# 第八讲 网络优化模型

北京航空航天大学计算机学院

# 本讲目标

- 了解网络模型的应用范围；
- 掌握最小生成树算法；
- 掌握最短路径算法。

# 本讲内容

**一、网络模型的基本知识**

二、最小生成树算法

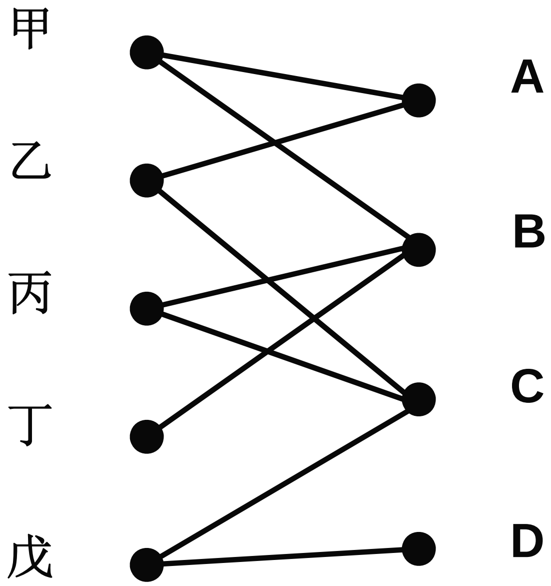
三、最短路径算法

# 网络模型的基本知识

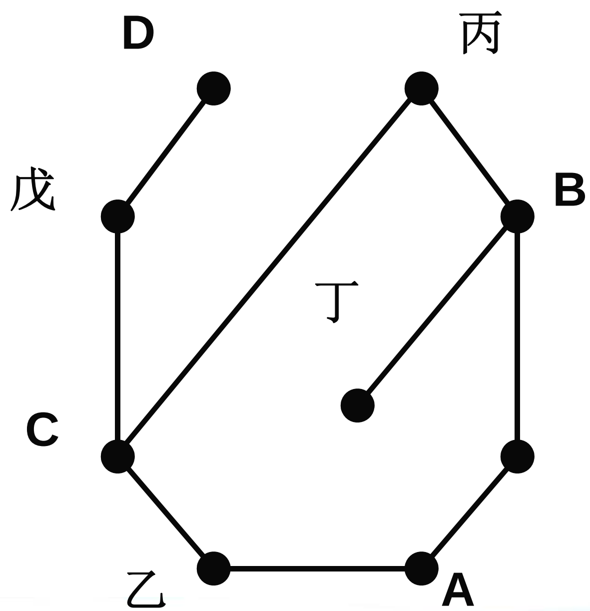
**图的定义**　　点集 $V \to$ 边集 $E \to$ 图 $G=(V,E)$

例　$V=\{$甲，乙，丙，丁，戊，A，B，C，D$\}$

$$E=\begin{cases}(\text{甲},\mathrm{A}),(\text{甲},\mathrm{B}),(\text{乙},\mathrm{A}),(\text{乙},\mathrm{C}),(\text{丙},\mathrm{B})\\(\text{丙},\mathrm{C}),(\text{丁},\mathrm{B}),(\text{戊},\mathrm{C}),(\text{戊},\mathrm{D})\end{cases}$$



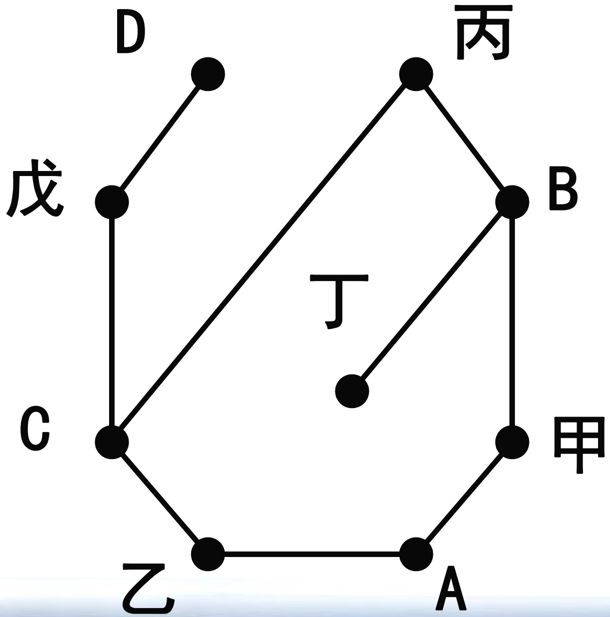

$\Leftrightarrow (V,E) \Leftrightarrow$



# 网络模型的基本知识

无向图

$$(甲, A) = (A, 甲)$$

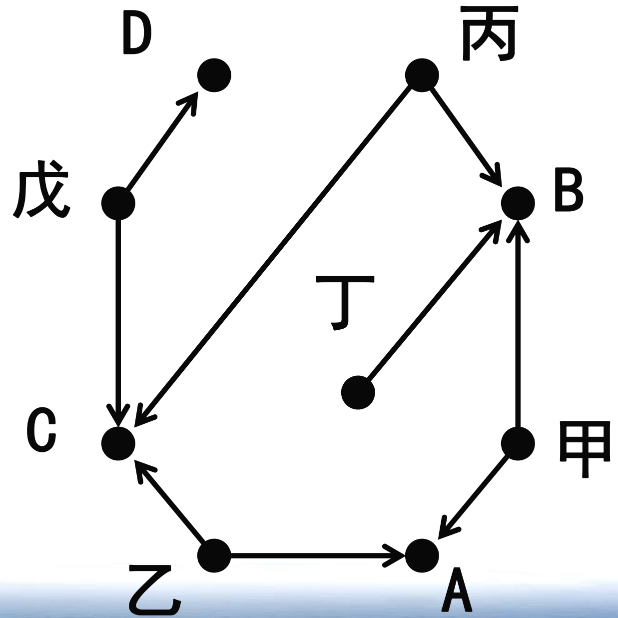



有向图

$$(甲, A) \neq (A, 甲)$$


始点

终点



# 网络模型的基本知识

相邻　　如果 $v_1, v_2 \in V$，$(v_1, v_2) \in E$，称 $v_1, v_2$ 相邻，

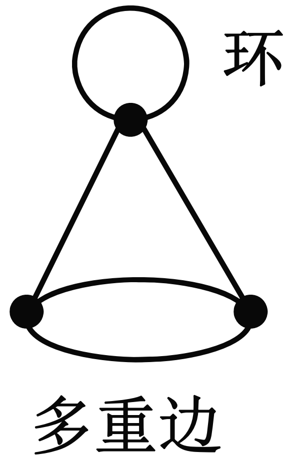
称 $v_1, v_2$ 为 $(v_1, v_2)$ 的端点

如果 $e_1, e_2 \in E$，并且有公共端点 $v \in V$，

称 $e_1, e_2$ 相邻，称 $e_1, e_2$ 为 $v$ 的关联边

自回路　　两端点相同的边，或称为环

多重边　　两点之间多于一条一样的边




简单图　　不含自回路和多重边的图

对 $G=(V, E)$，$m(G)=|E|, n(G)=|V|$ 表示边数和点数

# 网络模型的基本知识

顶点的次　　以点 $v$ 为端点的边数称为 $v$ 的次，记为 $\deg(v)$ ，或简记为 $d(v)$

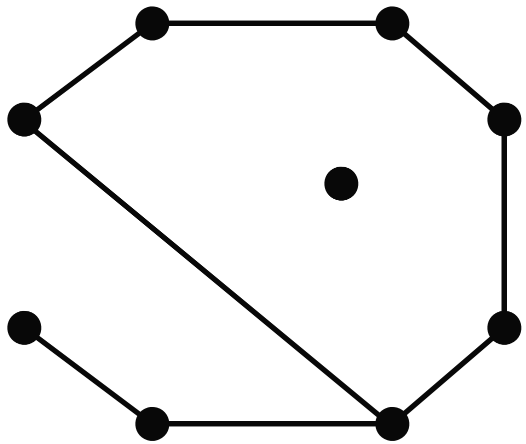
次为**0**的点称为***孤立点***，次为**1**的点称为***悬挂点***，连接悬挂点的边称为***悬挂边***，次为奇数的点称为***奇点***，次为偶数的点称为***偶点***

在有向图中，以 $v$ 为始点的边数称为 $v$ 的***出次***，用 $d^{+}(v)$ 表示，以 $v$ 为终点的边数称为 $v$ 的***入次***，用 $d^{-}(v)$ 表示

# 网络模型的基本知识

定理1　任何图中，顶点次数总和等于边数的2倍

定理2　任何图中，奇点的个数为偶数个



顶点次数总和等于16

边数等于8

奇点的个数为2个

定理1显然，下面证明定理2：根据定理1

奇点次数总和 ＋ 偶点次数总和 ＝ 偶数

若有奇数个奇点，其次数总和为奇数，上式不成立

# 网络模型的基本知识

## 子图

对于图 $G=(V,E)$，如果 $E'$ 是 $E$ 的子集，$V'$是 $V$ 的子集，并且 $E'$ 中的边仅与 $V'$ 中的顶点相关联，则称 $G'=(V',E')$ 是 $G$ 的子图，特别是，若 $V'=V$，则称 $G'$ 为 $G$ 的生成子图。

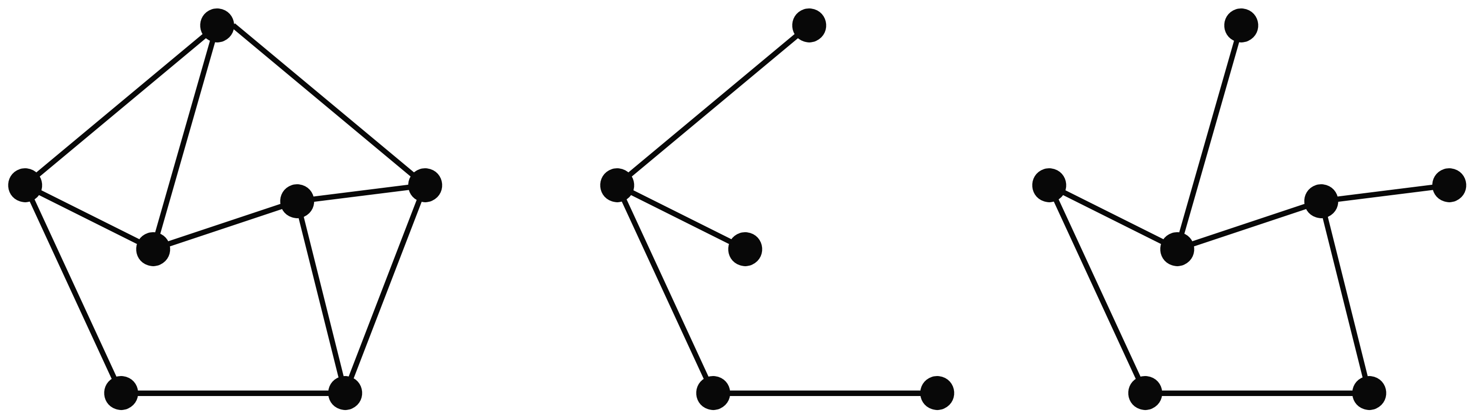


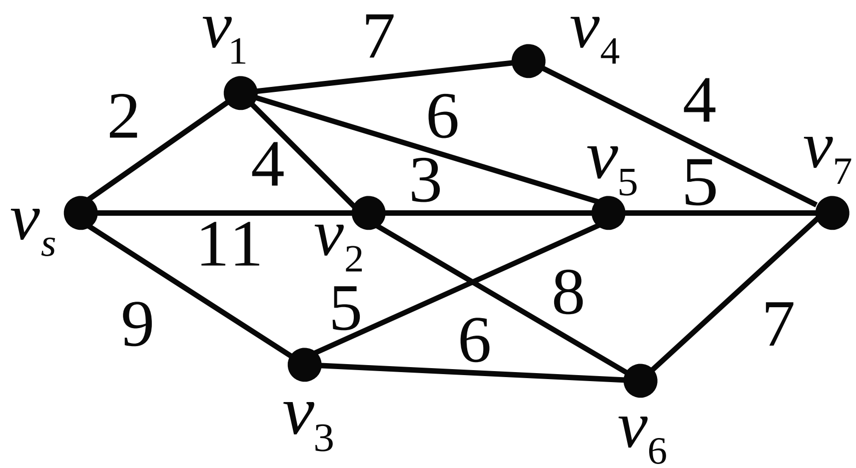
图　　　　子图　　　　生成子图

# 网络模型的基本知识

网络（赋权图）

点或边带有数值（权）的图称为网络

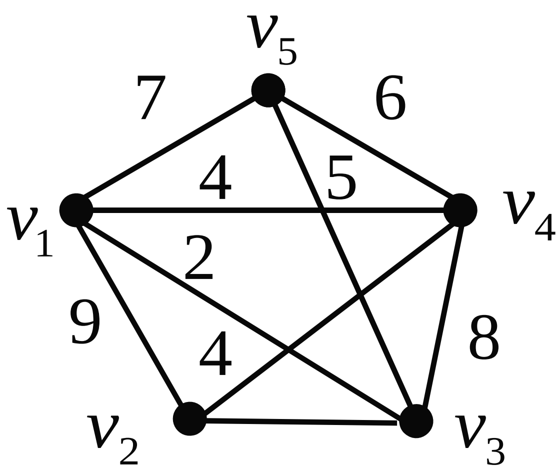
例 某物资供应站 $v_s$ 与用户之间的公路网络




上图是无向网络，如果通过管道输送输油或气体，每个边有方向，构成有向网络

# 网络模型的基本知识

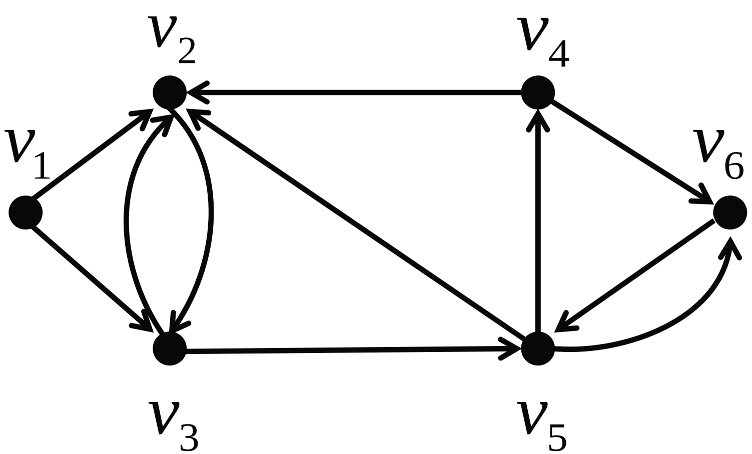
图的矩阵表示




$$\Leftrightarrow$$

$$\begin{array}{cc} \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 0 & 9 & 2 & 4 & 7 \\ 9 & 0 & 3 & 4 & \infty \\ 2 & 3 & 0 & 8 & 5 \\ 4 & 4 & 8 & 0 & 6 \\ 7 & \infty & 5 & 6 & 0 \end{bmatrix} \\ & \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \end{array}$$

# 网络模型的基本知识




$$\Leftrightarrow \quad \begin{array}{c c}
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} &
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} \\
& \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array}
\end{array}$$

# 本讲内容

一、网络模型的基本知识

**二、最小生成树算法**

三、最短路径算法

# 最小生成树算法

- 最小生成树算法是用来连接一个网络的所有节点，使树上边的总长度达到最小。

令N={1, 2, …, n}是网络中节点的集合，定义：

$C_k$ 表示在第k步时已经连接起来的节点的集合

$\overline{C_k}$ 表示在第k步以后需要连接的节点的集合

# 最小生成树算法

第0步　令 $C_0=\varnothing,\overline{C_0}=N.$

第1步　从 $\overline{C_0}$ 中的任意一个节点 $i$ 开始，令 $C_1=\{i\}$ 那么 $\overline{C_1}=N-\{i\}$。假定k=2.

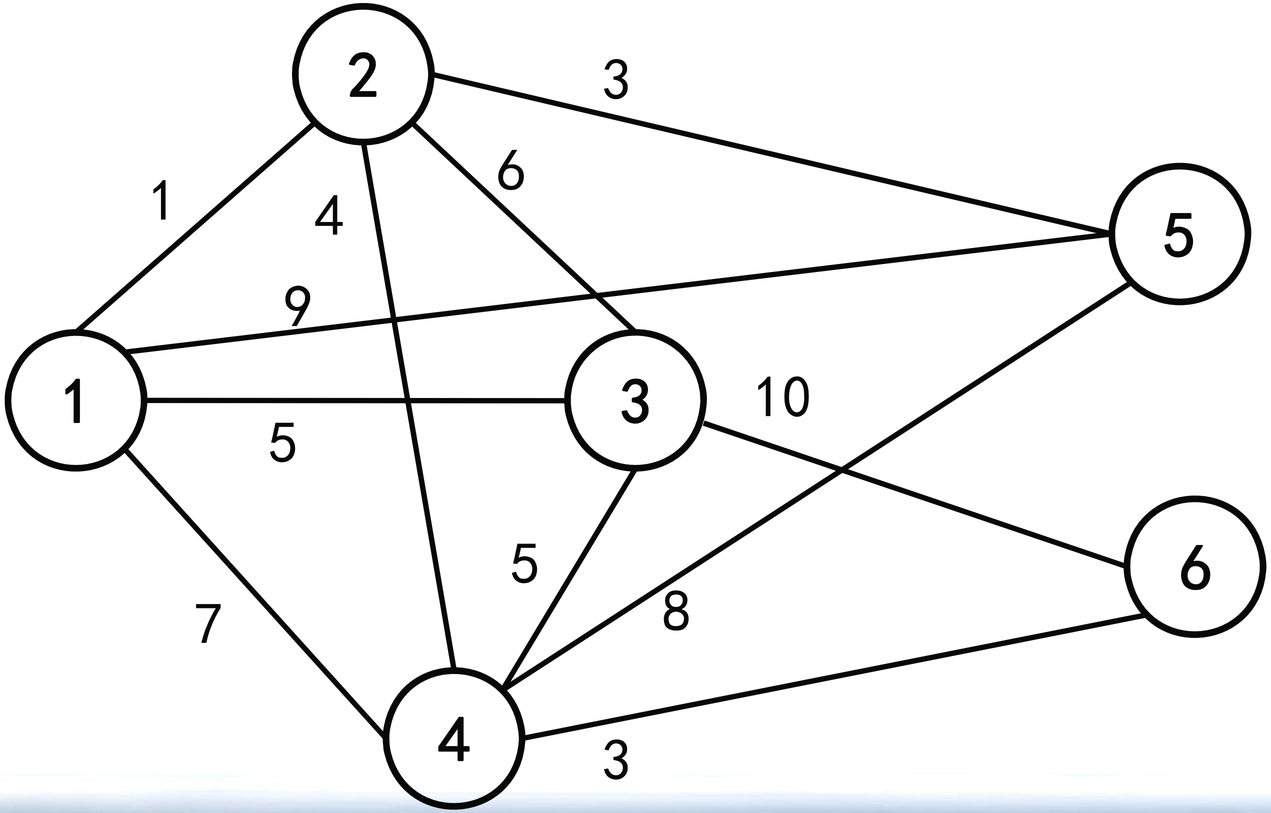
一般的第k步　在还没有连接的节点集合 $\overline{C_{k-1}}$ 中选择一个节点 $j^*$，使得 $j^*$ 到 $C_{k-1}$ 中某个节点之间的弧长最小。然后将 $j^*$ 放入 $C_{k-1}$，从 $\overline{C_{k-1}}$ 中删除 $j^*$，即

$$C_k=C_{k-1}+\{j^*\},\overline{C_k}=\overline{C_{k-1}}-\{j^*\}$$

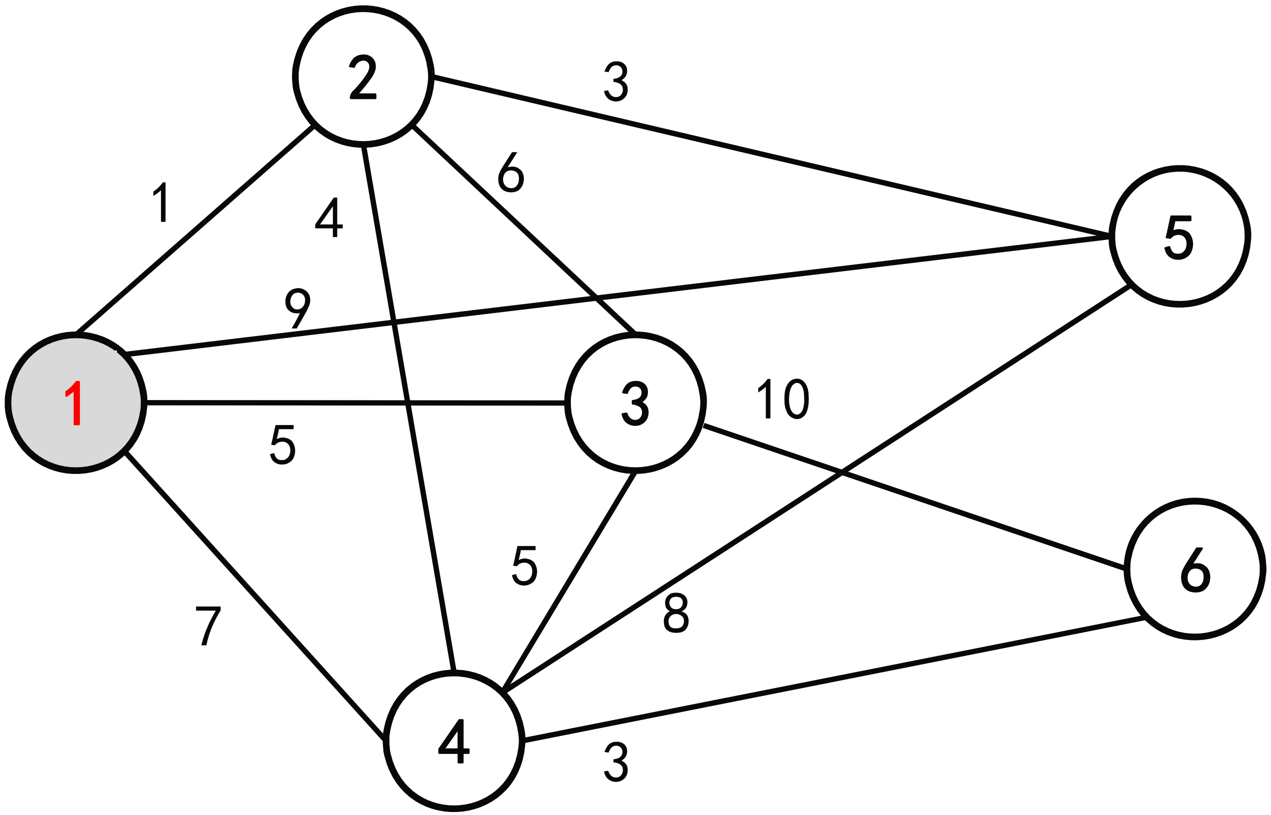
当未连接节点的集合是空集，停止，否则重复计算。

# 最小生成树算法

- 例1：某市广播电视局计划给5个新居民区提供有线电视服务，确定一种最经济的电缆铺设方案，使得5个小区可连接起来。

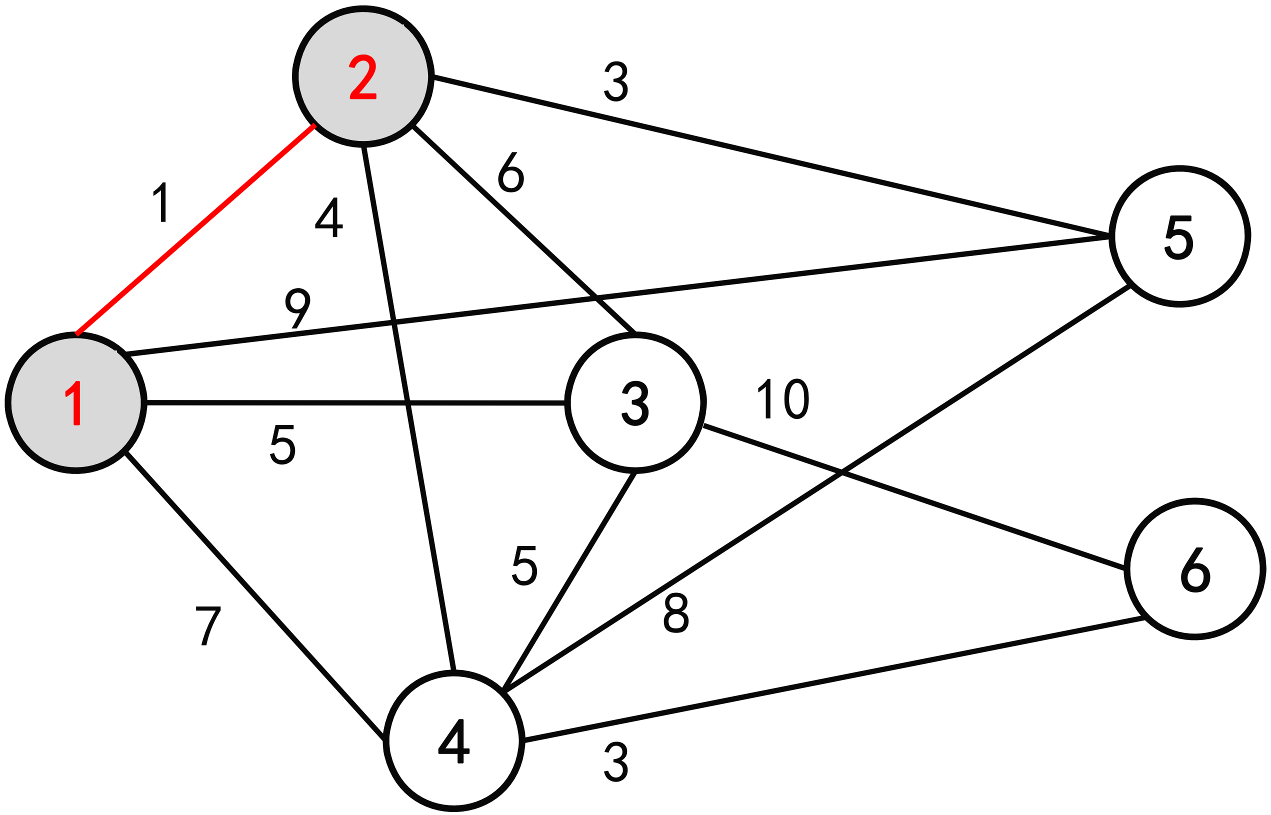


# 最小生成树算法



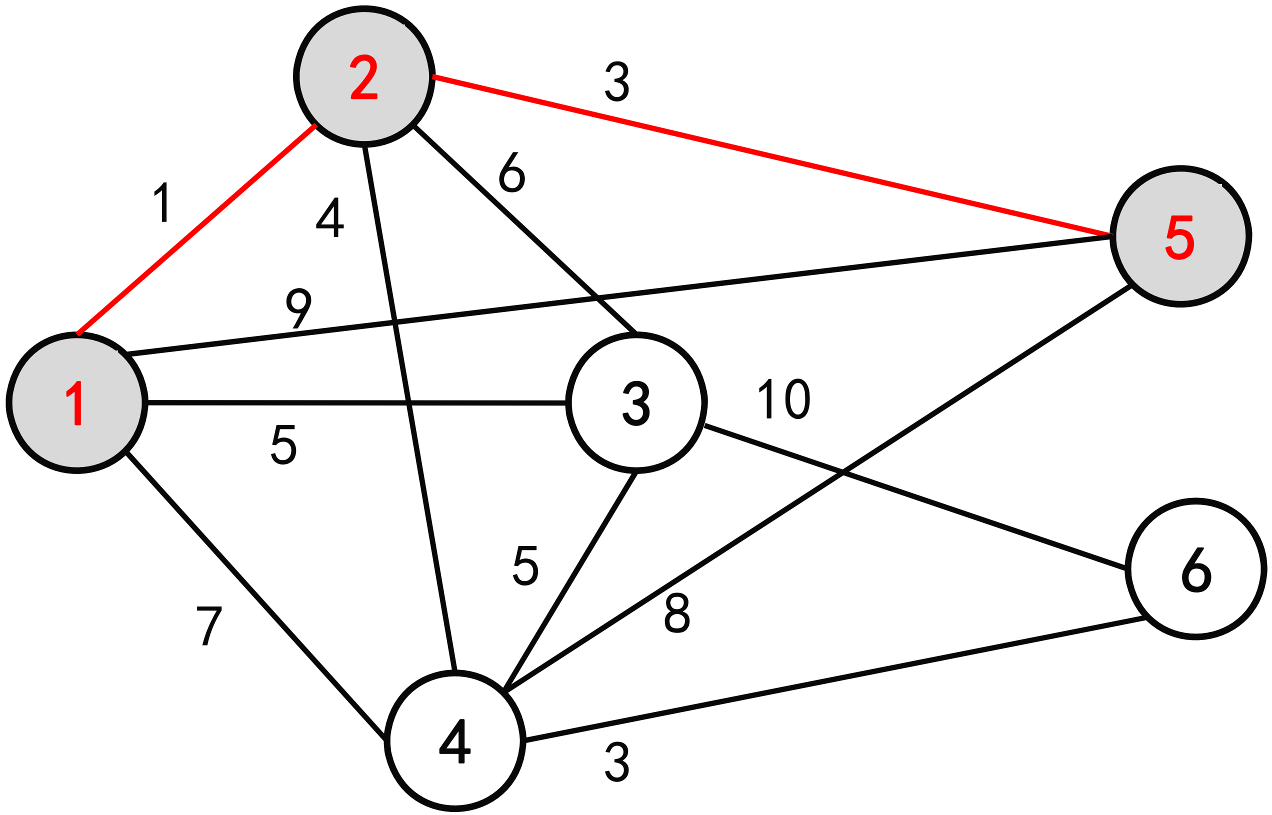

$$C_1 = \{1\}, \overline{C_1} = \{2, 3, 4, 5, 6\}$$

# 最小生成树算法



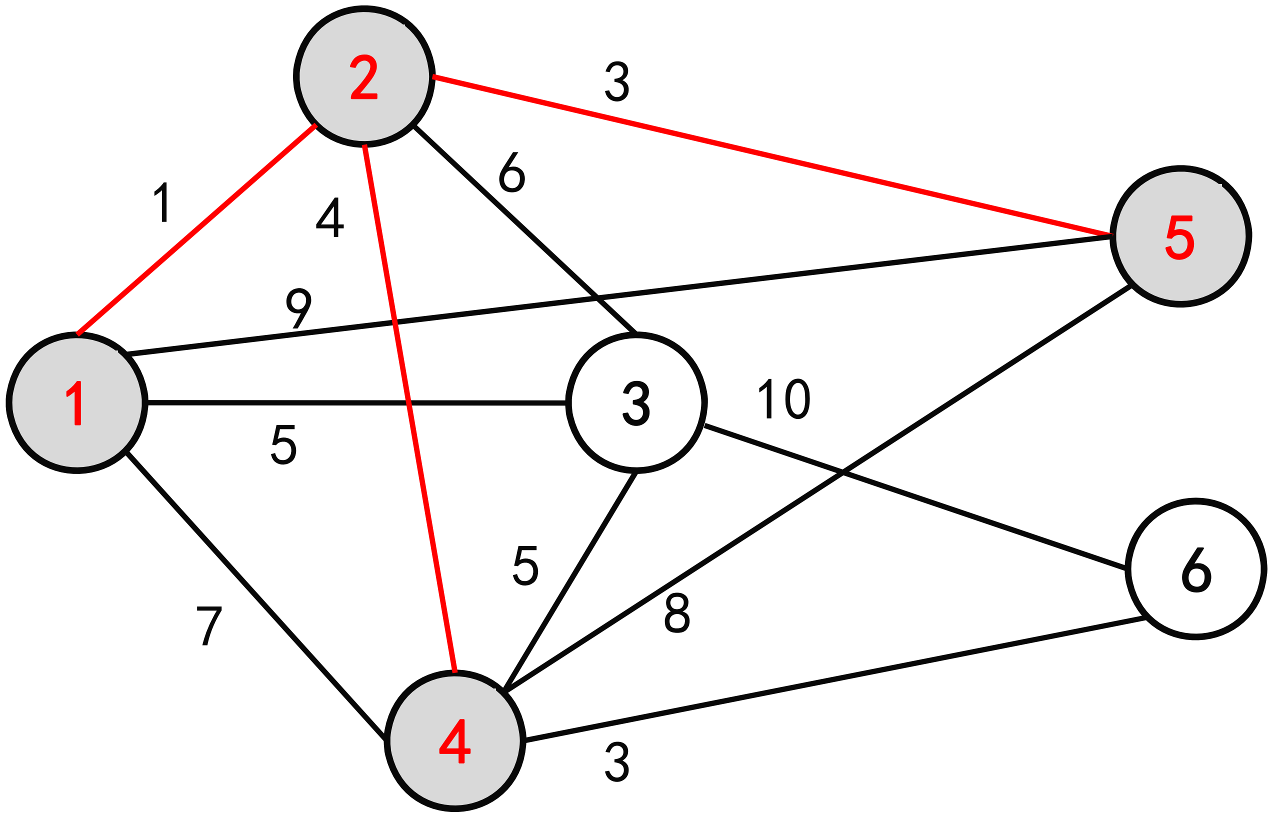

$$C_2 = \{1, 2\}, \overline{C_2} = \{3, 4, 5, 6\}$$

# 最小生成树算法



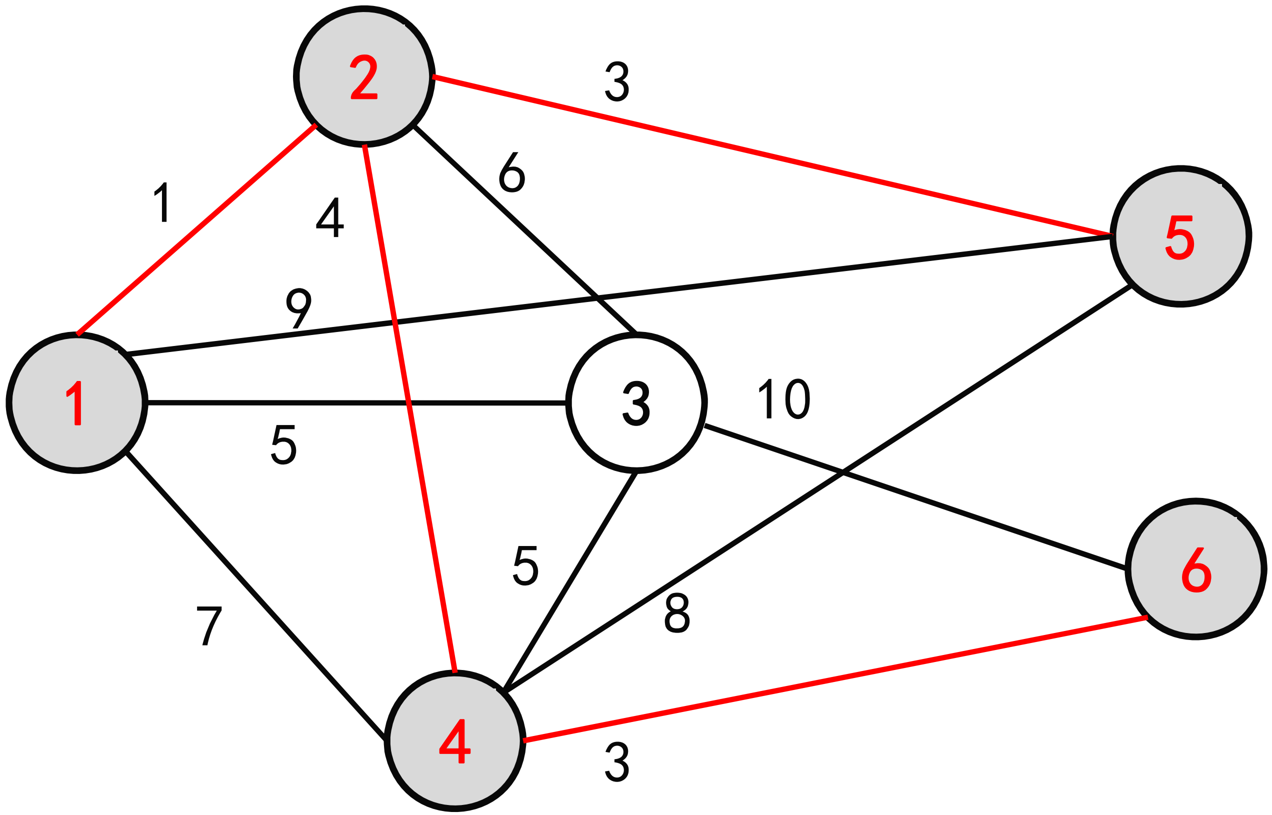

$$C_3 = \{1, 2, 5\}, \overline{C_3} = \{3, 4, 6\}$$

# 最小生成树算法



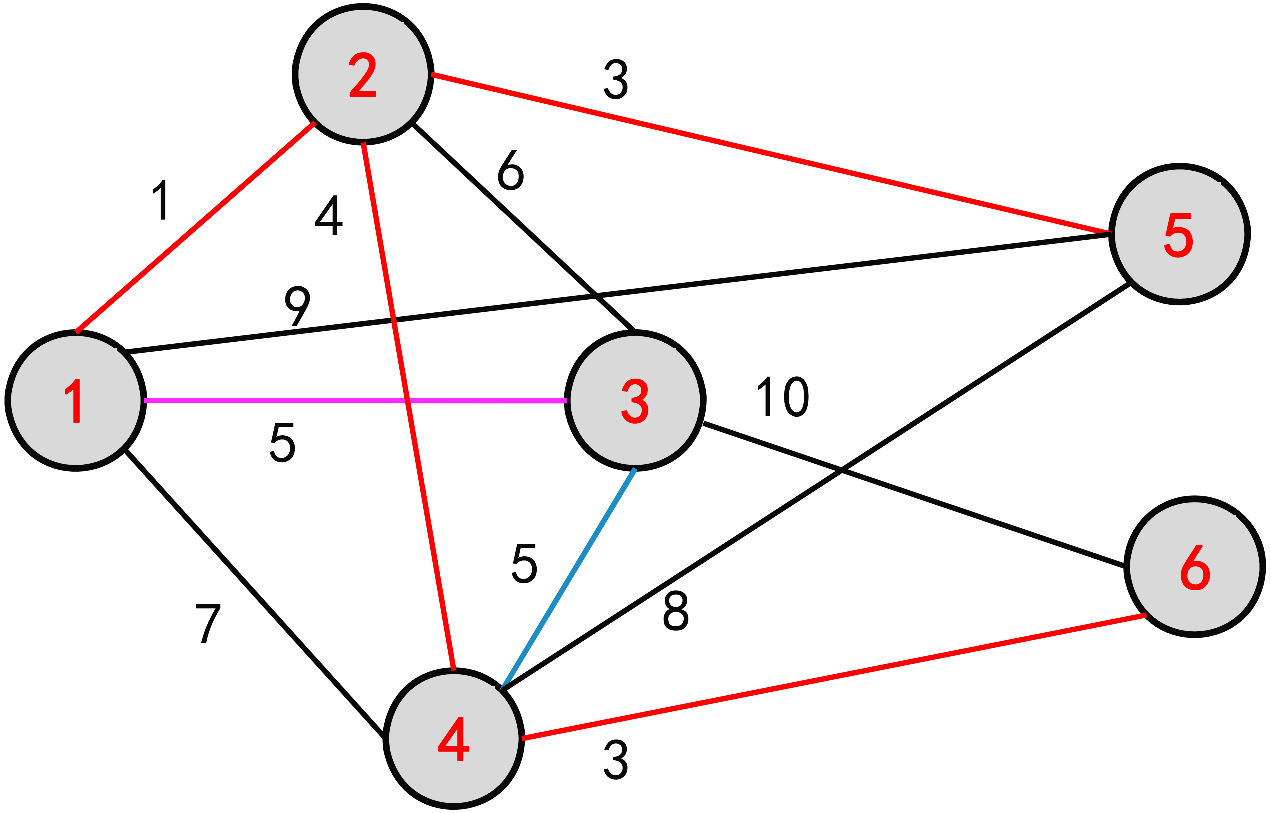

$$C_4 = \{1, 2, 5, 4\}, \overline{C_4} = \{3, 6\}$$

# 最小生成树算法




$$C_5 = \{1, 2, 5, 4, 6\}, \overline{C_5} = \{3\}$$

# 最小生成树算法




$$C_6 = \{1,2,5,4,6,3\}, \overline{C_6} = \varnothing$$

# 本讲内容

一、网络模型的基本知识

二、最小生成树算法

**三、最短路径算法**

# 最短路径算法

- Dijkstra算法，可求网络中从源点到其他任何一个节点的最短路径。

- Floyd算法，可求网络中任意两个节点之间的最短路径。

# 最短路径算法

- Dijkstra算法

用$u_i$表示从源点1到节点i的最短距离，定义$d_{ij}$为弧(i,j)的长度，算法可给出节点j的标号为

$$[u_j, i] = [u_i + d_{ij}, i], d_{ij} \geq 0$$

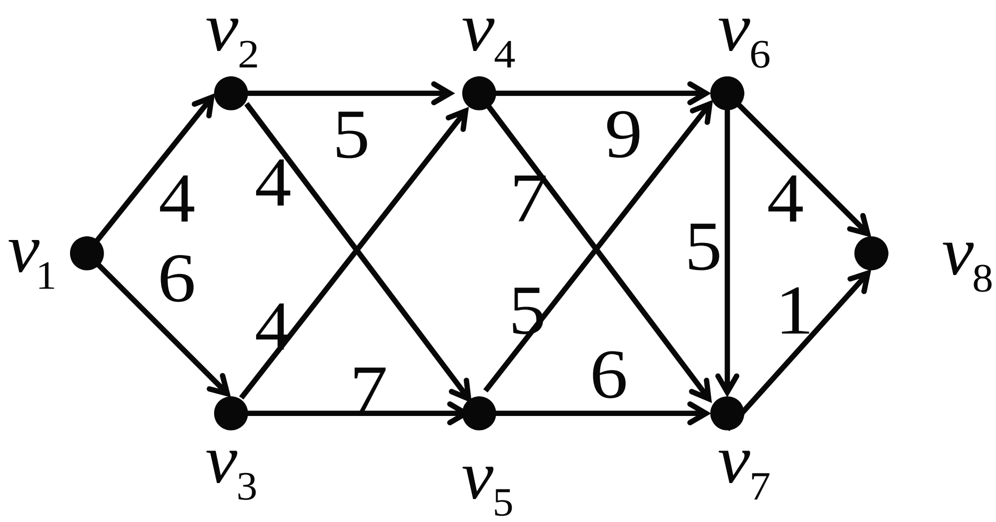
第0步 对源点进行永久的标号[0,-]，令i=1。

第1步 (a)计算节点i有边相连的每一个节点j(j没被永久标号)的暂时标号[$u_i$+$d_{ij}$,i]。如果节点j已通过其他点有暂时标号，取最小值为其暂时标号。

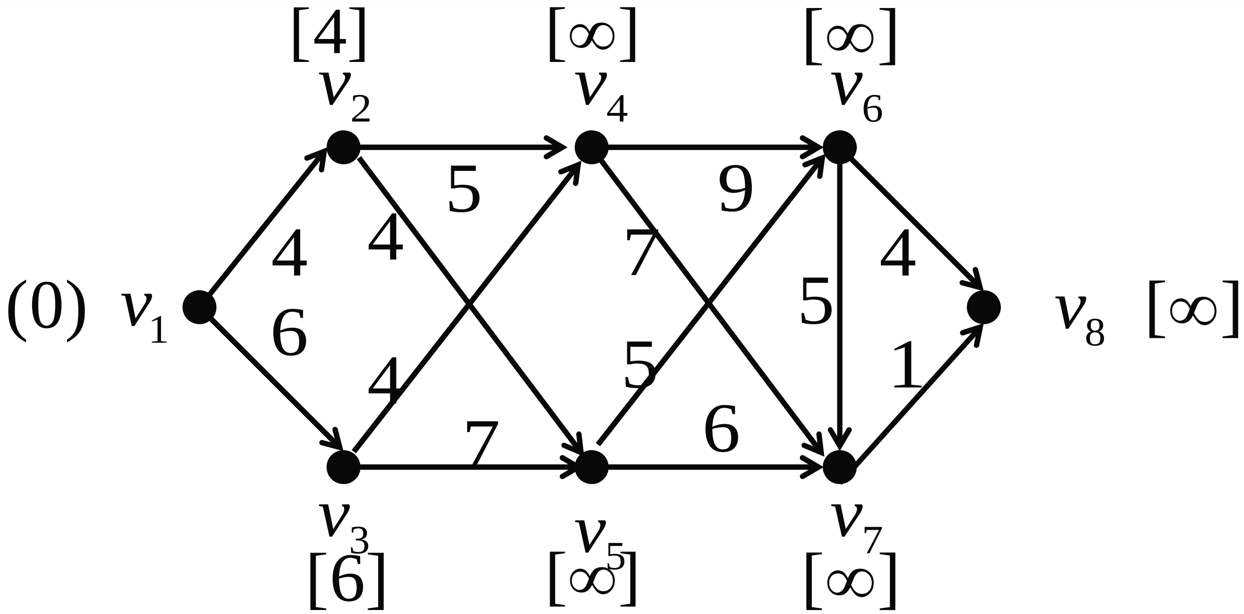
(b) 从所有暂时标号节点中选择具有最短距离的标号[$u_r$,s]。如果所有节点都是永久的标号，停止；否则令i=r，执行第i步。

# 最短路径算法

➢ 例2：求城市1到城市8的最短距离。



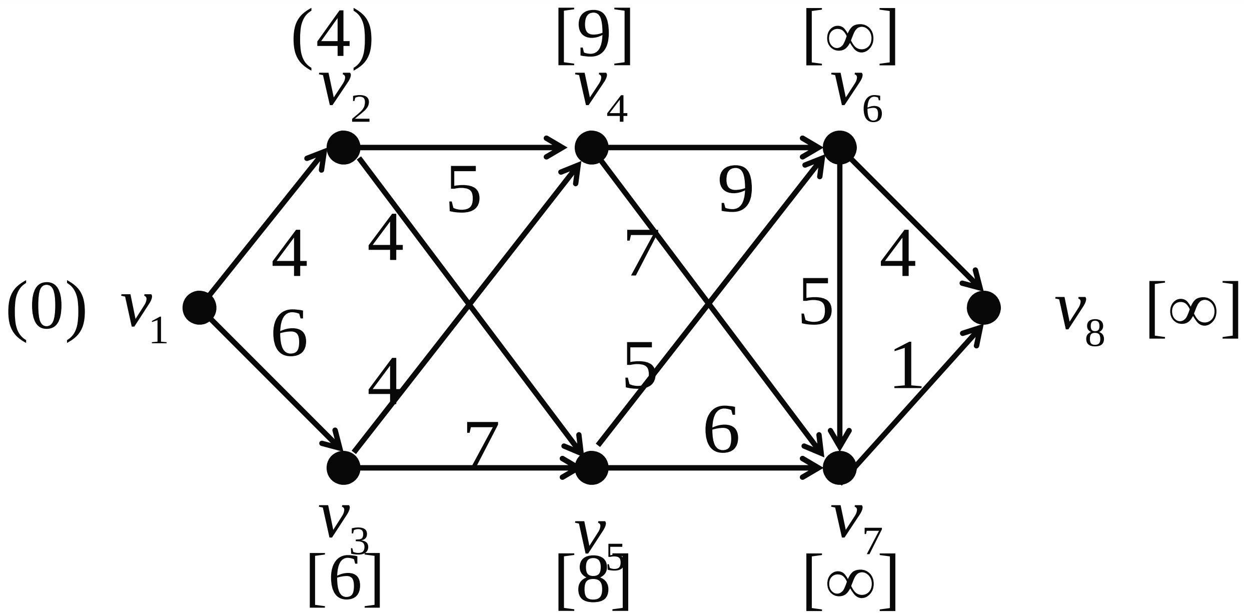
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 暂时 |
| 3 | [6,1] | 暂时 |

永久

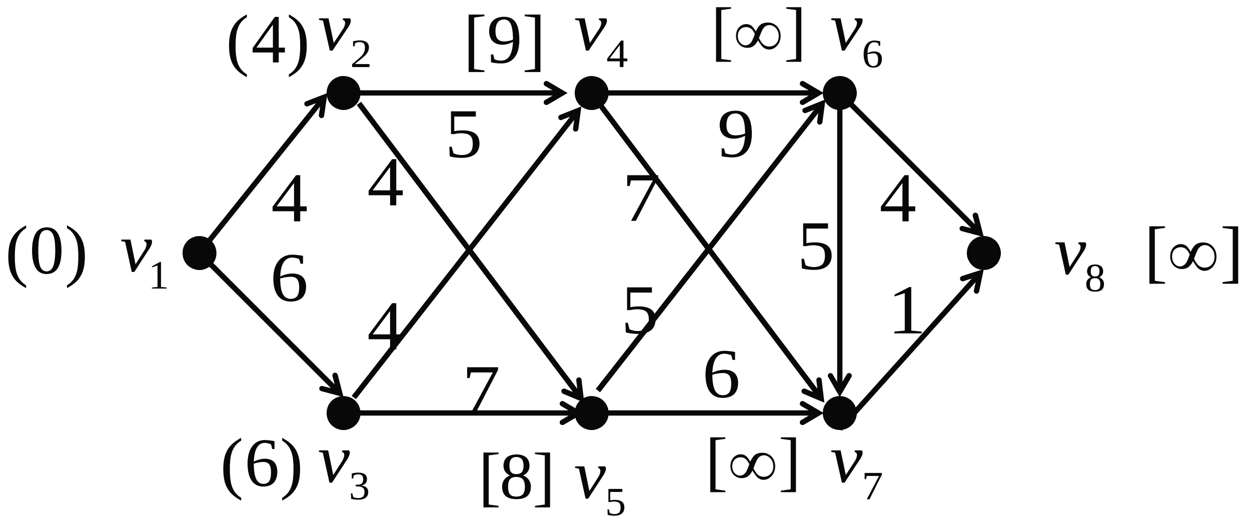
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 暂时 |
| 4 | [9,2] | 暂时 |
| 5 | [8,2] | 暂时 |

$\boxed{\text{永久}}$

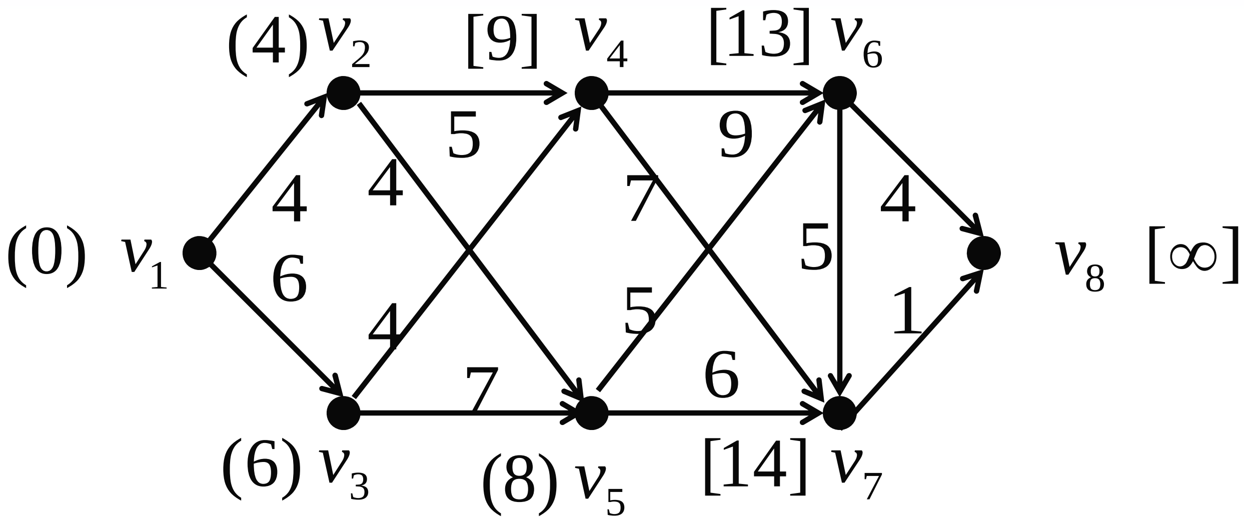
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 暂时 |
| 5 | [8,2] | 暂时 |

永久

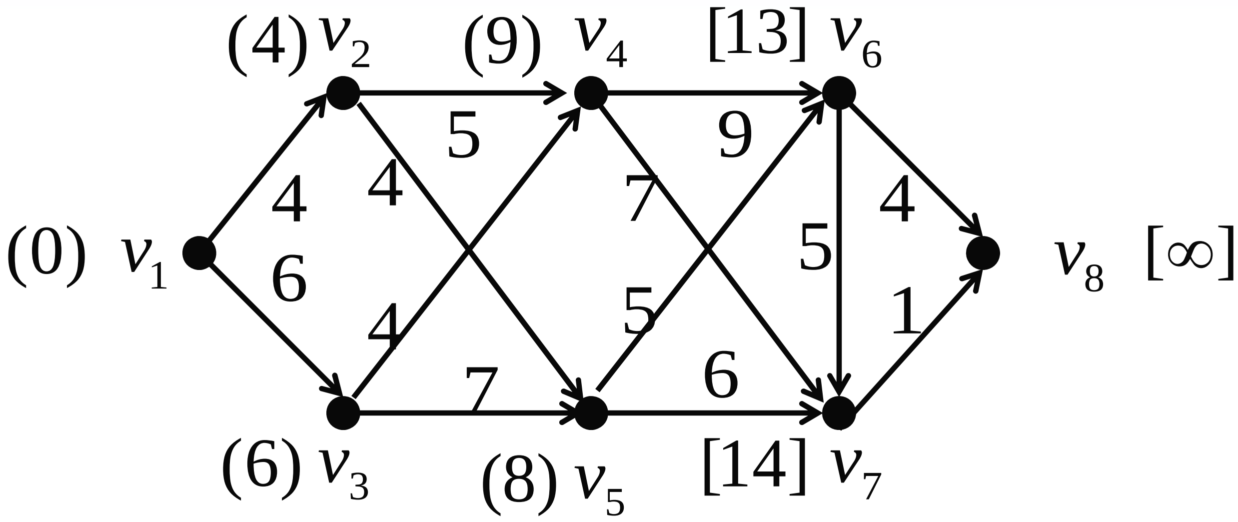
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 暂时 |
| 5 | [8,2] | 永久 |
| 6 | [13,5] | 暂时 |
| 7 | [14,5] | 暂时 |

永久

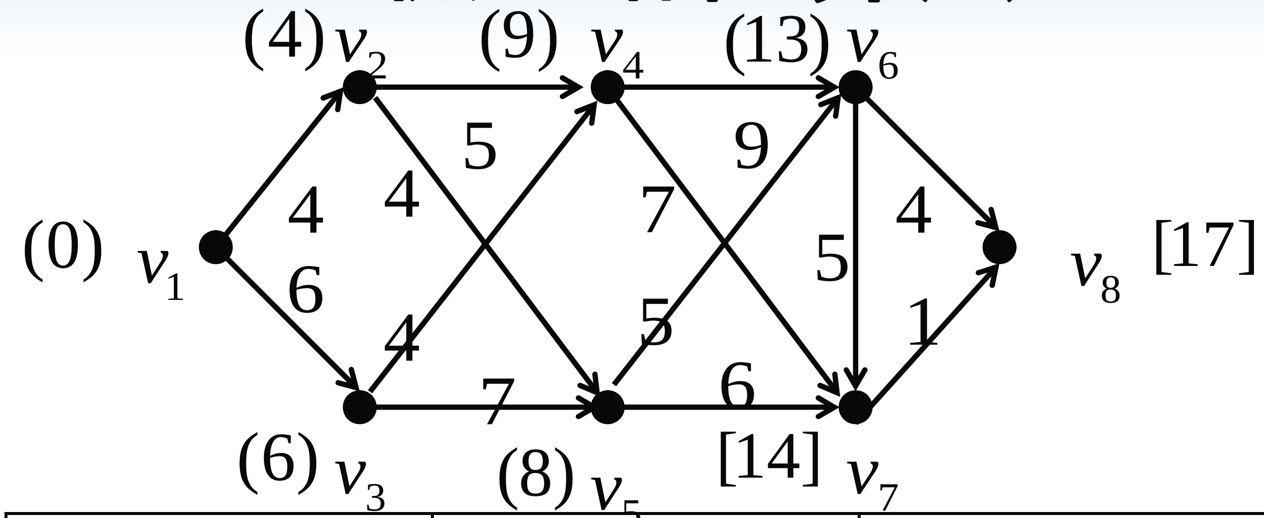
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 永久 |
| 5 | [8,2] | 永久 |
| 6 | [13,5] | 暂时 |
| 7 | [14,5] | 暂时 |

永久

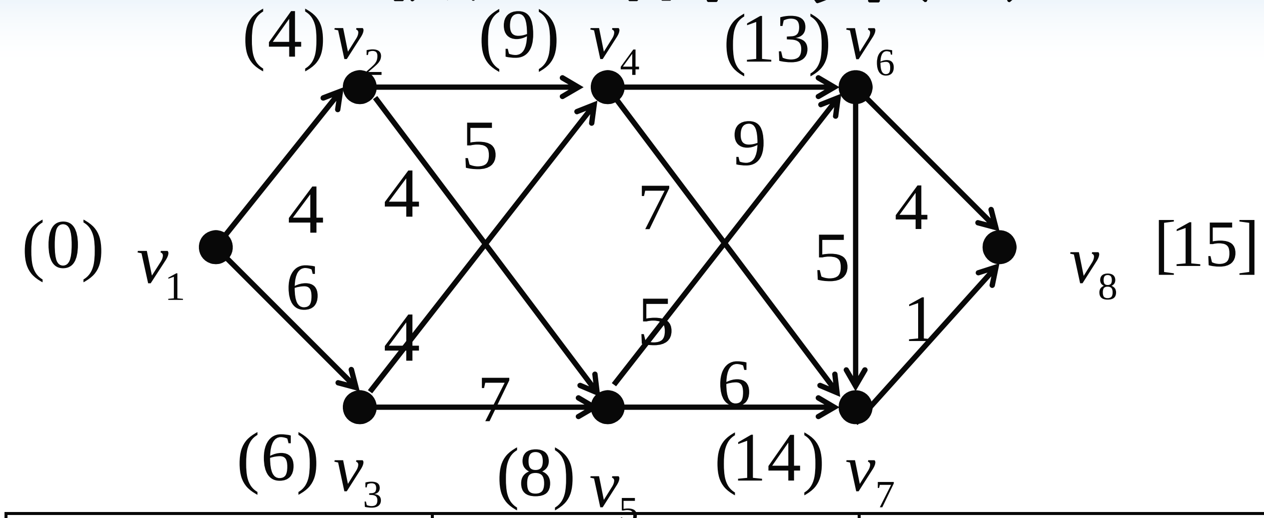
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
| --- | --- | --- |
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 永久 |
| 5 | [8,2] | 永久 |
| 6 | [13,5] | 永久 |
| 7 | [14,5] | 暂时 |
| 8 | [17,6] | 暂时 |

**永久**

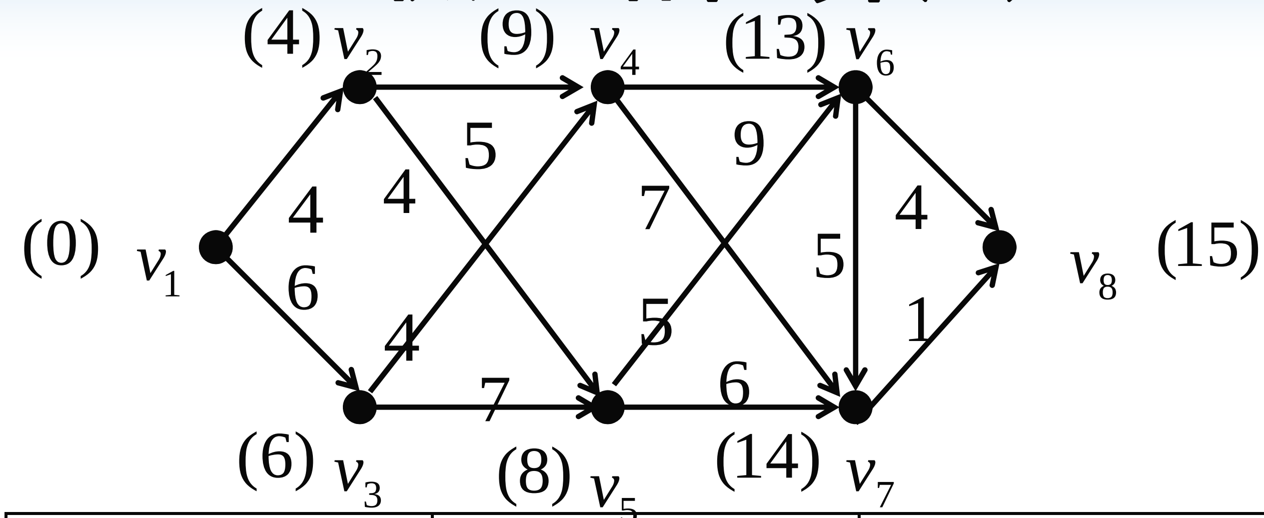
# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 永久 |
| 5 | [8,2] | 永久 |
| 6 | [13,5] | 永久 |
| 7 | [14,5] | 永久 |
| 8 | [15,7] | 暂时 |

永久

# 最短路径算法




| 节点 | 标号 | 标号的状态 |
|---|---|---|
| 1 | [0,--] | 永久 |
| 2 | [4,1] | 永久 |
| 3 | [6,1] | 永久 |
| 4 | [9,2] | 永久 |
| 5 | [8,2] | 永久 |
| 6 | [13,5] | 永久 |
| 7 | [14,5] | 永久 |
| 8 | [15,7] | 永久 |

# 最短路径算法

- Floyd算法

算法首先将n个节点的网络表示成一个n行n列的矩阵，矩阵中的元素(i,j)表示从节点i到节点j的距离$d_{ij}$，如果i,j之间没有边相连，那么相应的元素就是无穷。

给定3个节点i,j,k，以及他们之间的距离，如满足

$$d_{ik} + d_{kj} < d_{ij}$$

那么从i经过k到j更短。

# 最短路径算法

## ➢Floyd算法

第0步  定义初始的距离矩阵$D_0$和节点序列矩阵$S_0$，对角线用--表示，令k=1.

$D_0 =$

| | 1 | 2 | … | j | … | n |
|---|---|---|---|---|---|---|
| 1 | -- | $d_{12}$ | … | $d_{1j}$ | … | $d_{1n}$ |
| 2 | $d_{21}$ | -- | …. | $d_{2j}$ | …. | $d_{2n}$ |
| … | … | … | … | … | … | … |
| i | $d_{i1}$ | $d_{i2}$ | … | $d_{ij}$ | … | $d_{in}$ |
| … | … | … | … | … | … | … |
| n | $d_{n1}$ | $d_{n2}$ | … | $d_{nj}$ | … | -- |

$S_0 =$

| | 1 | 2 | … | j | … | n |
|---|---|---|---|---|---|---|
| 1 | -- | 2 | … | j | … | n |
| 2 | 1 | -- | …. | j | …. | n |
| … | … | … | … | … | … | … |
| i | 1 | 2 | … | j | … | n |
| … | … | … | … | … | … | … |
| n | 1 | 2 | … | j | … | -- |

# 最短路径算法

- Floyd算法

一般的第k步　令第k行和第k列为枢轴行和枢轴列。对于矩阵$D_{k-1}$中的每一个元素$d_{ij}$做三重操作，如满足条件：

$$d_{ik}+d_{kj}<d_{ij}\quad (i\neq k, j\neq k, i\neq j)$$

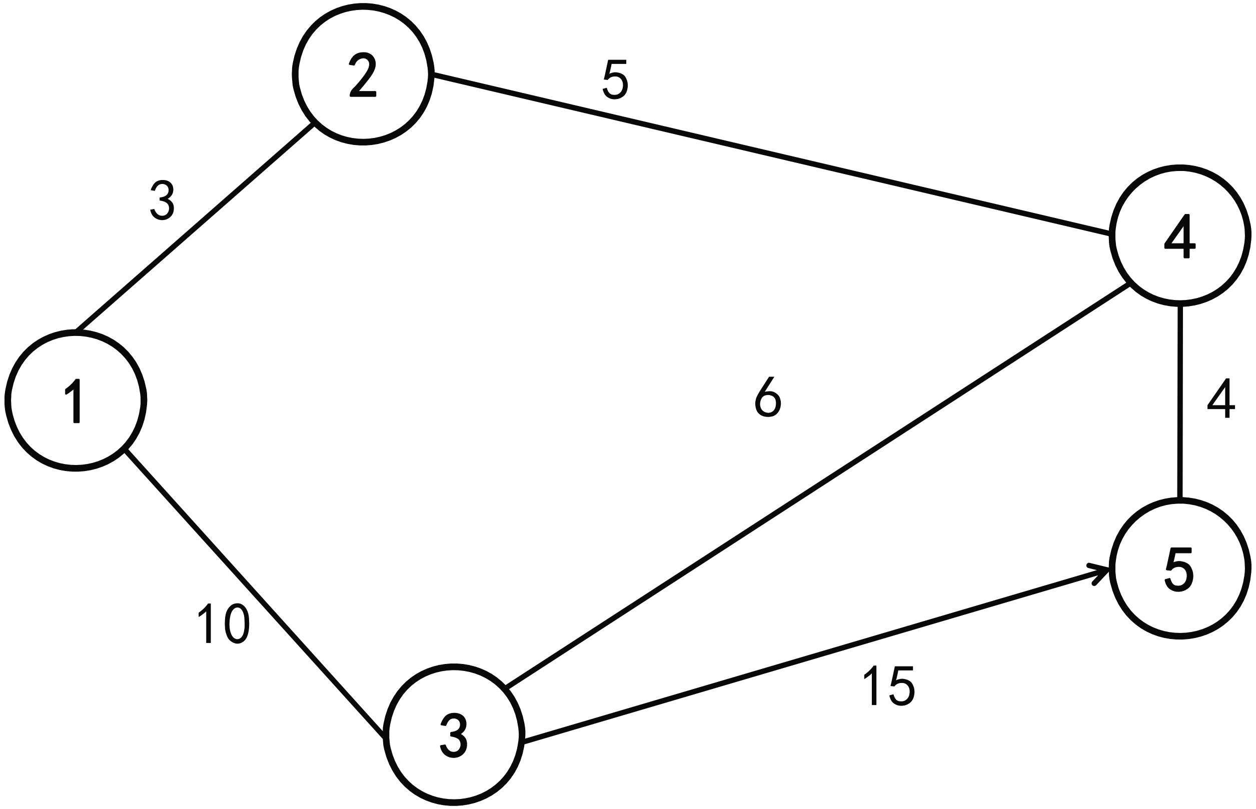
进行下面的转化：

(a)用$d_{ik}+d_{kj}$代替矩阵$D_{k-1}$中的元素$d_{ij}$，从而得到矩阵$D_k$；

(b)用k代替矩阵$S_{k-1}$中的元素$s_{ij}$，从而得到矩阵$S_k$。令k=k+1，如果k=n+1，停止；否则，重复第k步。

# 最短路径算法

➢ 例3：求任意两个节点之间的最短距离。



# 最短路径算法

- 迭代0

矩阵$D_0$和$S_0$代表初始的网络，除了$d_{53}=\infty$ 外，$D_0$是对称的。

$D_0=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | ∞ | ∞ |
| 2 | 3 | -- | ∞ | 5 | ∞ |
| 3 | 10 | ∞ | -- | 6 | 15 |
| 4 | ∞ | 5 | 6 | -- | 4 |
| 5 | ∞ | ∞ | ∞ | 4 | -- |

$S_0=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | 4 | 5 |
| 2 | 1 | -- | 3 | 4 | 5 |
| 3 | 1 | 2 | -- | 4 | 5 |
| 4 | 1 | 2 | 3 | -- | 5 |
| 5 | 1 | 2 | 3 | 4 | -- |

# 最短路径算法

- 迭代1

令k=1。

$D_1=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | ∞ | ∞ |
| 2 | 3 | -- | **13** | 5 | ∞ |
| 3 | 10 | **13** | -- | 6 | 15 |
| 4 | ∞ | 5 | 6 | -- | 4 |
| 5 | ∞ | ∞ | ∞ | 4 | -- |

$S_1=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | 4 | 5 |
| 2 | 1 | -- | **1** | 4 | 5 |
| 3 | 1 | **1** | -- | 4 | 5 |
| 4 | 1 | 2 | 3 | -- | 5 |
| 5 | 1 | 2 | 3 | 4 | -- |

# 最短路径算法

- 迭代2

令k=2。

$D_2 =$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | **8** | ∞ |
| 2 | 3 | -- | 13 | 5 | ∞ |
| 3 | 10 | 13 | -- | 6 | 15 |
| 4 | **8** | 5 | 6 | -- | 4 |
| 5 | ∞ | ∞ | ∞ | 4 | -- |

$S_2 =$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | **2** | 5 |
| 2 | 1 | -- | 1 | 4 | 5 |
| 3 | 1 | 1 | -- | 5 | 5 |
| 4 | **2** | 2 | 3 | -- | 5 |
| 5 | 1 | 2 | 3 | 4 | -- |

# 最短路径算法

- 迭代3

令k=3。

$D_3 =$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | 8 | **25** |
| 2 | 3 | -- | 13 | 5 | **28** |
| 3 | 10 | 13 | -- | 6 | 15 |
| 4 | 8 | 5 | 6 | -- | 4 |
| 5 | ∞ | ∞ | ∞ | 4 | -- |

$S_3 =$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | 2 | **3** |
| 2 | 1 | -- | 1 | 4 | **3** |
| 3 | 1 | 1 | -- | 4 | 5 |
| 4 | 2 | 2 | 3 | -- | 5 |
| 5 | 1 | 2 | 3 | 4 | -- |

# 最短路径算法

- 迭代4

令k=4。

$D_4=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | 8 | **12** |
| 2 | 3 | -- | **11** | 5 | **9** |
| 3 | 10 | **11** | -- | 6 | **10** |
| 4 | 8 | 5 | 6 | -- | 4 |
| 5 | **12** | **9** | **10** | 4 | -- |

$S_4=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | 2 | **4** |
| 2 | 1 | -- | **4** | 4 | **4** |
| 3 | 1 | **4** | -- | 4 | **4** |
| 4 | 2 | 2 | 3 | -- | 5 |
| 5 | **4** | **4** | **4** | 4 | -- |

# 最短路径算法

$D_5=$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 3 | 10 | 8 | 12 |
| 2 | 3 | -- | 11 | 5 | 9 |
| 3 | 10 | 11 | -- | 6 | 10 |
| 4 | 8 | 5 | 6 | -- | 4 |
| 5 | 12 | 9 | 10 | 4 | -- |

$S_5=$

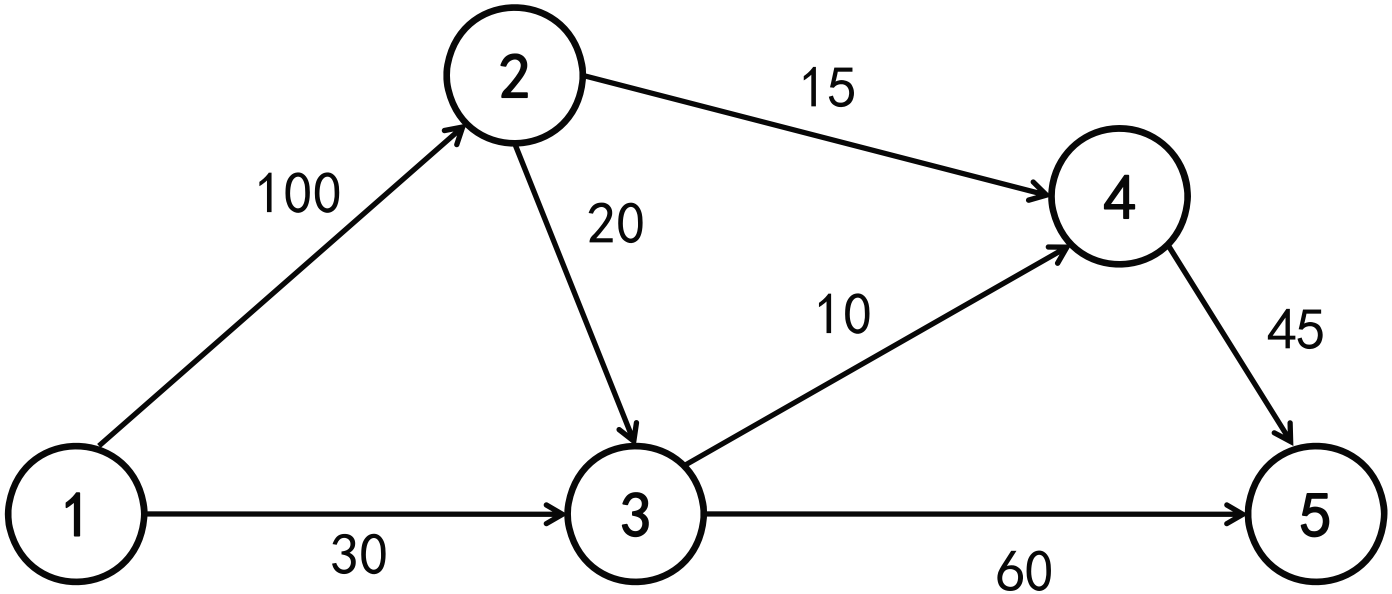
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | -- | 2 | 3 | 2 | 4 |
| 2 | 1 | -- | 4 | 4 | 4 |
| 3 | 1 | 4 | -- | 4 | 4 |
| 4 | 2 | 2 | 3 | -- | 5 |
| 5 | 4 | 4 | 4 | 4 | -- |

$d_{15}=12$.

$s_{15}=4, s_{14}=2, s_{12}=2$；路径1->2->4->5

# 本节作业

- 求城市1到城市5的最短路径



谢谢 ！