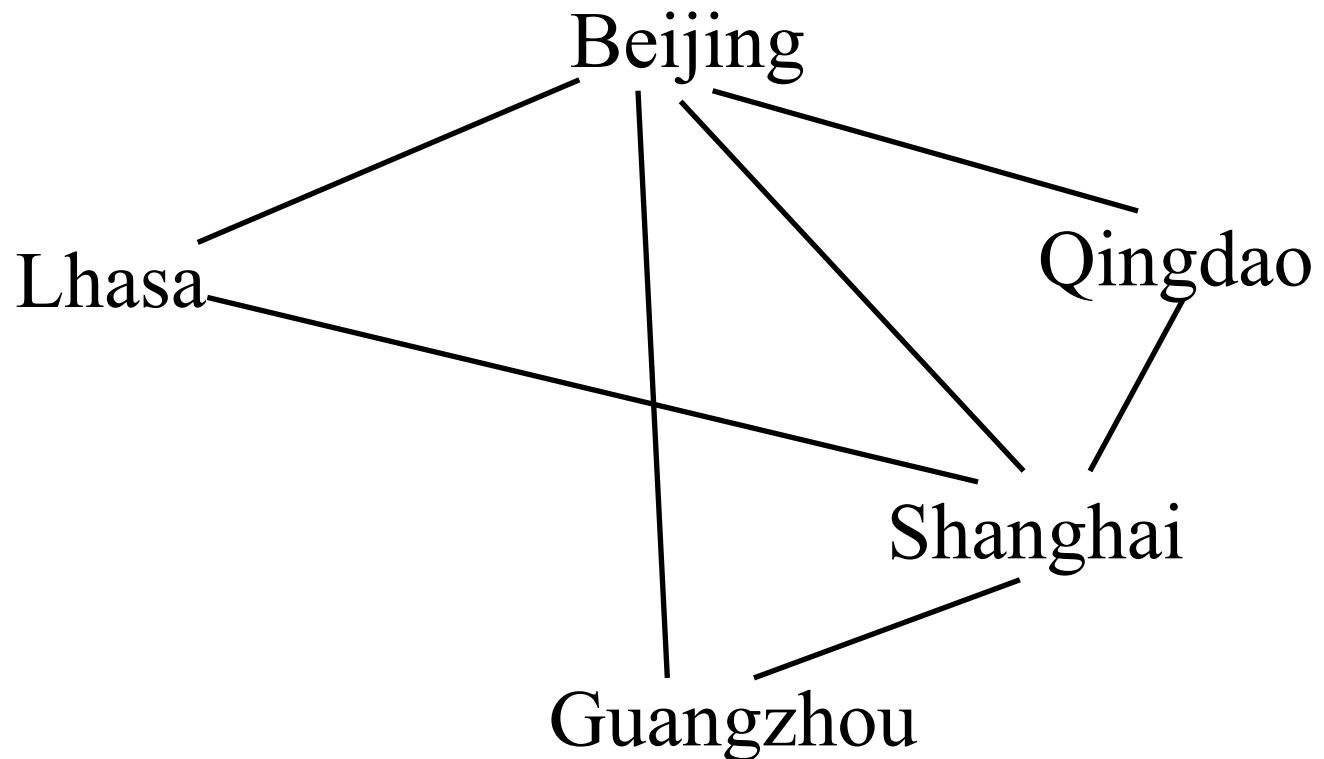
- 问题: 寻找加权有向无环图 (Directed Acyclic Graph, DAG) 中的单源最短路径
  - 在有向无环图中, 即使存在权重为负值的边, 但因为没有权重为负的环路, 最短路径也是存在的。
  - Bellman-Ford 时间是  $O(VE)$ 。
  - 能否做的好一点呢? (贪心思想)
- 思路: 使用拓扑排序 (按照有向图给出的次序关系, 将图中顶点排成一个线性序列, 对于有向图中没有限定次序关系的顶点, 则可以认为加上任意的次序关系。)

《算法导论》P383有一个算法

- 如果沿着最短路径, 则可以一遍完成
- DAG中的每条路径都是通过拓扑排序得到的结点序列的一个子序列, 那么, 如果按照这个顺序处理, 我们将沿着路径进行处理。

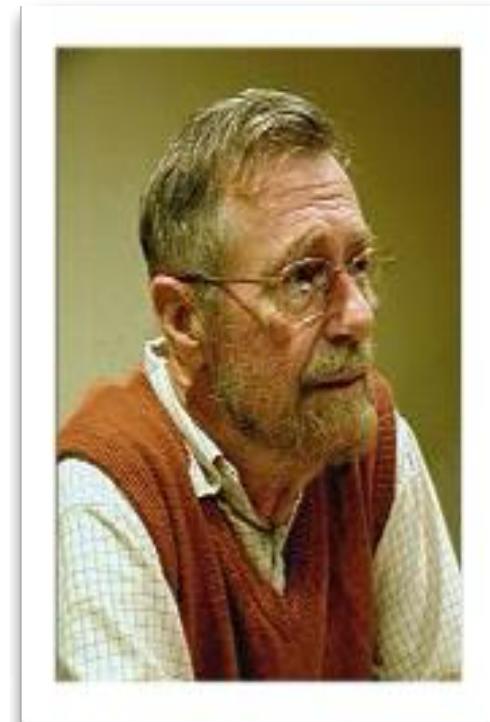
# 单源最短路径

- 问题：给定一个边加权的有向图 $G = (V, E)$ ，找到从给定源结点 $u$ 到另一个结点 $v$ 的最短路径。
- 没有负边



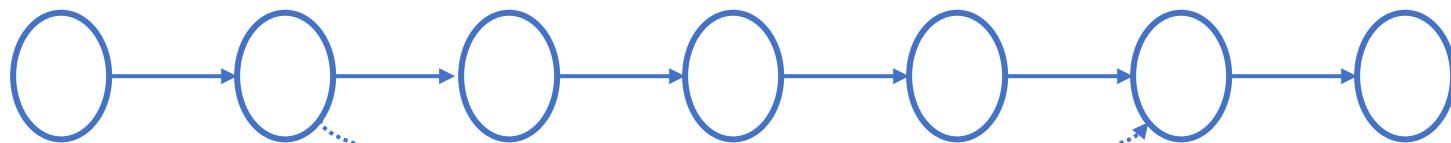
# Dijkstra 算法

- Dijkstra算法解决的是带权重的有向图上的单源最短路径问题，要求所有边的权重为非负的 $w(u, v) \geq 0$
- 如果图中没有负边，Dijkstra算法可以超越Bellman-Ford 算法
- 类似Best-First 搜索
  - 从队列中取结点
- 类似Prim算法
  - 使用以 $d[v]$ 为键的优先队列



# 最短路径的性质

- **优化子结构:** 两个结点之间的一条最短路径包含着其他的最短子路径



- 证明: (反证法) 如果某条子路径不是最短子路径
  - 必然存在最短子路径
  - 用最短子路径替换当前子路径
  - 当前路径不是最短路径, 矛盾!

# Dijkstra 算法的正确性

贪心选择性：图G里从顶点s出发的所有边里最短的一条为(s,v)，那么图G里顶点s到v的最短路径就是w(s,v)。

证明：反证法

假设w(s,v)不是图G里顶点s到v的最短路径，那么必然存在一条s->u->...->v的路径是s到v的最短路径

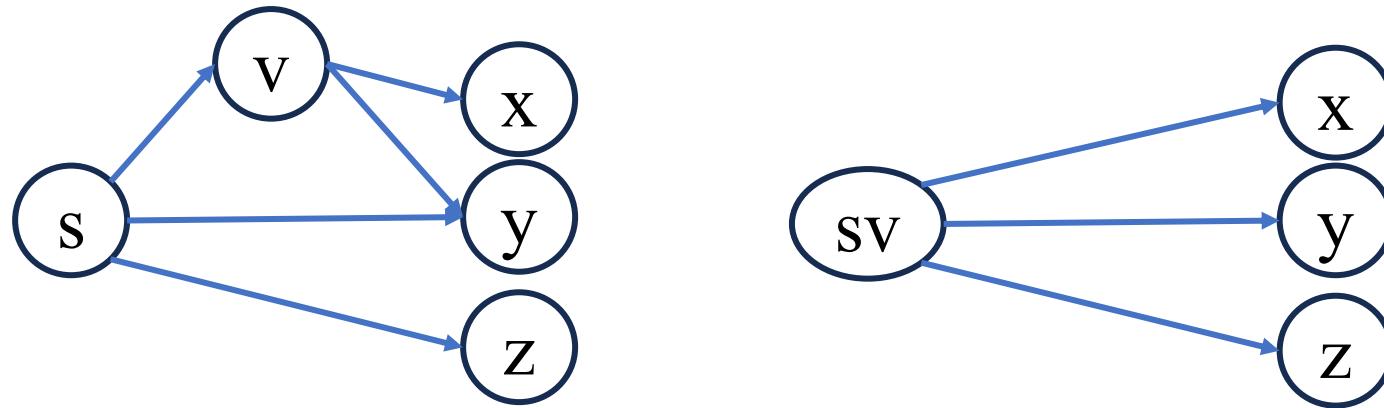
$$\begin{aligned}\delta(s, v) &= w(s, u) + \cdots + w(., v) \geq \\ w(s, u) &> w(s, v)\end{aligned}$$

新的路径比w(s,v)长，与新路径是最短路径矛盾！

# Dijkstra 算法的正确性

贪心选择性：图G里从顶点s出发的所有边里最短的一条为 $(s,v)$ ，那么图G里顶点s到v的最短路径就是 $w(s,v)$ 。

子问题：



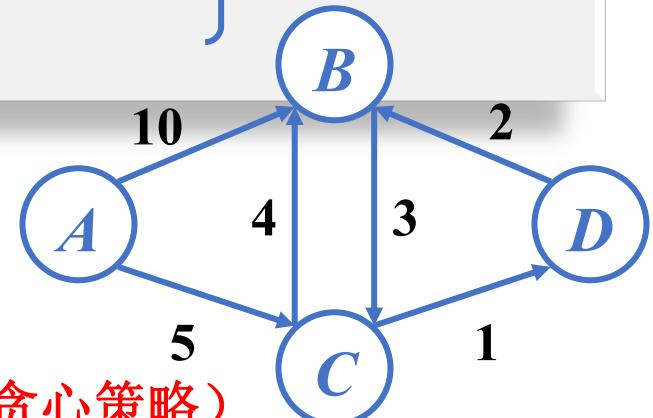
	x	y	z	v
s	$\infty$	$w(s,y)$	$w(s,z)$	$w(s,v)$
sv	$w(s,v)+w(v,x)$	$\min(w(s,y), w(s,v)+w(v,y))$	$w(s,z)$	

# Dijkstra 算法

Dijkstra( $G, s$ )

1. For each  $v \in V$  Do
2.    $d[v] \leftarrow \infty$ ; /\*从起点  $s$  到顶点  $v$  的路径权值\*/
3.    $d[s] \leftarrow 0$ ;  $S \leftarrow \emptyset$ ;  $Q \leftarrow V$ ; /\* $S$ 是已访问结点集合;  $Q$ 最小优先队列是未访问结点集合\*/
4. While ( $Q \neq \emptyset$ ) Do
5.    $u \leftarrow \text{Extract-Min}(Q)$ ; /\* which removes and returns the element with the smallest key from  $Q$  \*/
6.    $S \leftarrow S \cup \{u\}$ ;
7.   For  $v \in u \rightarrow \text{Adj}()$  Do /\*更新与u邻接的结点v\*/
8.     If ( $d[v] > d[u] + w(u, v)$ ) Then
9.        $d[v] \leftarrow d[u] + w(u, v)$ ;

松弛步骤



对比Bellman-Ford 算法

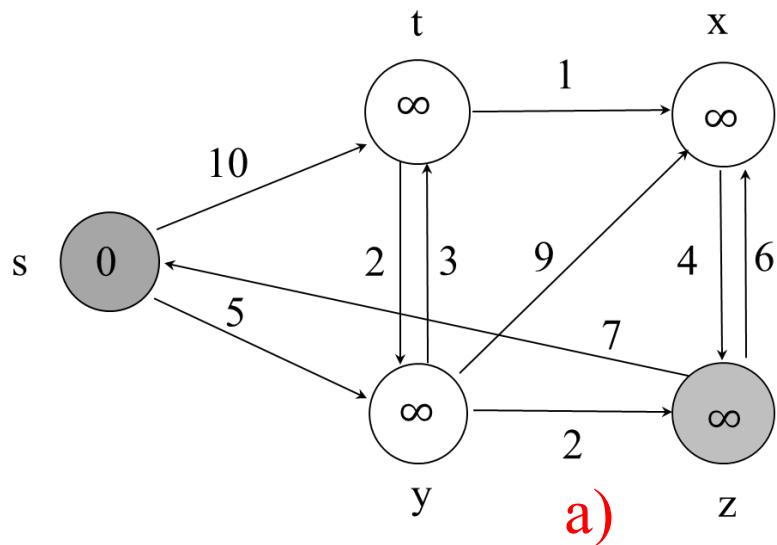
➤ Bellman-Ford 算法对所有的边松弛操作  $|V| - 1$  次

➤ Dijkstra算法对所有的边松弛操作一次且仅一次（贪心策略）

# Dijkstra 算法

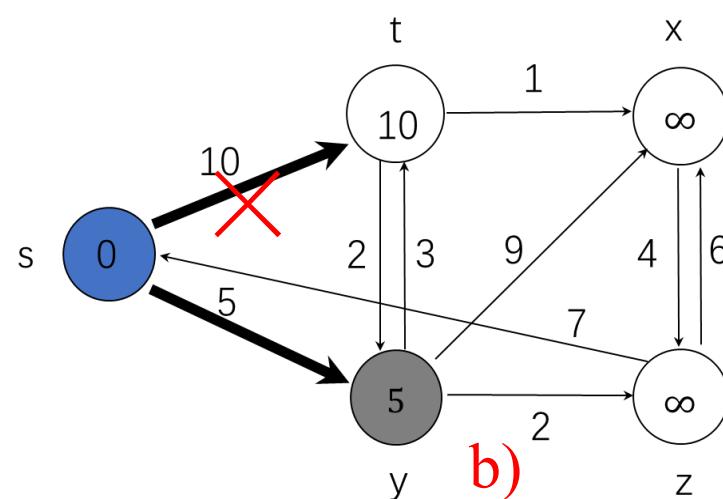
Dijkstra算法的执行过程：源点为 $s$ 。最短路径的估计被标记在顶点内，加粗的边为前趋边。蓝色顶点在集合 $S$ 中，而白色顶点在最小优先队列 $Q = V - S$ 中。

a) 第4~8行While循环第一次迭代前的情形。灰色顶点具有最小的 $d$ 的值，在第5行被选为顶点 $u$ 。b) 至 f) while循环在每一次连续迭代后的情形。每个图中灰色的顶点被选作下一次迭代第5行的顶点 $u$ 。



$$S = \emptyset$$

$$Q = \{s, t, x, y, z\}$$

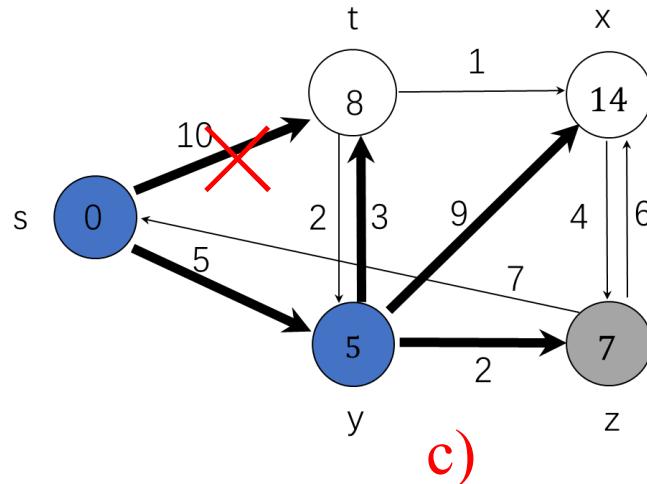


$$S = \{s\}$$

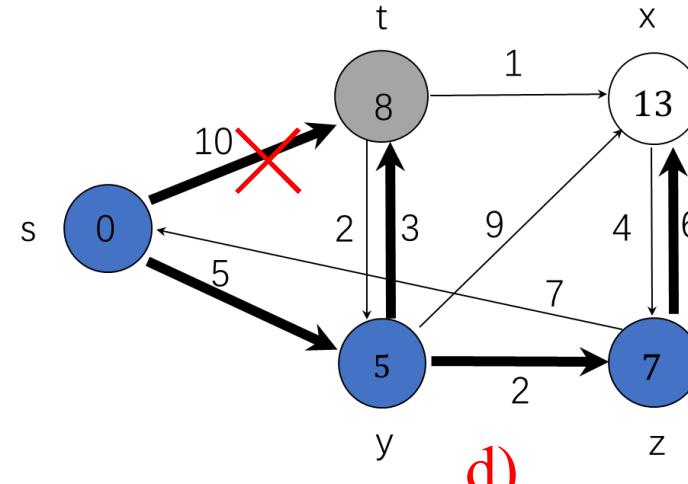
$$Q = \{t, x, y, z\}$$

# Dijkstra 算法

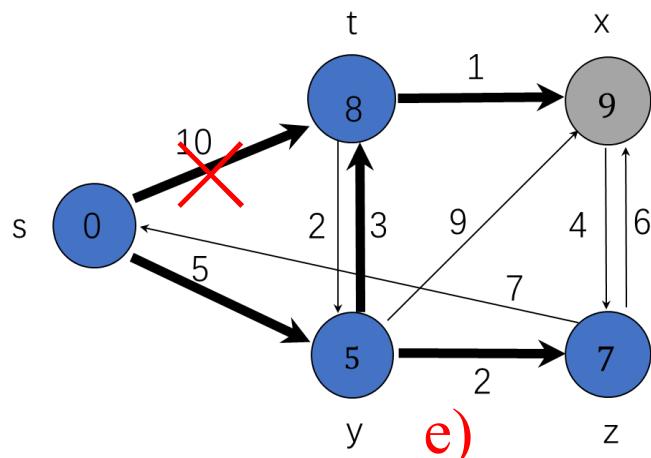
Dijkstra算法的执行过程: b) 至 f) while循环在每一次连续迭代后的情形。每个图中灰色的顶点被选作下一次迭代第5行的顶点*u*。 f) 图是最终结果



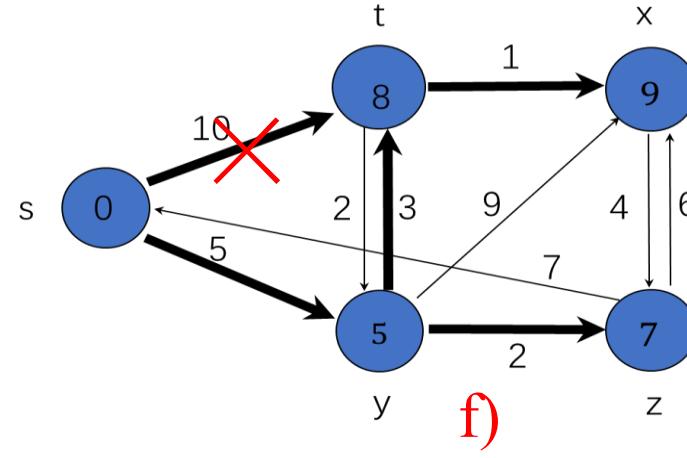
$$S = \{s, y\} \quad Q = \{t, x, z\}$$



$$S = \{s, y, z\} \quad Q = \{t, x\}$$



$$S = \{s, y, z, t\} \quad Q = \{x\}$$



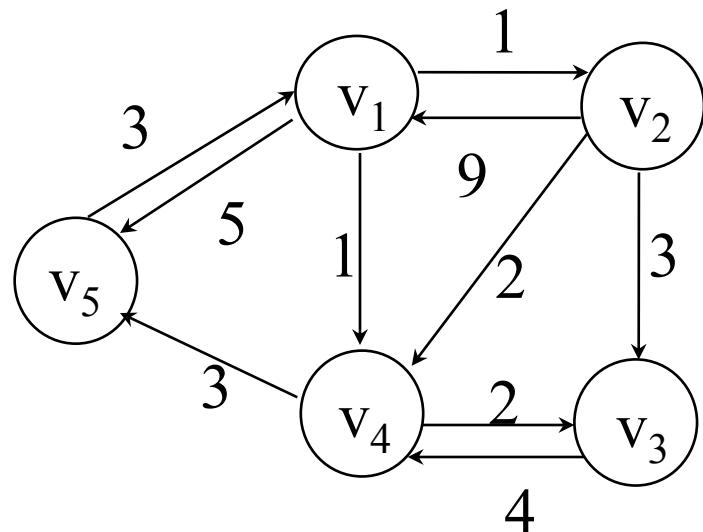
$$S = \{s, y, z, t, x\} \quad Q = \emptyset$$

# 多源最短路径

- 问题：给定一个边加权的有向图 $\textcolor{blue}{G} = (\textcolor{blue}{V}, E)$ ，找到图中每一对结点 $(\textcolor{blue}{u}, \textcolor{blue}{v})$ 间最短路径
  - 图 $\textcolor{blue}{G}$ 可能包含负边但是不包含负圈
- 依次把有向图 $\textcolor{blue}{G}$ 中的每个顶点作为源节点，执行 $|\textcolor{blue}{V}|$ 次单源最短路径算法，从而得到每对顶点之间的最短路径
- 动态规划-**Floyd算法**

# 图和权矩阵

$$\begin{matrix} & \color{red}{1} & \color{red}{2} & \color{red}{3} & \color{red}{4} & \color{red}{5} \\ \color{red}{1} & 0 & 1 & \infty & 1 & 5 \\ \color{red}{2} & 9 & 0 & 3 & 2 & \infty \\ \color{red}{3} & \infty & \infty & 0 & 4 & \infty \\ \color{red}{4} & \infty & \infty & 2 & 0 & 3 \\ \color{red}{5} & 3 & \infty & \infty & \infty & 0 \end{matrix} = \color{blue}{W}$$



$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ \text{有向边}(i,j) \text{的权重} & \text{if } i \neq j \text{ 且 } (i,j) \in E \\ \infty & \text{if } i \neq j \text{ 且 } (i,j) \notin E \end{cases}$$

# 动态规划算法设计步骤

- 分析优化解的结构
- 递归地定义最优值的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优值的信息
- 根据构造最优值的信息构造优化解

# 子问题

- 如何定义更小的问题?
- 一种方法是将路径限制在仅包含一个**有限集合**中的结点
- 开始这个集合是空的
- 这个集合可以一直增长到包含所有结点。

## 子问题

- 令  $D^{(k)}[i, j] =$  从结点  $v_i$  到结点  $v_j$  仅包含  $\{v_1, v_2, \dots, v_k\}$  的一条最短路径的权重。 (从结点  $v_i$  到结点  $v_j$  的所有中间结点全部取自前  $k$  个结点的一条最短路径的权重)
  - $D^{(0)} = W$
  - $D^{(n)} = D$  目标矩阵
- 如何从  $D^{(k-1)}$  计算  $D^{(k)}$ ? (递归方程)

# 动态规划算法设计步骤

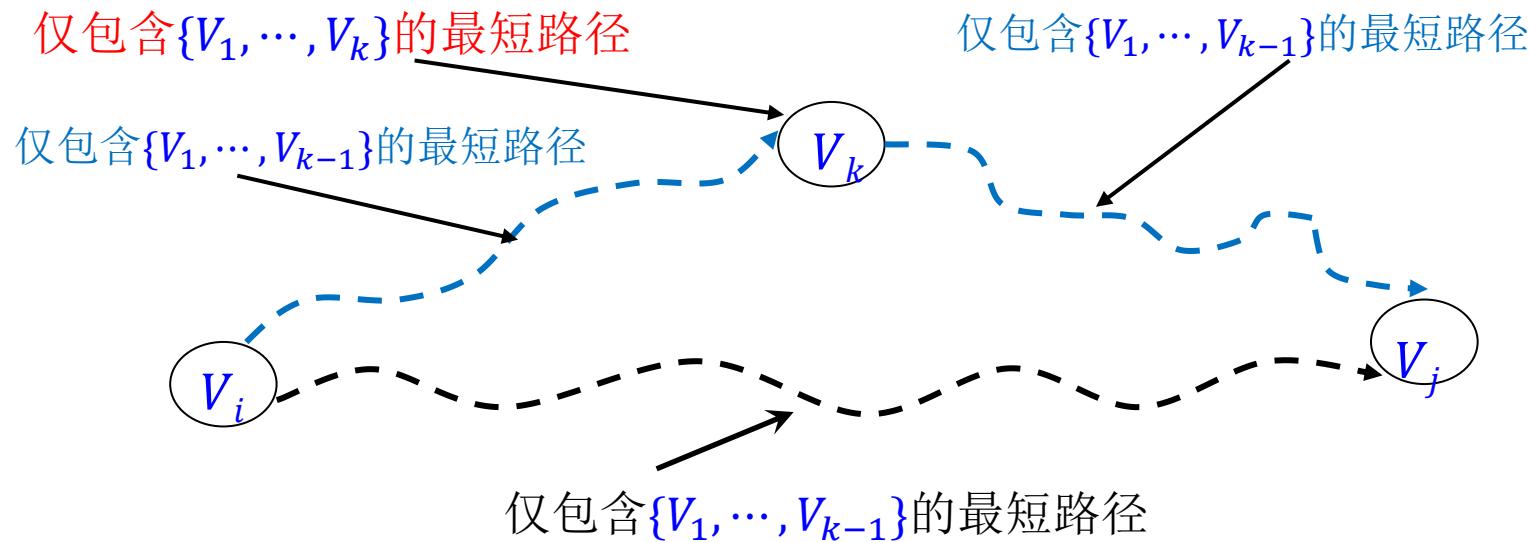
- 分析优化解的结构
- 递归地定义最优值的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优值的信息
- 根据构造最优值的信息构造优化解

# 递归定义

从结点  $v_i$  到结点  $v_j$  的最短路径  $P$  的中间结点全部取自  $\{v_1, v_2, \dots, v_k\}$ ，

对于结点  $v_k$ ：

- Case 1：结点  $v_k$  不是  $P$  上的中间结点。  $D^{(k)}[i, j] = D^{(k-1)}[i, j]$ 。
- Case 2：结点  $v_k$  是  $P$  上的中间结点。  $D^{(k)}[i, j] = D^{(k-1)}[i, k] + D^{(k-1)}[k, j]$ 。



$$D^{(k)}[i, j] = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{D^{(k-1)}[i, j], D^{(k-1)}[i, k] + D^{(k-1)}[k, j]\} & \text{if } k \geq 1 \end{cases}$$

# 动态规划算法设计步骤

- 分析优化解的结构
- 递归地定义最优值的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优值的信息
- 根据构造最优值的信息构造优化解

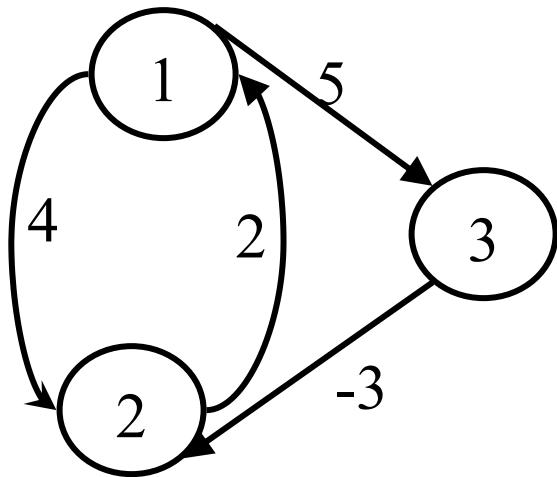
# Floyd算法

Floyd(W)

1.  $D^{(0)} \leftarrow W$  /\*初始化 D\*/
2.  $P \leftarrow 0$  /\*初始化 P\*/
3. For  $k \leftarrow 1$  To  $n$  Do
4.     For  $i \leftarrow 1$  To  $n$  Do
5.         For  $j \leftarrow 1$  To  $n$  Do
6.             If ( $D^{(k-1)}[i,j] > D^{(k-1)}[i,k] + D^{(k-1)}[k,j]$ ) Then
7.                  $D^{(k)}[i,j] \leftarrow D^{(k-1)}[i,k] + D^{(k-1)}[k,j]$
8.                  $P[i,j] \leftarrow k;$
9.         Else  $D^{(k)}[i,j] \leftarrow D^{(k-1)}[i,j]$

时间复杂度  $O(n^3)$ ,  $n$  是顶点个数

## 示例 1

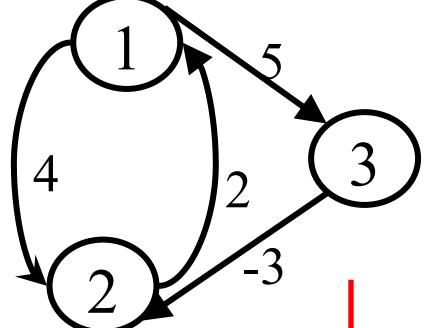


$$W = D^{(0)} =$$

1	2	3
0	4	5
2	0	$\infty$
$\infty$	-3	0

$$P =$$

1	2	3
0	0	0
0	0	0
0	0	0



$$D^{(0)} = \begin{array}{|c|c|c|} \hline & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 5 \\ \hline 2 & 2 & 0 & \infty \\ \hline 3 & \infty & -3 & 0 \\ \hline \end{array}$$

$\xrightarrow{\quad}$

$$P = \begin{array}{|c|c|c|} \hline & 1 & 2 & 3 \\ \hline 1 & 0 & 0 & 0 \\ \hline 2 & 0 & 0 & 1 \\ \hline 3 & 0 & 0 & 0 \\ \hline \end{array}$$

	1	2	3
1	0	4	5
2	2	0	$\infty$
3	$\infty$	-3	0

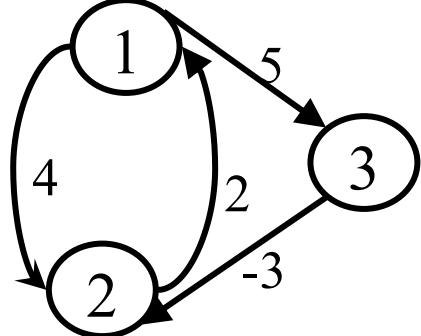
$$D^{(0)} =$$

$$\begin{aligned} D^{(1)}[2,3] &= \min(D^{(0)}[2,3], D^{(0)}[2,1] + D^{(0)}[1,3]) \\ &= \min(\infty, 7) \\ &= 7 \end{aligned}$$

$$\begin{aligned} D^{(1)}[3,2] &= \min(D^{(0)}[3,2], D^{(0)}[3,1] + D^{(0)}[1,2]) \\ &= \min(-3, \infty) \\ &= -3 \end{aligned}$$

$$D^{(k)}[i,j]$$

$$= \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{D^{(k-1)}[i,j], D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\} & \text{if } k \geq 1 \end{cases}$$



$$\mathbf{D}^{(1)} =$$

	1	2	3
1	0	4	5
2	2	0	7
3	$\infty$	-3	0

	1	2	3
1	0	4	5
2	2	0	7
3	-1	-3	0

	1	2	3
1	0	0	0
2	0	0	1
3	2	0	0

$$\mathbf{D}^{(2)} =$$

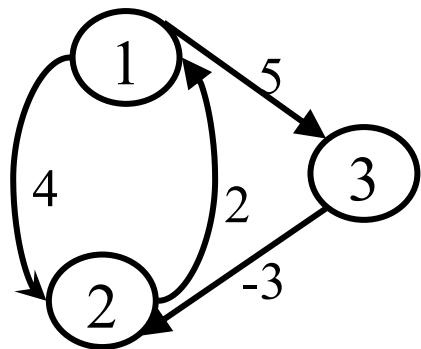
$$P =$$

$$\mathbf{D}^{(2)}[1,3] = \min(D^{(1)}[1,3], D^{(1)}[1,2] + D^{(1)}[2,3]) \\ = \min(5, 4 + 7) \\ = 5$$

$$\mathbf{D}^{(2)}[3,1] = \min(D^{(1)}[3,1], D^{(1)}[3,2] + D^{(1)}[2,1]) \\ = \min(\infty, -3 + 2) \\ = -1$$

$$\mathbf{D}^{(k)}[i,j]$$

$$= \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{\mathbf{D}^{(k-1)}[i,j], \mathbf{D}^{(k-1)}[i,k] + \mathbf{D}^{(k-1)}[k,j]\} & \text{if } k \geq 1 \end{cases}$$



$$\mathbf{D}^{(2)}$$

	1	2	3
1	0	4	5
2	2	0	7
3	-1	-3	0

$$\mathbf{D}^{(3)}$$

	1	2	3
1	0	2	5
2	2	0	7
3	-1	-3	0

$$\mathbf{D}^{(3)}[1,2]$$

$$\begin{aligned}
 \mathbf{D}^{(3)}[1,2] &= \min(D^{(2)}[1,2], D^{(2)}[1,3] + D^{(2)}[3,2]) \\
 &= \min(4, 5 + (-3)) \\
 &= 2
 \end{aligned}$$

$$\mathbf{D}^{(3)}[2,1]$$

$$\begin{aligned}
 \mathbf{D}^{(3)}[2,1] &= \min(D^{(2)}[2,1], D^{(2)}[2,3] + D^{(2)}[3,1]) \\
 &= \min(2, 7 + (-1)) \\
 &= 2
 \end{aligned}$$

$$P =$$

	1	2	3
1	0	2	5
2	2	0	7
3	-1	-3	0

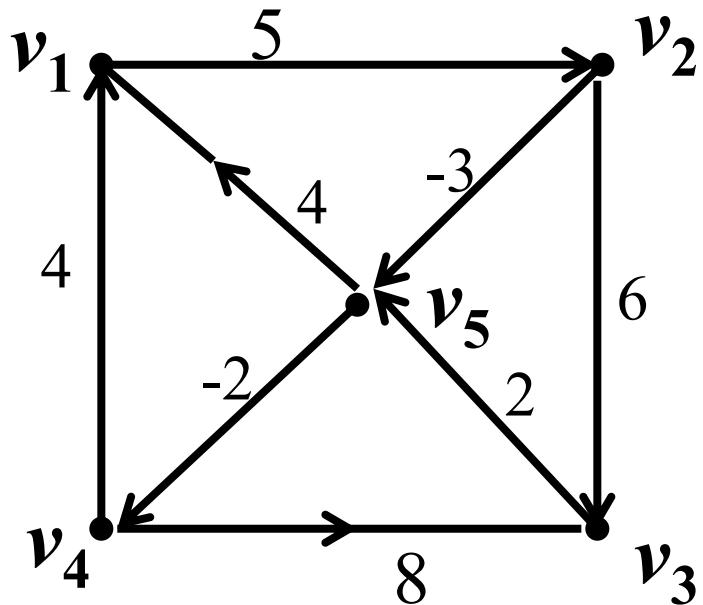
$$\mathbf{D}^{(3)}[2,1]$$

$$\begin{aligned}
 \mathbf{D}^{(3)}[2,1] &= \min(D^{(2)}[2,1], D^{(2)}[2,3] + D^{(2)}[3,1]) \\
 &= \min(2, 7 + (-1)) \\
 &= 2
 \end{aligned}$$

$$\mathbf{D}^{(k)}[i,j]$$

$$= \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\{\mathbf{D}^{(k-1)}[i,j], \mathbf{D}^{(k-1)}[i,k] + \mathbf{D}^{(k-1)}[k,j]\} & \text{if } k \geq 1 \end{cases}$$

## 示例 2

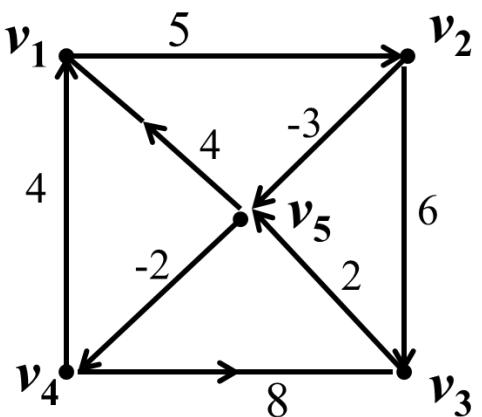


$$W = D^{(0)} =$$

$$\begin{bmatrix} 0 & 5 & \infty & \infty & \infty \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & \infty & 8 & 0 & \infty \\ 4 & \infty & \infty & -2 & 0 \end{bmatrix}$$

$$P =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\mathbf{D}^{(0)} = \begin{bmatrix} 0 & 5 & \infty & \infty & \infty \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & \infty & 8 & 0 & \infty \\ 4 & \infty & \infty & -2 & 0 \end{bmatrix}$$

$$\mathbf{D}^{(1)} = \begin{bmatrix} 0 & 5 & \infty & \infty & \infty \\ 0 & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & \infty \\ 4 & 9 & \infty & -2 & 0 \end{bmatrix}$$

$$k = 1$$

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

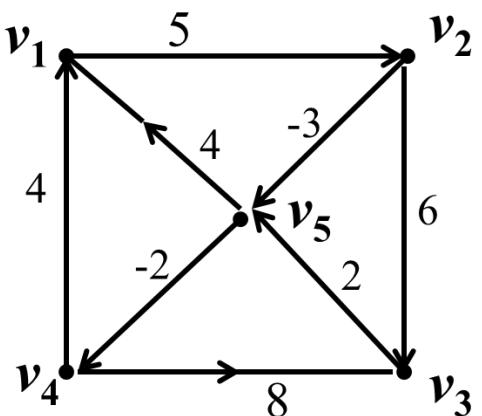
$$\mathbf{D}^{(1)}[2,3] = \min(D^{(0)}[2,3], D^{(0)}[2,1] + D^{(0)}[1,3]) \\ = \min(6, \infty + \infty) = 6$$

$$\mathbf{D}^{(1)}[2,4] = \min(D^{(0)}[2,4], D^{(0)}[2,1] + D^{(0)}[1,4]) \\ = \min(\infty, \infty + 4) = \infty$$

$$\mathbf{D}^{(1)}[2,5] = \min(D^{(0)}[2,5], D^{(0)}[2,1] + D^{(0)}[1,5]) \\ = \min(-3, \infty + \infty) = -3$$

$$\mathbf{D}^{(1)}[4,2] = \min(D^{(0)}[4,2], D^{(0)}[4,1] + D^{(0)}[1,2]) \\ = \min(\infty, 4 + 5) = 9$$

$$\mathbf{D}^{(1)}[5,2] = \min(D^{(0)}[5,2], D^{(0)}[5,1] + D^{(0)}[1,2]) \\ = \min(\infty, 4 + 5) = 9$$



$$D^{(1)} = \begin{bmatrix} 0 & 5 & \infty & \infty & \infty \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & \infty \\ 4 & 9 & \infty & -2 & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & 3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 4 & 9 & 15 & -2 & 0 \end{bmatrix}$$

$$k = 2$$

$$D^{(2)}[1,3] = \min(D^{(1)}[1,3], D^{(1)}[1,2] + D^{(1)}[2,3]) \\ = \min(\infty, 5 + 6) = 11$$

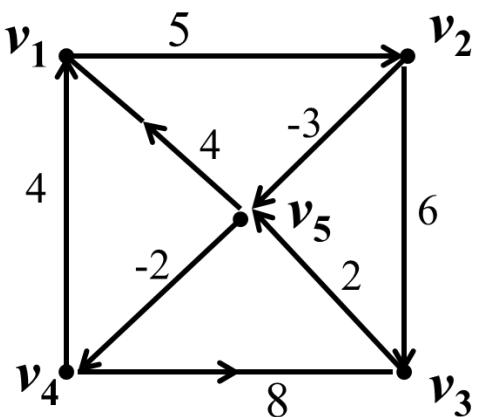
$$D^{(2)}[1,4] = \min(D^{(1)}[1,4], D^{(1)}[1,2] + D^{(1)}[2,4]) \\ = \min(\infty, 5 + \infty) = \infty$$

$$D^{(2)}[1,5] = \min(D^{(1)}[1,5], D^{(1)}[1,2] + D^{(1)}[2,5]) \\ = \min(\infty, 5 - 3) = 2$$

$$D^{(2)}[4,5] = \min(D^{(1)}[4,5], D^{(1)}[4,2] + D^{(1)}[2,5]) \\ = \min(\infty, 9 - 3) = 6$$

$$D^{(2)}[5,3] = \min(D^{(1)}[5,3], D^{(1)}[5,2] + D^{(1)}[2,3]) \\ = \min(\infty, 9 + 6) = 15$$

$$P = \begin{bmatrix} 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 2 & 0 & 0 \end{bmatrix}$$



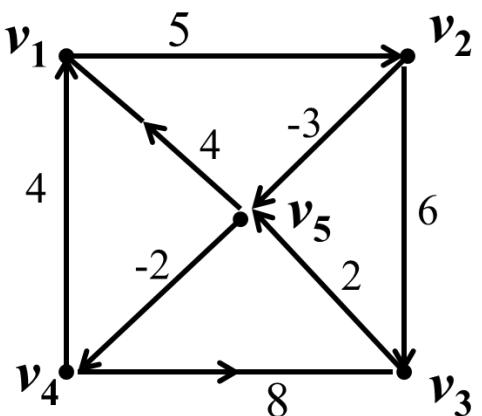
$$\mathbf{D}^{(2)} = \begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 4 & 9 & 15 & -2 & 0 \end{bmatrix}$$

$\mathbf{D}^{(3)}$

$$\begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 4 & 9 & 15 & -2 & 0 \end{bmatrix}$$

$$k = 3$$

$$P = \begin{bmatrix} 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 2 & 0 & 0 \end{bmatrix}$$

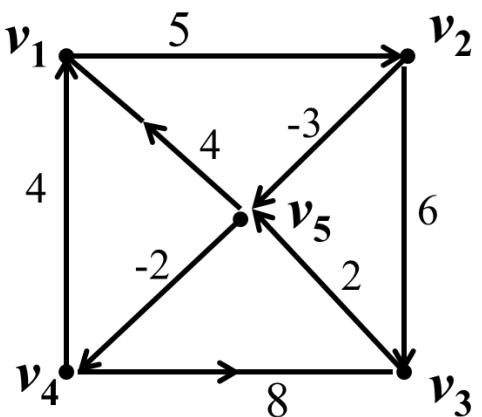


$$\mathbf{D}^{(3)} = \begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 4 & 9 & 15 & -2 & 0 \end{bmatrix}$$

$$\mathbf{D}^{(4)} = \begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 2 & 7 & 6 & -2 & 0 \end{bmatrix}$$

$$k = 4$$

$$P = \begin{bmatrix} 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 4 & 4 & 4 & 0 & 0 \end{bmatrix}$$



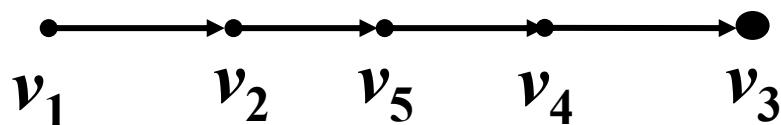
$$D^{(4)} = \begin{bmatrix} 0 & 5 & 11 & \infty & 2 \\ \infty & 0 & 6 & \infty & -3 \\ \infty & \infty & 0 & \infty & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 2 & 7 & 6 & -2 & 0 \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} 0 & 5 & 8 & 0 & 2 \\ -1 & 0 & 3 & -5 & -3 \\ 4 & 9 & 0 & 0 & 2 \\ 4 & 9 & 8 & 0 & 6 \\ 2 & 7 & 6 & -2 & 0 \end{bmatrix}$$

$$k = 5$$

$$P = \begin{bmatrix} 0 & 0 & 5 & 5 & 2 \\ 5 & 0 & 5 & 5 & 0 \\ 5 & 5 & 0 & 5 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 4 & 4 & 4 & 0 & 0 \end{bmatrix}$$

$P_{13} = 5,$   
 $P_{15} = 2,$   
 $P_{53} = 4,$   
 $P_{12} = 0, P_{25} = 0, P_{54} = 0, P_{43} = 0$



从 $v_1$ 到 $v_3$ 的最短路长度  $D^{(5)}[1,3] = 8$ 。

# Floyd算法: 使用两个 D 矩阵

Floyd(W)

1.  $D \leftarrow W$
2.  $P \leftarrow 0$
3. For  $k \leftarrow 1$  To  $n$  Do
4.     For  $i \leftarrow 1$  To  $n$  Do
5.         For  $j \leftarrow 1$  To  $n$  Do
6.             If ( $D[i,j] > D[i,k] + D[k,j]$ ) Then
7.                  $D'[i,j] \leftarrow D[i,k] + D[k,j]$
8.                  $P[i,j] \leftarrow k;$
9.             Else  $D'[i,j] \leftarrow D[i,j]$
10.        Move  $D'$  to  $D$

# Floyd算法： 使用一个D

Floyd(W)

1.  $D \leftarrow W$
2.  $P \leftarrow 0$
3. For  $k \leftarrow 1$  To  $n$  Do
4.     For  $i \leftarrow 1$  To  $n$  Do
5.         For  $j \leftarrow 1$  To  $n$  Do
6.             If  $(D[i,j] > D[i,k] + D[k,j])$  Then
7.                  $D[i,j] \leftarrow D[i,k] + D[k,j]$
8.              $P[i,j] \leftarrow k;$

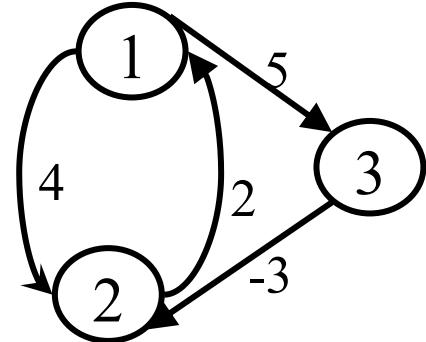
# 动态规划算法设计步骤

- 分析优化解的结构
- 递归地定义最优值的代价
- 自底向上地计算优化解的代价保存之，并获取构造最优值的信息
- 根据构造最优值的信息构造优化解

# 打印出从q到r的路径

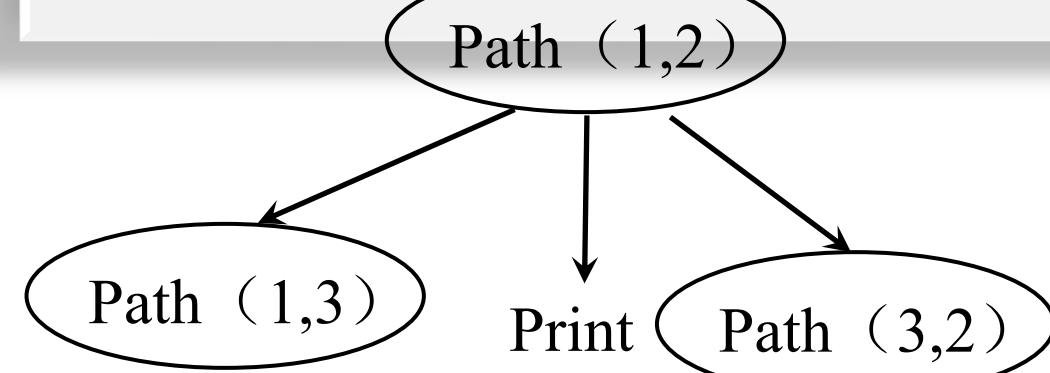
path(index q, r)

1. If ( $P[q, r] \neq 0$ ) Then
2. path(q,  $P[q, r]$ )
3. `println( "v" + P[q, r])`
4. path( $P[q, r]$ , r)
5. `return;`
6. Else return /\*没有中间结点\*/



$P =$

1	2	3
0	3	0
0	0	1
2	0	0



$$P(1,3) = 0$$

 $v_3$ 

$$P(3,2) = 0$$

多源最短路径的  
Floyd算法和矩阵  
链乘的关系？

结点  $v_1$  到结点  $v_2$  的最短路径是:  $v_1 \rightarrow v_3 \rightarrow v_2$

# 本讲内容

8.1 最短路径问题

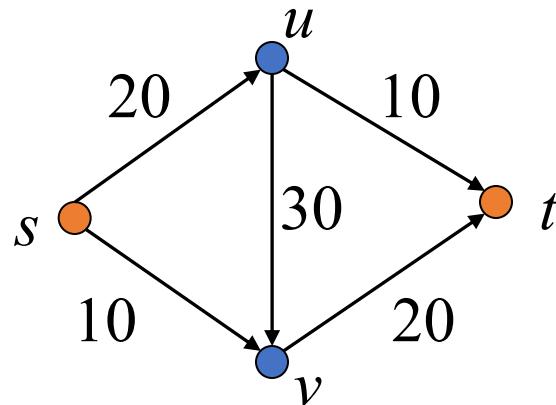
8.2 网络流问题

8.3 匹配问题

# 网络

网络指的是一个有向图  $G = (V, E)$  满足

- 每个有向边  $e$  有一个非负容量  $c_e$
- 有一个源结点  $s$  (Source) 没有入边
- 有一个目标点  $t$  (Target) 没有出边

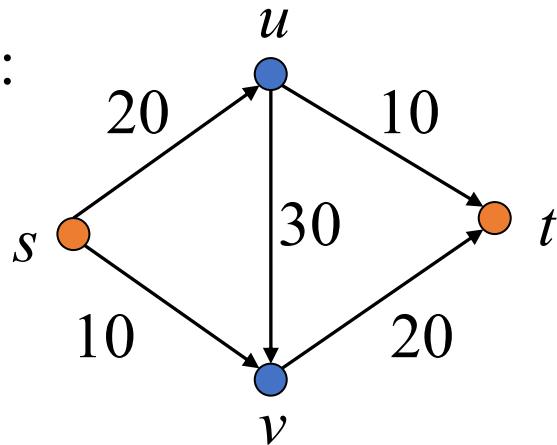


# 流

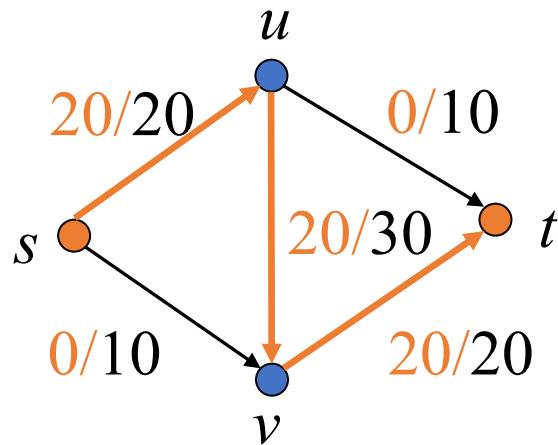
有向图  $G = (V, E)$  中的  $s \rightarrow t$  流是一个实值函数  $f: V \times V \rightarrow R^+$ , 且满足

- 容量条件: 对每个边  $e$ ,  $0 \leq f(e) \leq c_e$
- 保存条件: 对每个中间结点  $v$ ,  $\sum_{e \text{ in } v} f(e) = \sum_{e \text{ out } v} f(e)$
- 源和汇满足:  $\sum_{e \text{ in } t} f(e) = \sum_{e \text{ out } s} f(e)$

网络:



流:

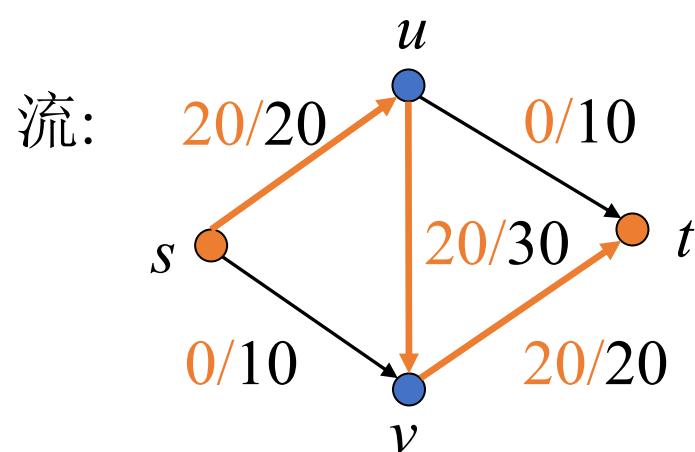
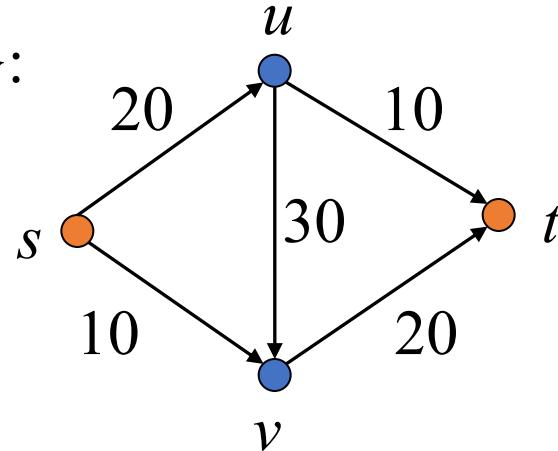


# 相关定义

给定图  $G = (V, E)$  中  $s \rightarrow t$  流  $f$  和任意结点集合  $S$

- $f^{in}(S) = \sum_{e \text{ in } S} f(e)$
- $f^{out}(S) = \sum_{e \text{ out } S} f(e)$
- 满足:  $f^{in}(S) = f^{out}(S)$ , 如  $f^{in}(u, v) = f^{out}(u, v) = 20$

网络:

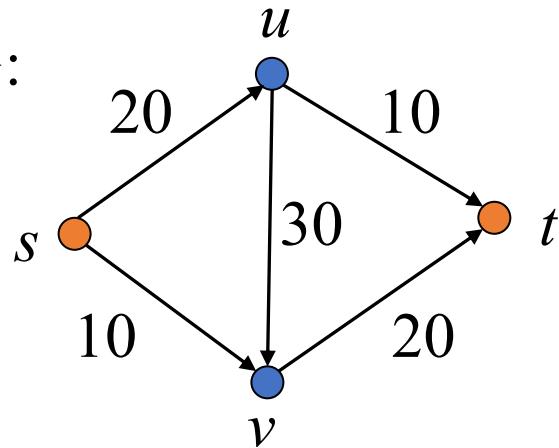


# 最大流问题

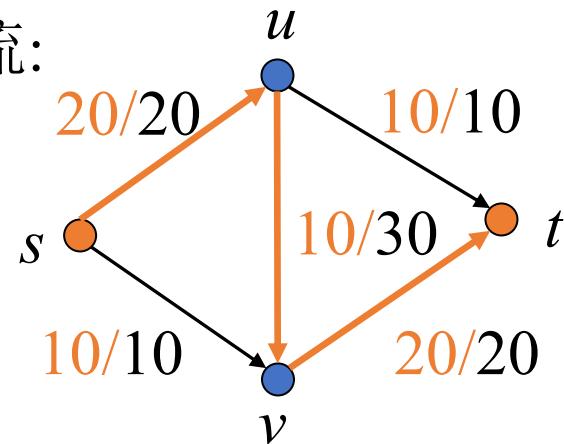
对于给定的有向图  $G = (V, E)$ , 容量都是正数, 如何求最大流。

- 如何高效地计算?
- 如最大流:  $f^{in}(t) = f^{out}(s) = 30$

网络:



最大流:

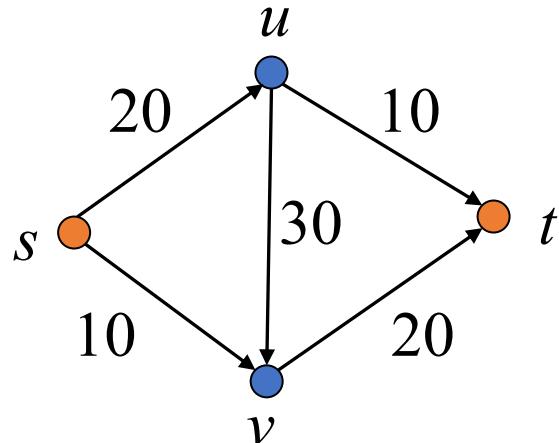


# 最大流问题

对于给定的有向图  $G = (V, E)$ , 容量都是正数, 如何求最大流。

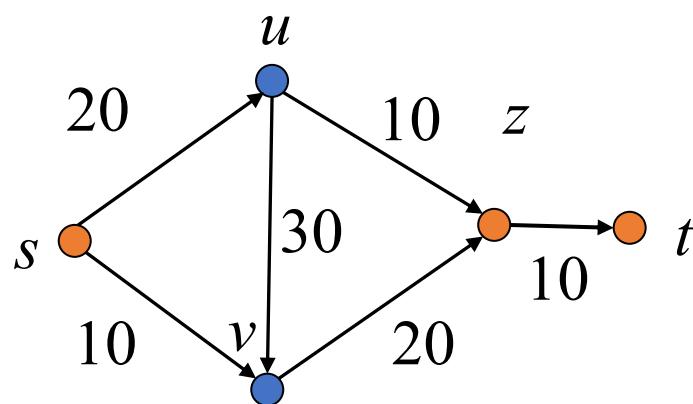
- 如何高效地计算?

网络:



动态规划?

- 考虑u的最大流, 考虑v的最大流
- S的最大流= $\min(20, u\text{的最大流}) + \min(10, v\text{的最大流})$



贪心?

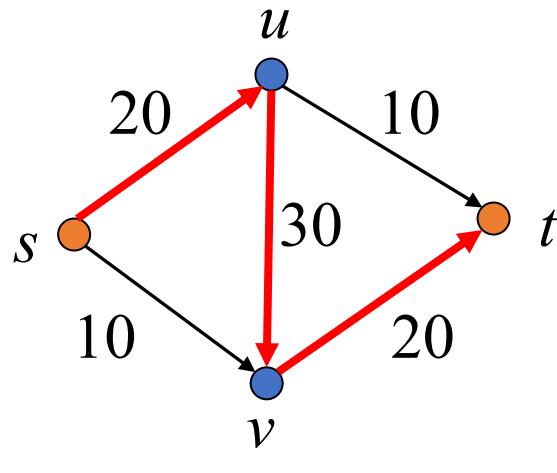
- 随便找一条  $s \rightarrow t$  的路径, 计算流, 更新图
- 接着在更新后的图里找新的路径

# 最大流问题

对于给定的有向图  $G = (V, E)$ , 容量都是正数, 如何求最大流。

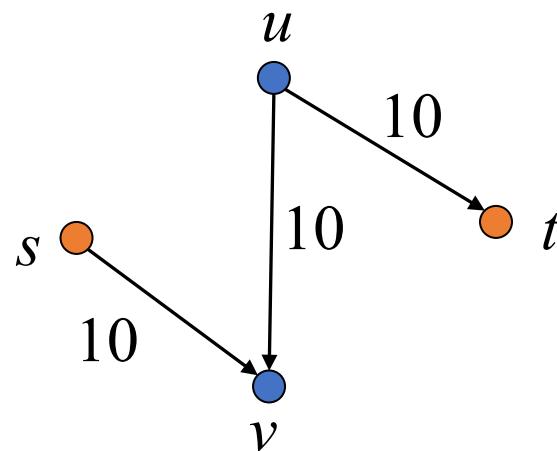
- 如何高效地计算?

网络:



贪心?

- 随便找一条  $s \rightarrow t$  的路径, 计算流, 更新图
- 接着在更新后的图里找新的路径
- 求出来的最大流为 20, 比真实的最大流 30 小

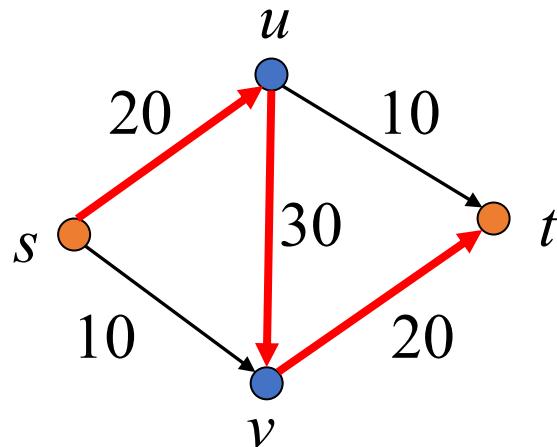


# 最大流问题

对于给定的有向图  $G = (V, E)$ , 容量都是正数, 如何求最大流。

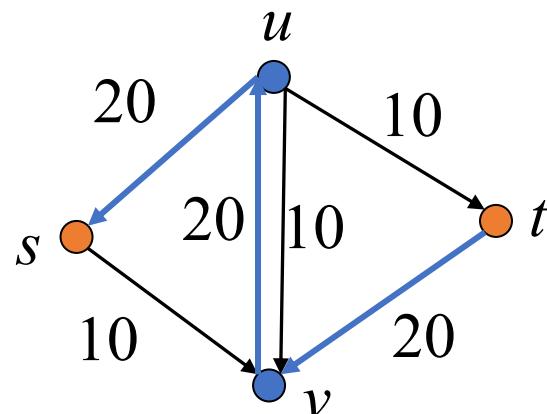
- 如何高效地计算?

网络:



贪心?

- 允许退回去
- 加入  $s \rightarrow u \rightarrow v \rightarrow t$ , 更新图
- 增加反向流  $t \rightarrow v \rightarrow u \rightarrow s$



# 余图（残差网络）

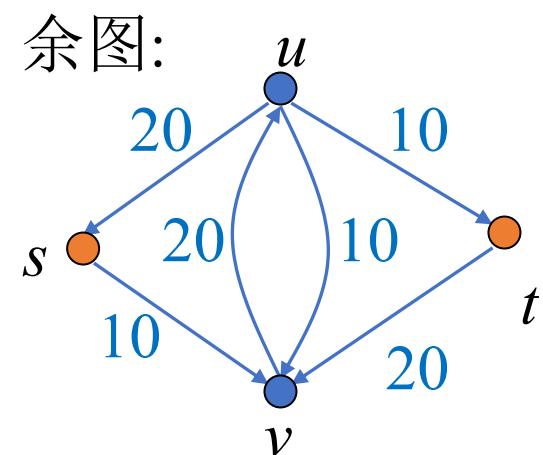
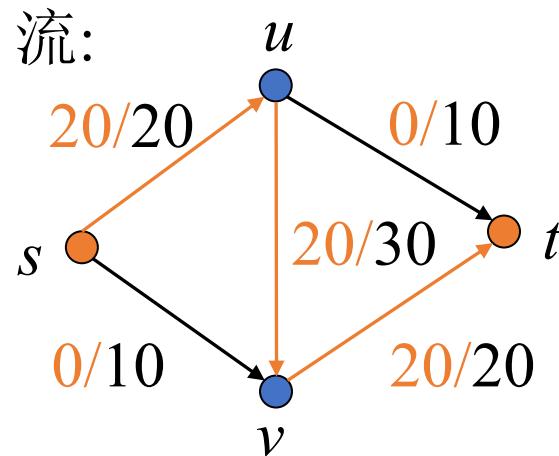
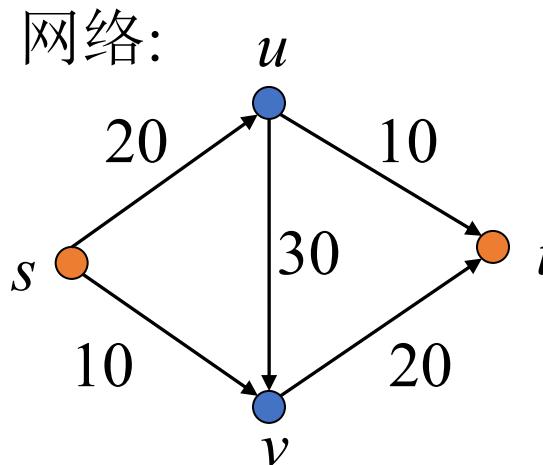
给定有向图  $G = (V, E)$  和一个流  $f$ , 则由流  $f$  诱导的有向图  $G_f = (V, E_f)$ , 其中

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

$$c_f(u, v) = c_e - f(e)$$

$$c_f(v, u) = \begin{cases} f(e), & \text{if } (v, u) \notin E \\ f(e) + c_f(v, u), & \text{otherwise} \end{cases}$$

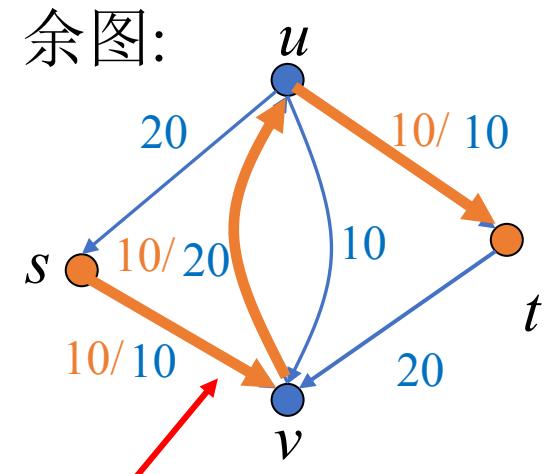
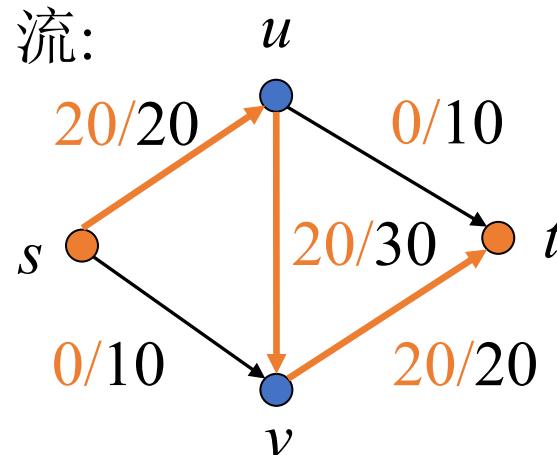
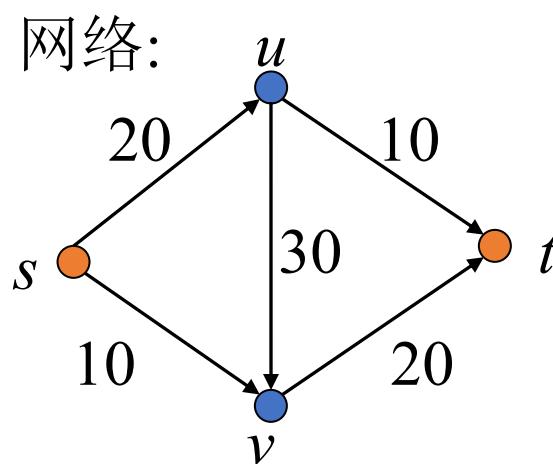
- 同样的结点, 中间结点和  $s, t$
- 满足  $c_e > f(e)$  的边  $e = (u, v)$  赋给权重  $c_f(u, v) = c_e - f(e)$  (剩余容量)
- 每条边  $e = (u, v)$ , 给其逆向边  $(v, u)$  赋给权重  $c_f(v, u) = f(e)$  (剩余容量)



# 增广路径和增广

有向图  $G = (V, E)$  和一个流  $f$ , 及其由流  $f$  诱导的余图  $G_f$

- 找到余图  $G_f$  中的一条流, 该流通过一条没有重复结点的路径  $p$ , 并且值和该路径  $p$  上的最小容量相等 (增广路径)



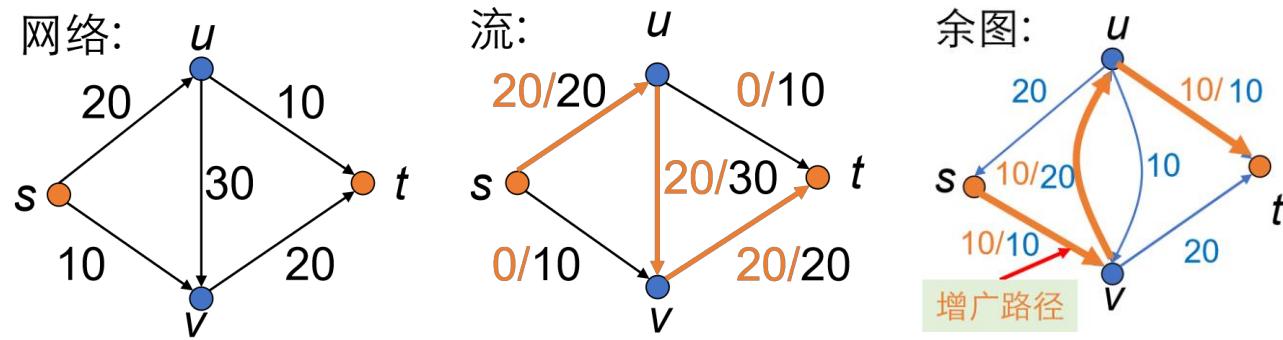
增广路径

- 增广的流量  $c_f(p) = \min\{c_f(u, v): (u, v) \in p\}$
- 如何寻找增广路径?

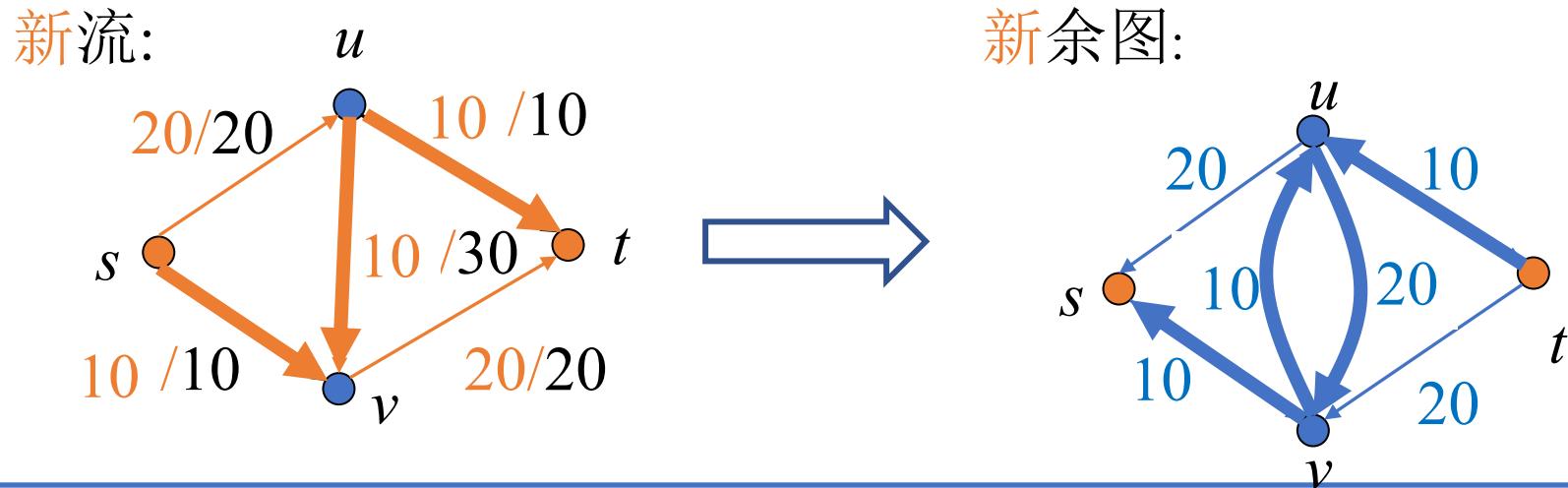
# 增广路径和增广

有向图  $G = (V, E)$  和一个流  $f$ , 及其由流  $f$  诱导的余图  $G_f$

- 找到余图中的一条流, 该流通过一条没有重复结点的路径  $p$ , 并且值和该路径上的最小容量相等 (增广路径)



- 根据增广路径上的最小流  $c_f(p)$ , 沿着增广路径更新流 (增广)

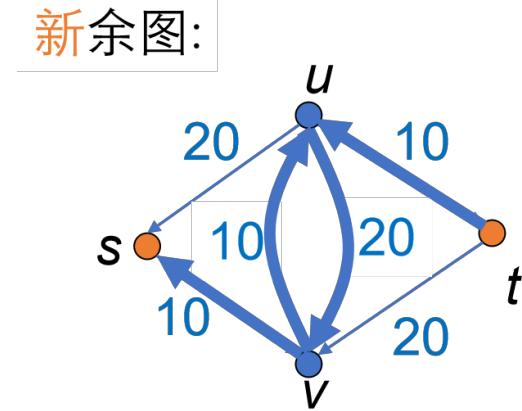
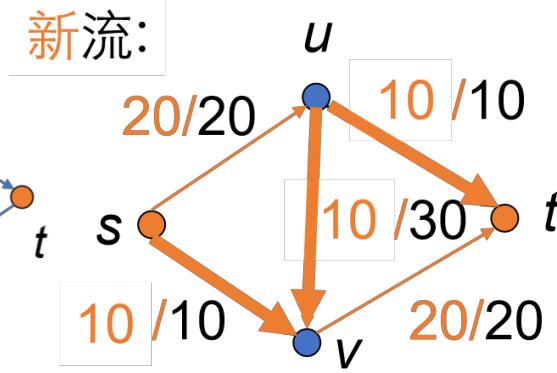
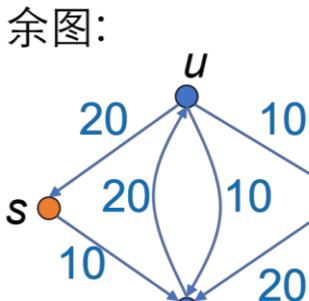
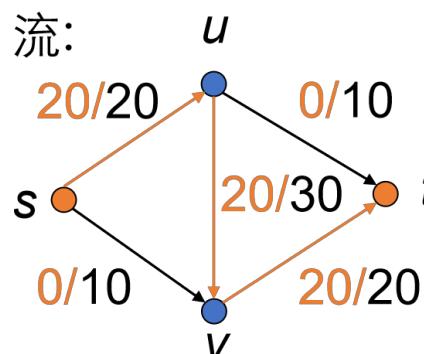


# Ford-Fulkerson 方法

Ford-Fulkerson( $G, s, t$ )

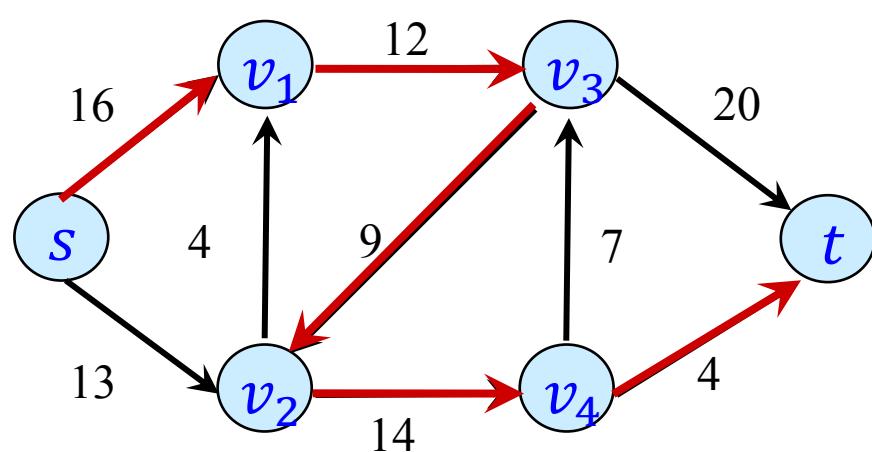
1. For each edge  $(u, v)$  in  $G.E$  Do
2.  $f(u, v) = 0$
3. While there exists a path  $p$  from  $s$  to  $t$  in  $G_f$  Do
4.  $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
5. For each edge  $(u, v)$  in  $p$  Do
6. If  $(u, v)$  in  $G.E$  Do
7.  $f(u, v) \leftarrow f(u, v) + c_f(p)$
8. Else  $f(v, u) \leftarrow f(v, u) - c_f(p)$

} 对于所有  $e$  初始化  $f(e) = 0$   
} 找到路径  $p$  的最小容量  
} 沿着路径  $p$  修改流



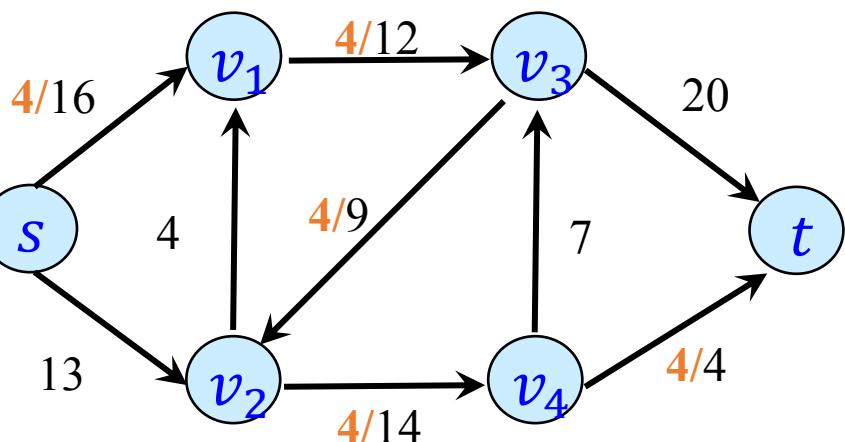
# 示例

余图

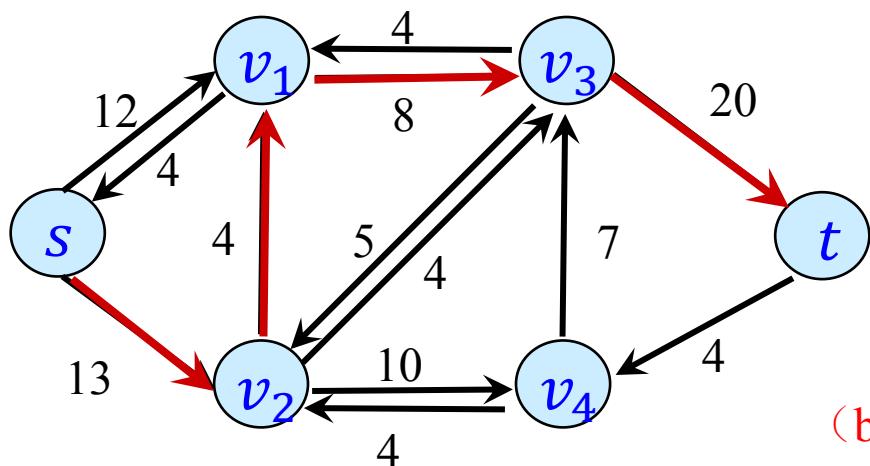


(a) 输入网络

流

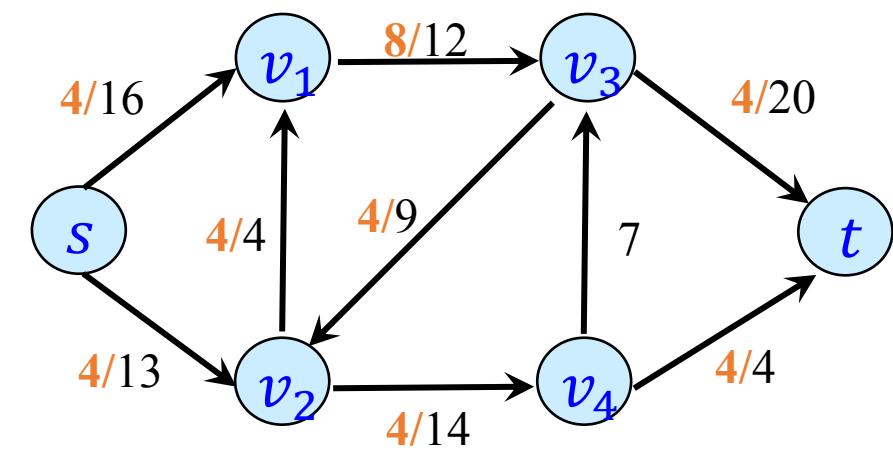


$$c_f(p) = \{16, 12, 9, 14, 4\} = 4$$



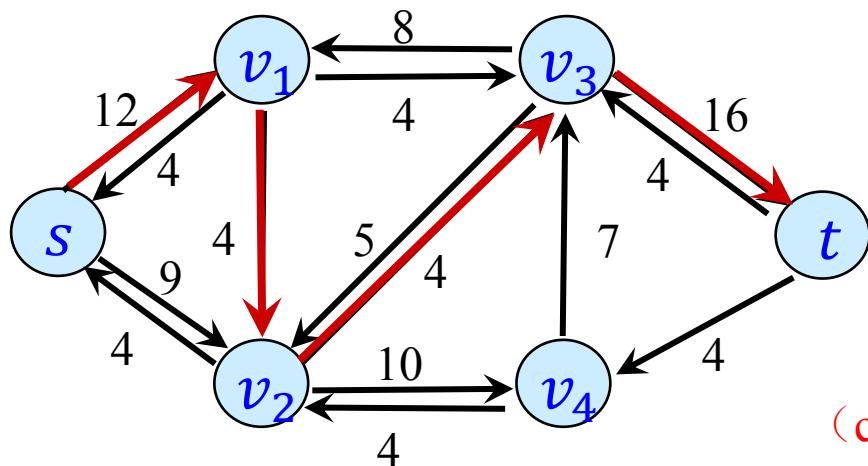
(b)

$$c_f(p) = \{13, 4, 8, 20\} = 4$$



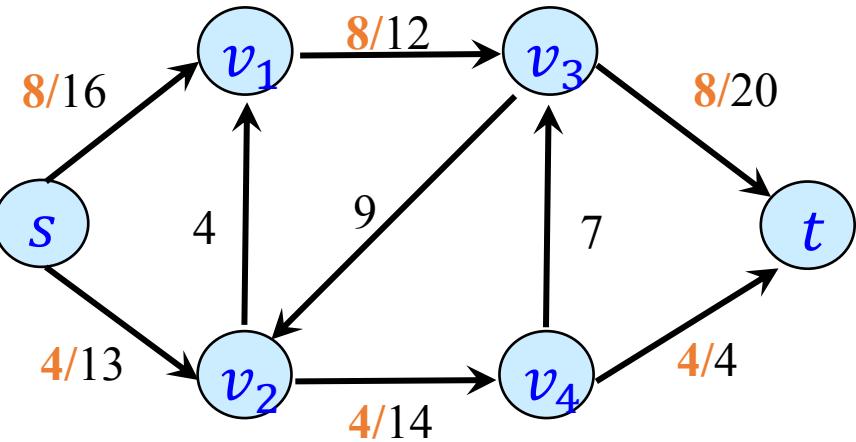
# 示例

余图

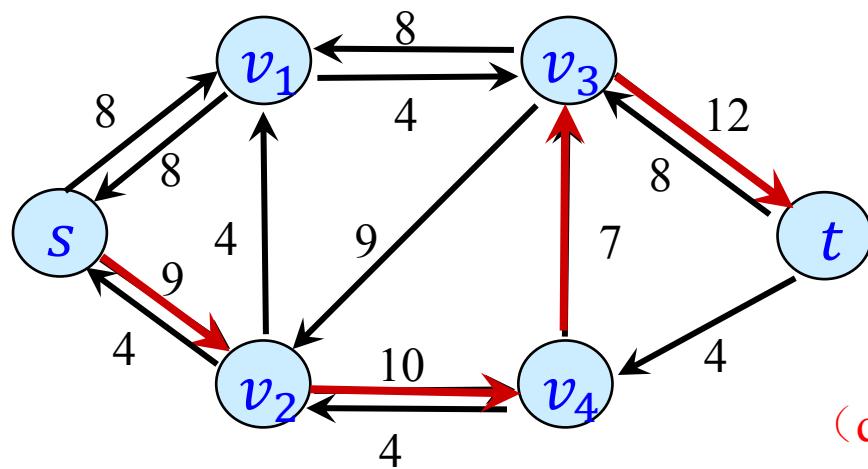


(c)

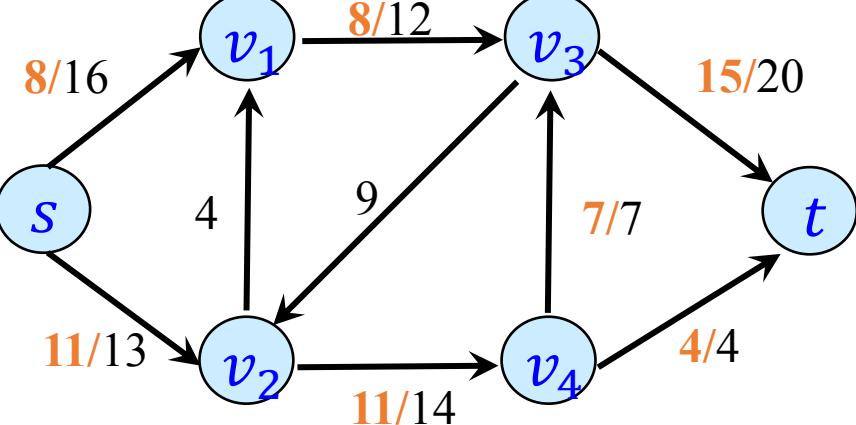
流



$$c_f(p) = \{12, 4, 4, 16\} = 4$$



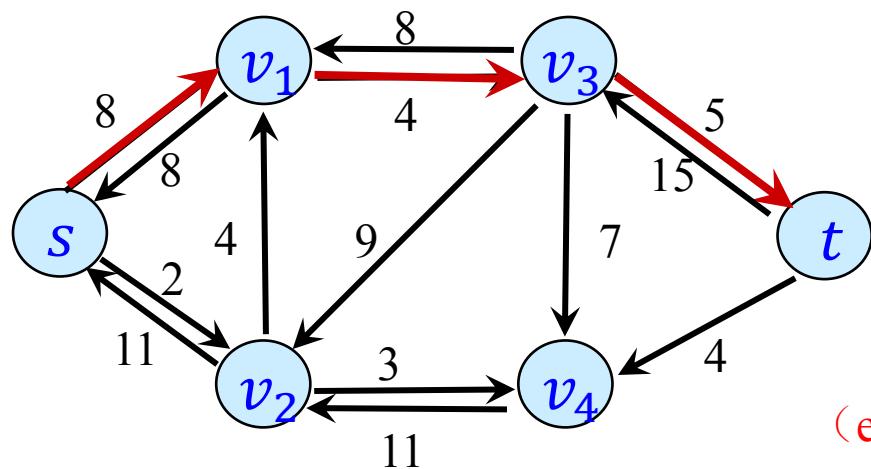
(d)



$$c_f(p) = \{9, 10, 7, 12\} = 7$$

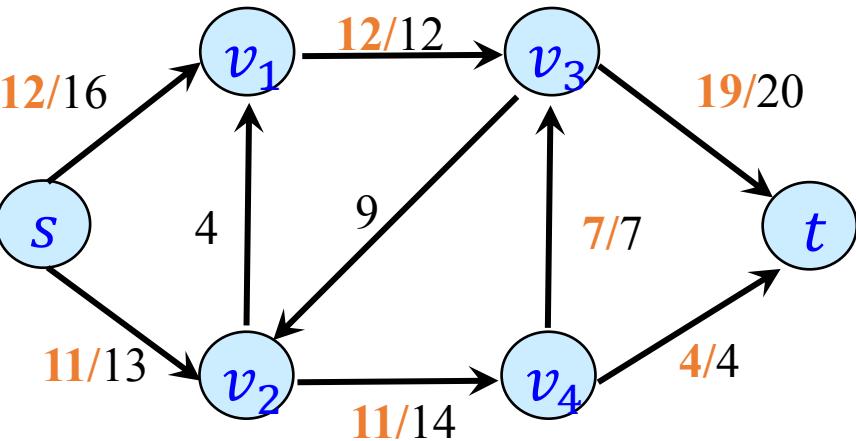
# 示例

余图

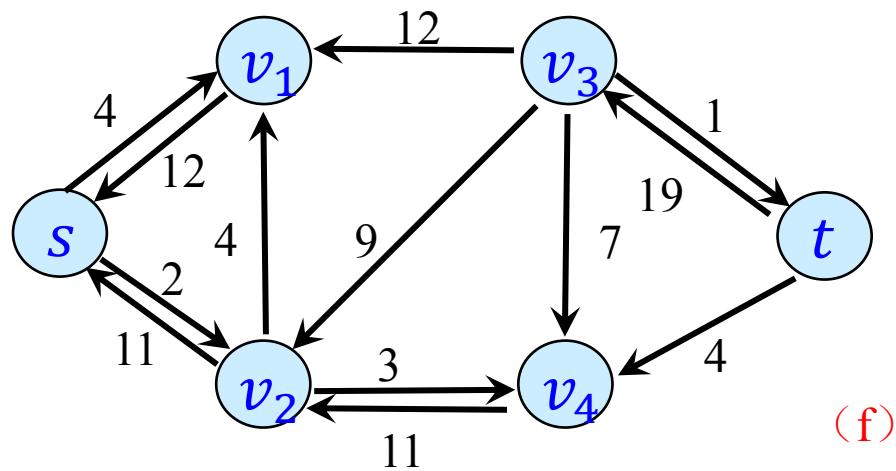


(e)

流



$$c_f(p) = \{8, 4, 12\} = 4$$



(f)

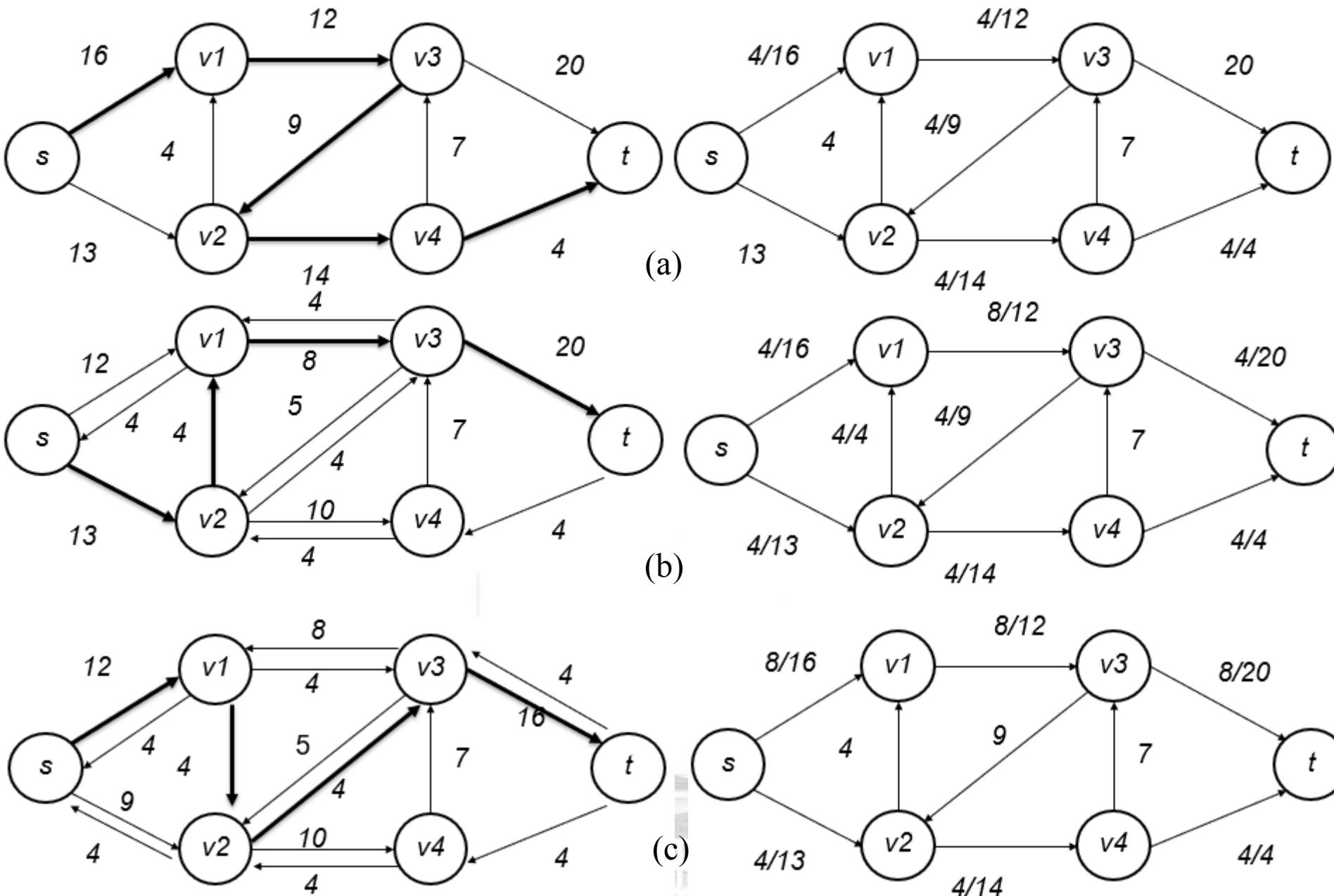
没有增广路径

最大流是  $12 + 11 = 23$

# 示例

余图

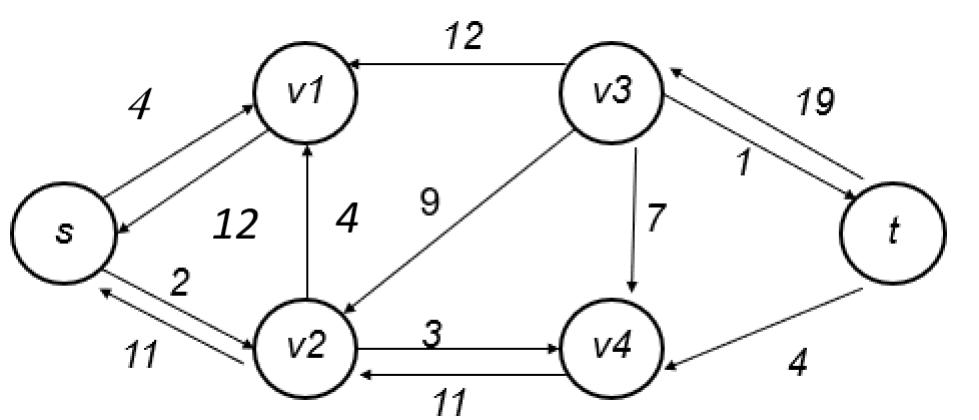
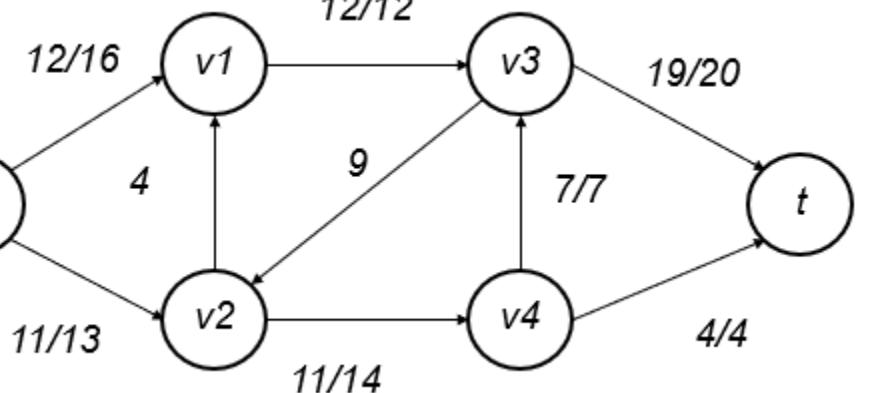
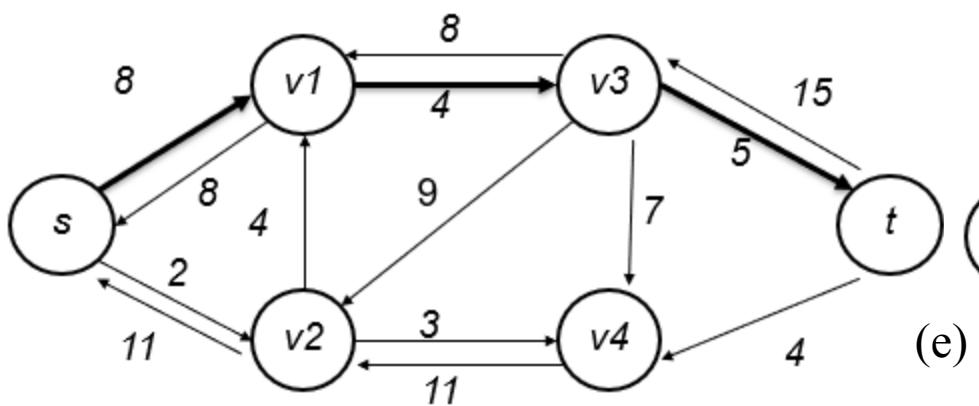
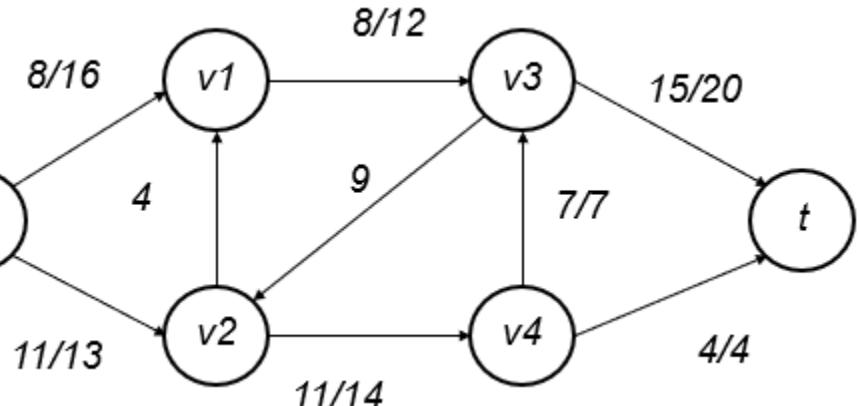
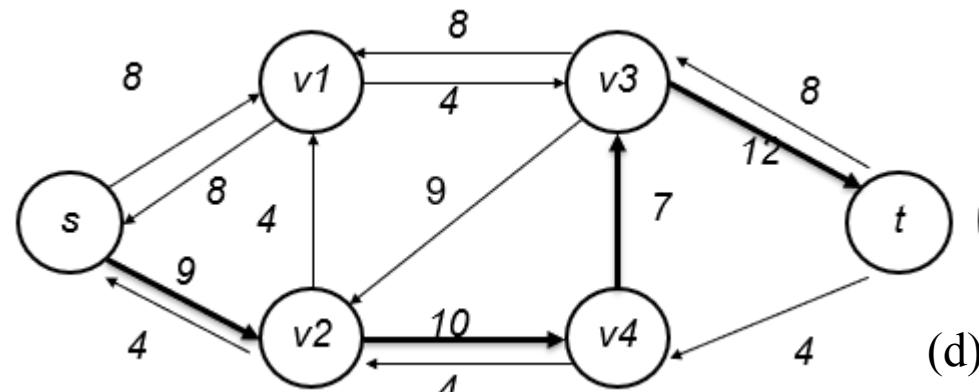
流



余图

示例

流



# 分析

正确性:

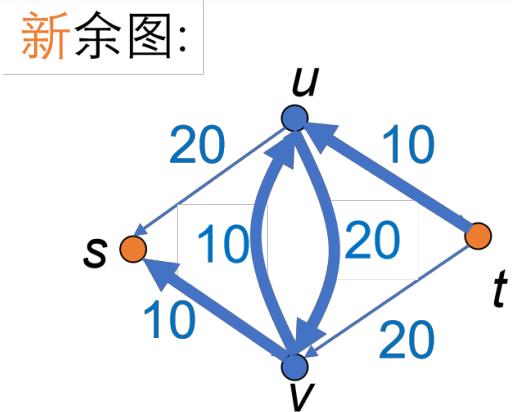
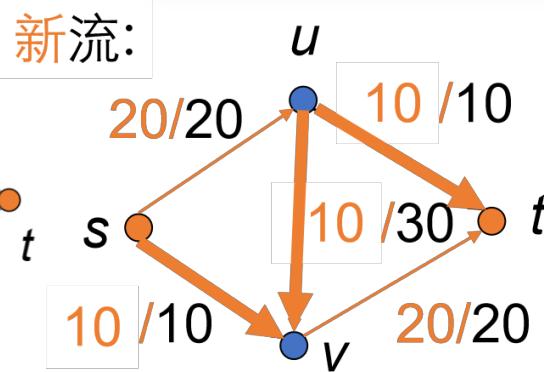
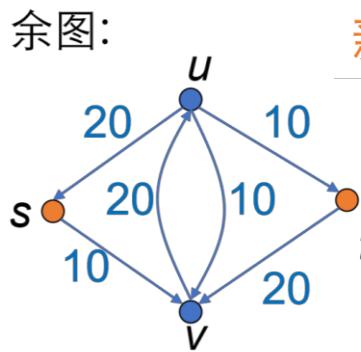
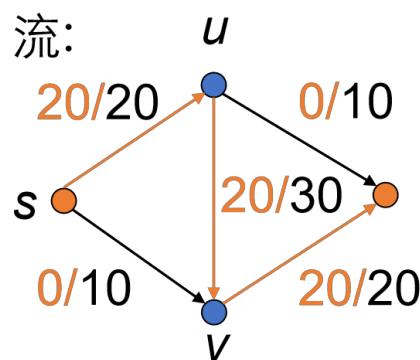
- 得到的是否是最大流 (证明不做要求)
- 可终止性- 每次流是一个整数且最少增加1 (假定整数容量)

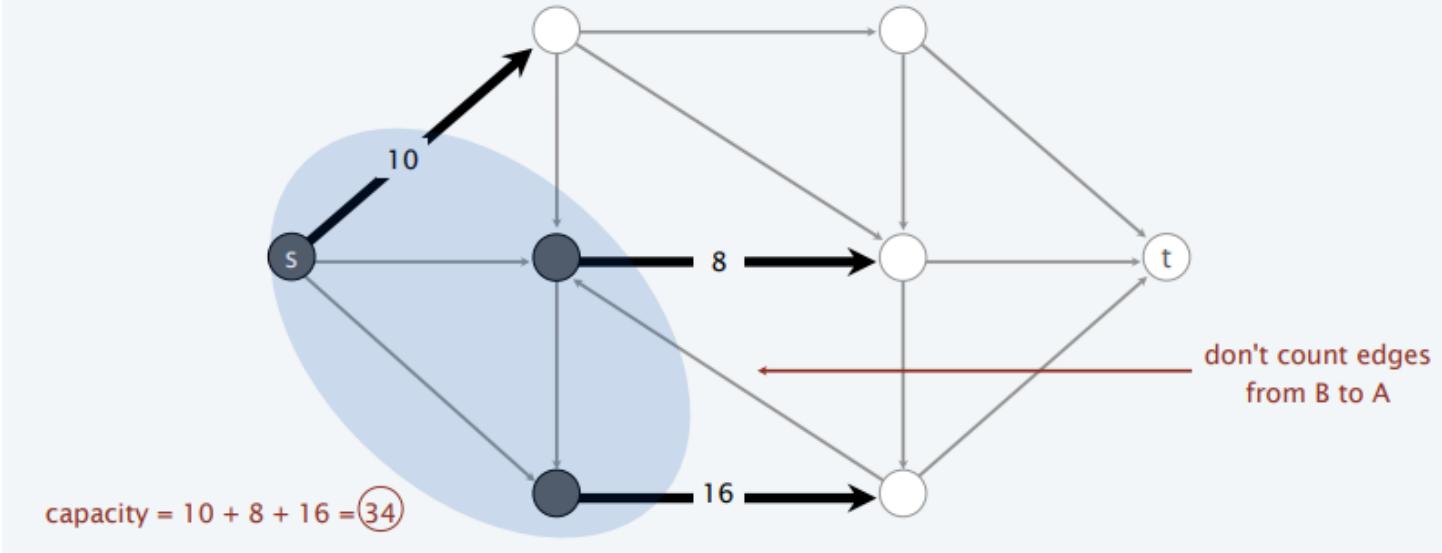
时间:  $O(mC)$

- 最多  $C$  轮iterations, 其中  $C$  是最大流的值,  $m$  是边数
- 每一轮  $O(m + n)$  步, 使用深度优先搜索 (DFS) 查找路径  $p$

Ford-Fulkerson( $G,s,t$ )

1. For each edge  $(u,v)$  in  $G.E$  Do
2.      $f(u,v)=0$
3. While there exists a path  $p$  from  $s$  to  $t$  in  $G_f$  Do
4.      $c_f(p) \leftarrow \min\{c_f(u,v):(u,v) \text{ is in } p\}$
5.     For each edge  $(u,v)$  in  $p$  Do
6.         If  $(u,v)$  in  $G.E$  Do
7.              $f(u,v) \leftarrow f(u,v)+c_f(p)$
8.         Else  $f(v,u) \leftarrow f(v,u)-c_f(p)$





**割：割（Cut）** 是对图上节点的分割 $(A,B)$ ，其中 $s \in A$  且 $t \in B$ 。

**割的流量：**所有从点集 $A$ 到点集 $B$ 的边的流量之和  
(注意一定从 $A$ 指向 $B$ )。

**最小割问题：**找到一个流量最小的割。

$$cap(A, B) = \sum_{e \text{ out } A} c(e)$$

令  $f$  为流网络  $G=(V,E)$  中的一个流，该网络的源点为  $s$ ，汇点为  $t$ ，则下面条件等价：

- 1、存在  $(A,B)$  是流网络  $G$  的一个割，使得  $cap(A,B) = val(f)$ 。
- 2、 $f$  是  $G$  的一个最大流。
- 3、残存网络  $G_f$  不包括任何增广路径。

证明 1->2:

- 显然，任何从  $s \rightarrow t$  的流  $f'$  都必须经过从  $A$  到  $B$  的边，又流的量不能超过边能承载的上限，所以有  $val(f') \leq cap(A,B)$ 。
- 又  $val(f) = cap(A,B)$ 。
- $f$  是所有的  $f'$  中最大的那个。

令  $f$  为流网络  $G=(V,E)$  中的一个流，该网络的源点为  $s$ ，汇点为  $t$ ，则下面条件等价：

- 1、存在  $(A,B)$  是流网络  $G$  的一个割，使得  $cap(A,B) = val(f)$ 。
- 2、 $f$  是  $G$  的一个最大流。
- 3、残存网络  $G_f$  不包括任何增广路径。

证明  $2 \rightarrow 3$ : (反证法)

- 假设对于流  $f$  存在一条增广路径。
- 那么我们可以通过在这条路径上加流量来增加。
- 因此， $f$  不是最大流。

矛盾！

令  $f$  为流网络  $G=(V,E)$  中的一个流，该网络的源点为  $s$ ，汇点为  $t$ ，则下面条件等价：

证明  $3 \Rightarrow 1$ ：

- 令  $f$  是没有增广路径的流。
- 令  $A$  是残存网络  $G_f$  中源点  $s$  可达的点的集合。
- 根据割  $A$  的定义， $s \in A$ 。
- 根据流  $f$  的定义，因为没有增广路径，所以  $t \notin A$ 。
- 对于任何从  $A$  到  $B$  的边  $e=(v,w)$  其中，有  $v \in A, w \in B$ ，有  $f(e) = c(e)$ 。
- 若不然， $G_f$  中存在一条边  $(v,w)$  的容量大于 0，则  $w$  与  $A$  连通，矛盾！
- $\sum f(e) = \sum c(e)$

# 本讲内容

- 8.1 最短路径问题
- 8.2 网络流问题
- 8.3 匹配问题

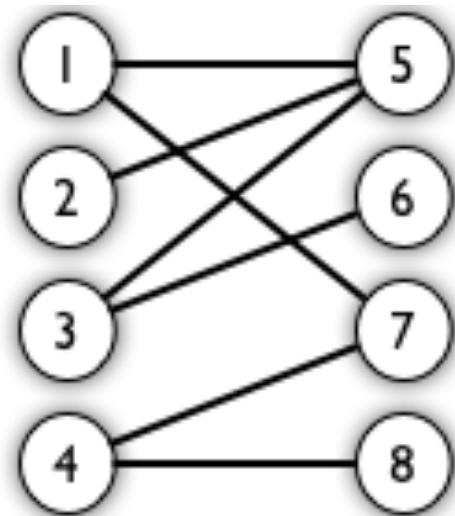
# 匹配

- 无向图
- 边集合  $M \subseteq E$ ,  $M$  中任意两条边都没有公共顶点。  $M$  中的边称为匹配边，  $M$  中的点称为匹配点。不在  $M$  中的边和点则称为未匹配边，未匹配点。
  - 极大匹配，不存在边  $e \notin M$  满足  $M \cup \{e\}$  也是匹配
  - 最大匹配， $|M|$  最大的匹配
  - 完美匹配， $|M| = \frac{n}{2}$ : 每个结点都是  $M$  中边的顶点

# 二分图匹配

# 二分图

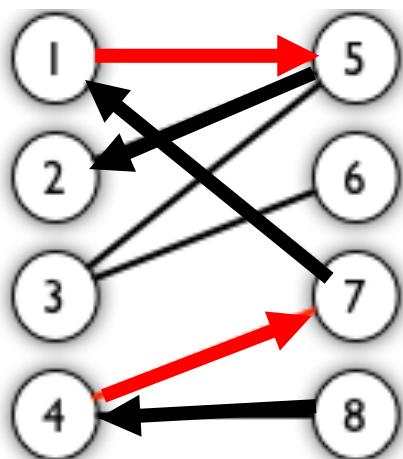
- 二分图又称作二部图，是图论中的一种特殊模型。设  $G = (V, E)$  是一个无向图，如果顶点  $V$  可分割为两个互不相交的子集  $(A, B)$ ，并且图中的每条边  $(i, j)$  所关联的两个顶点  $i$  和  $j$  分别属于这两个不同的顶点集 ( $i \in A, j \in B$ )，则称图  $G$  为一个二分图。



二分图的最大匹配问题：  
任务安排问题：机器集合  $A$  和任  
务集合  $B$  相匹配

# 交替路和增广路

- **交替路**: 从一个未匹配点出发，依次经过非匹配边、匹配边、非匹配边…形成的路径叫交替路。
- **增广路**: 从一个未匹配点出发，走交替路，**如果终点为另一个未匹配点（出发的点不算）**，则这条交替路称为增广路。

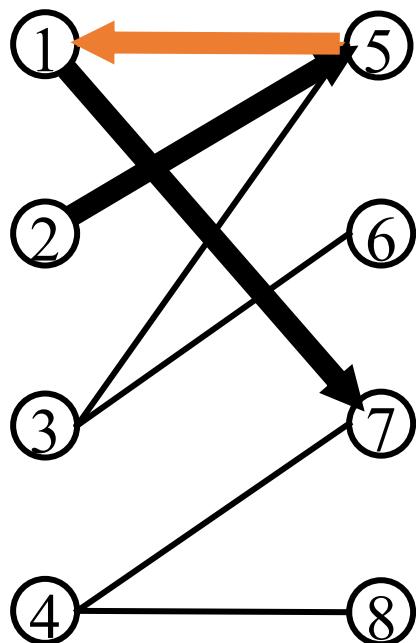


增广路:  $8 \rightarrow 4 \rightarrow 7 \rightarrow 1 \rightarrow 5 \rightarrow 2$

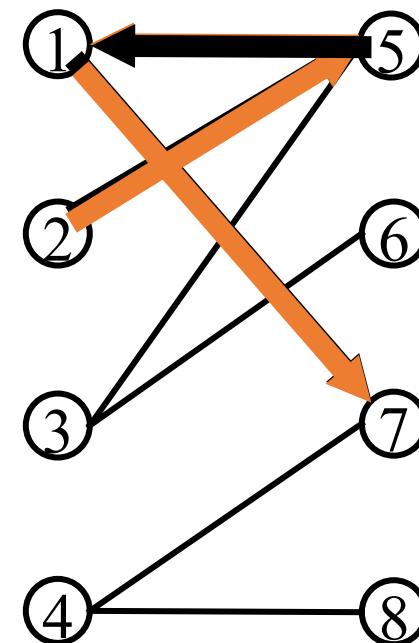
$1 \rightarrow 5, 4 \rightarrow 7$ 是匹配边

# 关于增广路的推论

- 增广路的路径长度必定为奇数，第一条边和最后一条边都不属于 $M$ 。
- 增广路经过取反操作（把增广路径上的匹配边变成非匹配边，把非匹配边变成匹配边），可以得到一个更大的匹配。
- $M$ 为 $G$ 的最大匹配当且仅当不存在相对于 $M$ 的增广路径。（不给证明）



取反



匹配 $M = \{(1,5)\}$

增广路:  $2 \rightarrow 5 \rightarrow 1 \rightarrow 7$

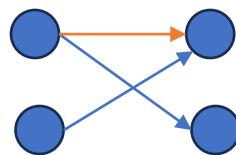
匹配 $M = \{(1,7), (2,5)\}$

不存在相对于 $M$ 的增广路径  $\Rightarrow M$  为  $G$  的最大匹配

证明：(反证法)

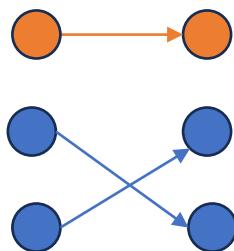
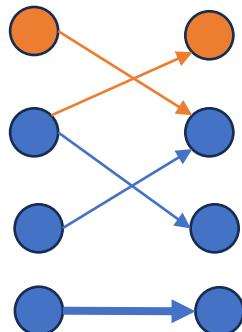
$M$  为一个匹配，但不是最大匹配， $M'$  为最大匹配，令  $H = (M - M') \cup (M' - M)$ ，考虑  $H$ 。黄色边为  $M$  中独有的边，蓝色边为  $M'$  中独有的边，因为  $M'$  的匹配数量比  $M$  多，蓝边至少比黄边多 1 条。

情况1：存在一条从蓝点指向蓝点的黄边。



蓝黄蓝边为  $M$  中  
的新的交替路

情况2：不存在一条从蓝点指向蓝点的黄边。

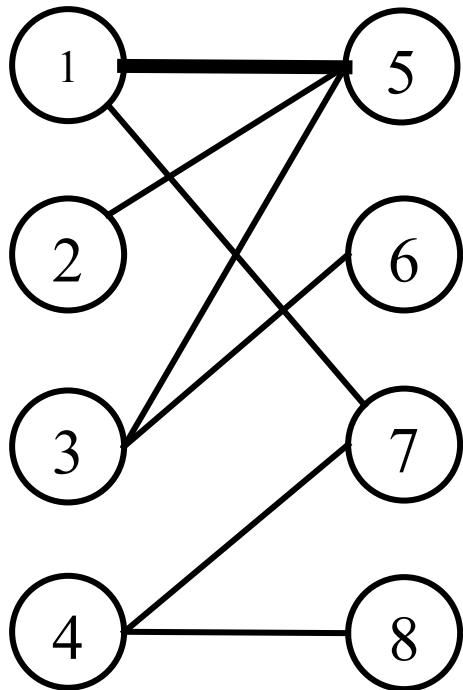


$M'$  中可以加入黄  
边，不是最大匹  
配，矛盾

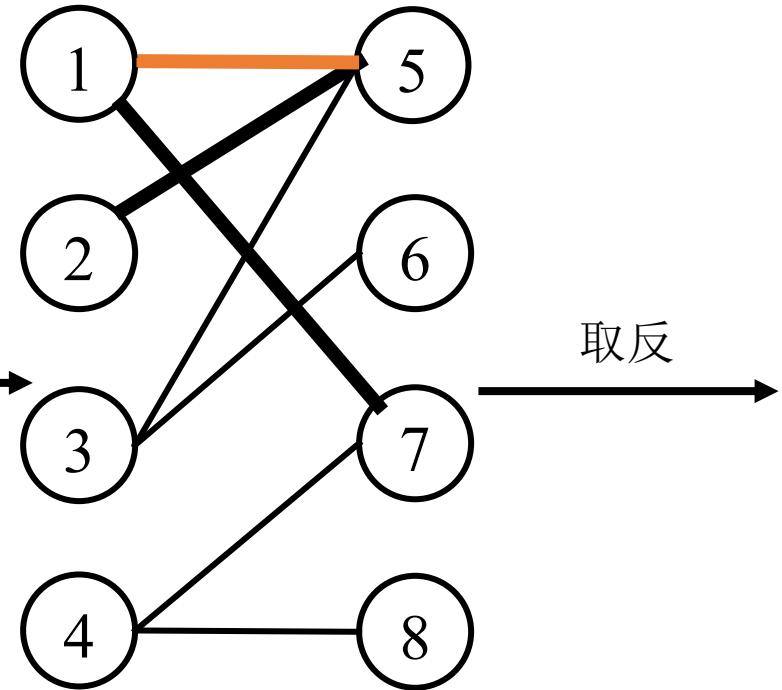
# 匈牙利算法

- 匹配 $M$ 是最大匹配，当且仅当对于任意匹配 $M'$ ，有 $|M| \geq |M'|$ 。
- 求二分图的最大匹配 $M$ 。
- 基本步骤：
  1. 置 $M$ 为空；
  2. 找出一条增广路径 $P$ ，通过取反操作，得到更大的匹配；
  3. 重复步骤2，直到找不出增广路为止。

# 例子



取反

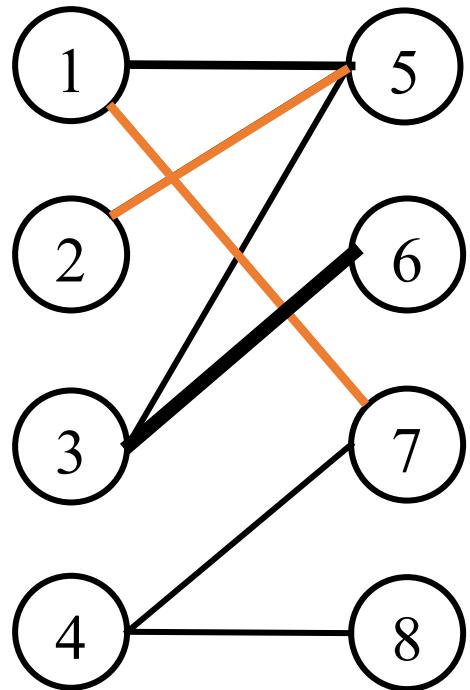


取反

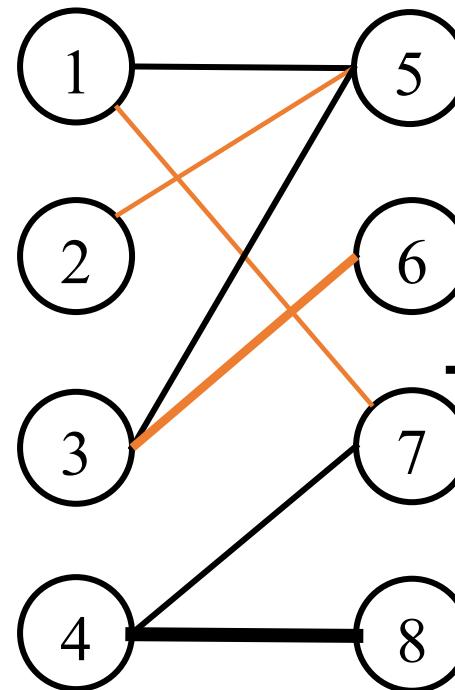
匹配 $M = \phi$   
增广路:  $1 \rightarrow 5$

匹配 $M = \{(1,5)\}$   
增广路:  $2 \rightarrow 5 \rightarrow 1 \rightarrow 7$

# 例子



取反



取反

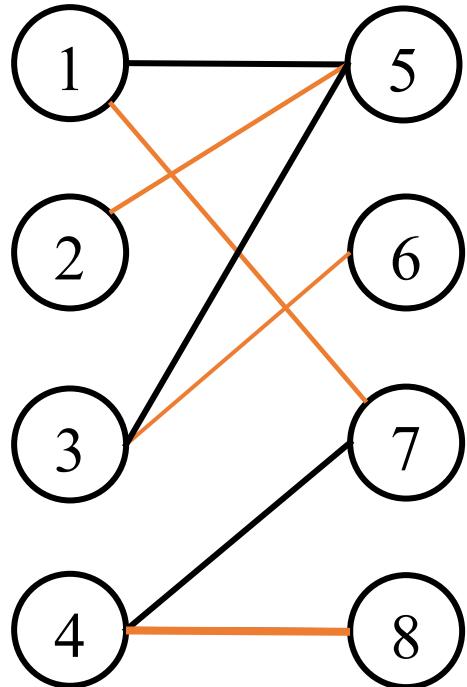
匹配  $M = \{(2,5), (1,7)\}$

增广路:  $3 \rightarrow 6$

匹配  $M = \{(2,5), (1,7), (3,6)\}$

增广路:  $4 \rightarrow 8$

# 例子



此时图中已无增广路，故该二分图的最大匹配为4（完美匹配）

$$\text{匹配 } M = \{(2,5), (1,7), (3,6), (4,8)\}$$

# 应用

现要给4个工人  $A, B, C, D$  分配任务，每个工人可完成不同的任务，但最多只能接受一个任务。共有4个任务，每个任务也只能分配给一个工人，问最多可以分配多少个任务给工人？（二分图最大匹配问题）

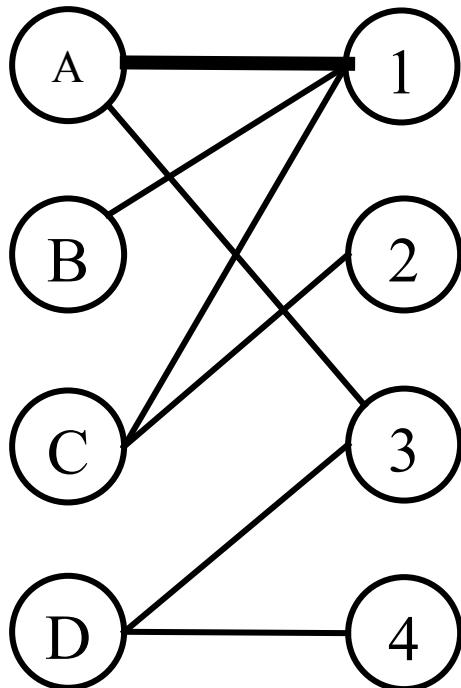
	任务1	任务2	任务3	任务4
A	1	0	1	0
B	1	0	0	0
C	1	1	0	0
D	0	0	1	1

1代表能完成，0代表不能完成

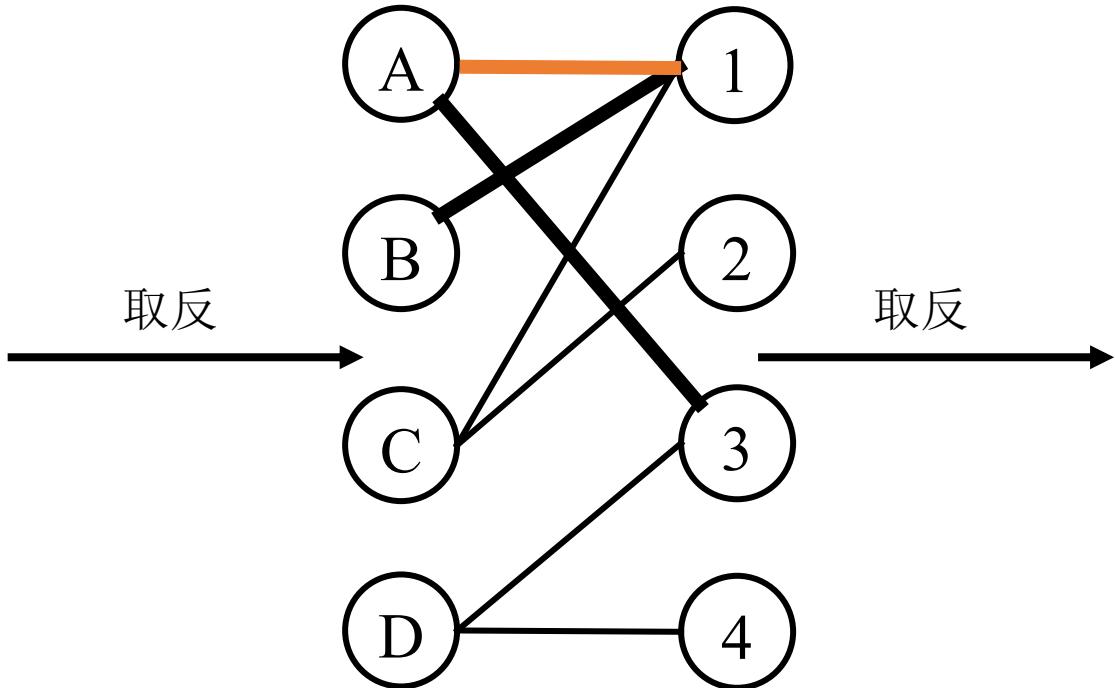
# 应用

- 分析
  - 该问题可以转化为一个图模型。工人和任务可以看作两个不相交的点集合，将工人和他能完成的任务相连（比如工人A能完成任务1，则图中含有边(A, 1)），因此图中的每条边的两个顶点分别落在两个集合里，所以此图是一个二分图。又因为“每个工人只能接受一个任务，每个任务只能分配给一个工人”，则意味着我们要寻找一个边集合，使得任意两条边没有公共顶点，这就是该图的匹配。“最多可以分配多少个任务”就是要寻找最大匹配，可以用匈牙利算法求解。

# 应用

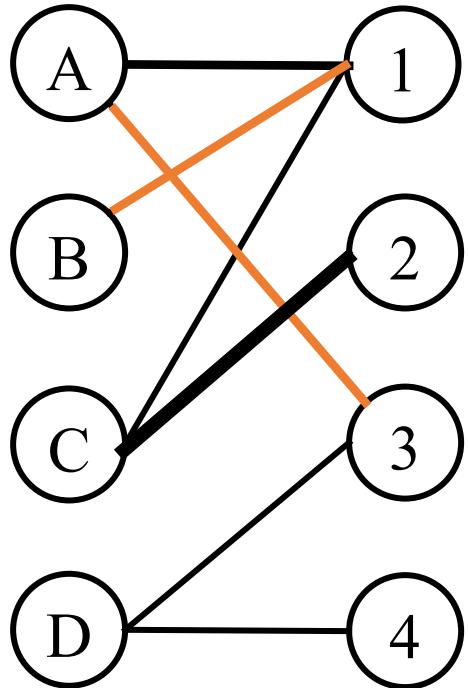


匹配  $M = \phi$   
增广路:  $A \rightarrow 1$



匹配  $M = \{(A, 1)\}$   
增广路:  $B \rightarrow 1 \rightarrow A \rightarrow 3$

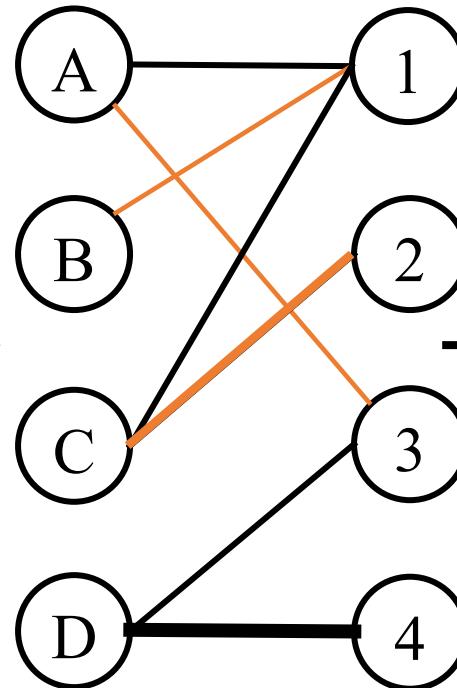
# 应用



匹配  $M = \{(A, 3), (B, 1)\}$

增广路:  $C \rightarrow 2$

取反

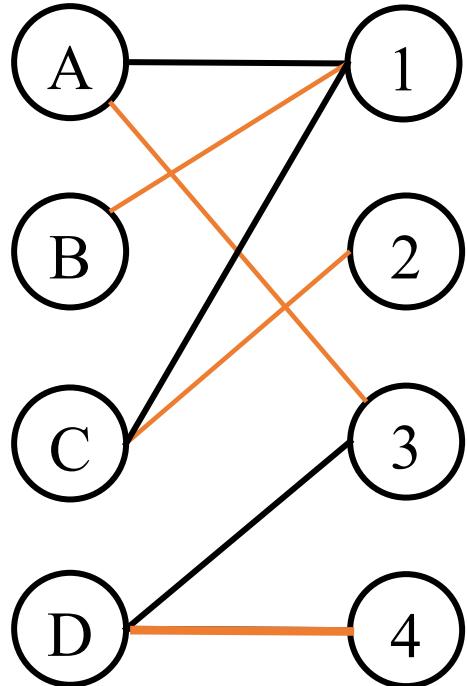


匹配  $M = \{(A, 3), (B, 1), (C, 2)\}$

增广路:  $D \rightarrow 4$

取反

# 应用



此时图中已无增广路，故该二分图的最大匹配为4。  
所以最多能分配4个任务。

$$\text{匹配 } M = \{(A, 3), (B, 1), (C, 2), (D, 4)\}$$