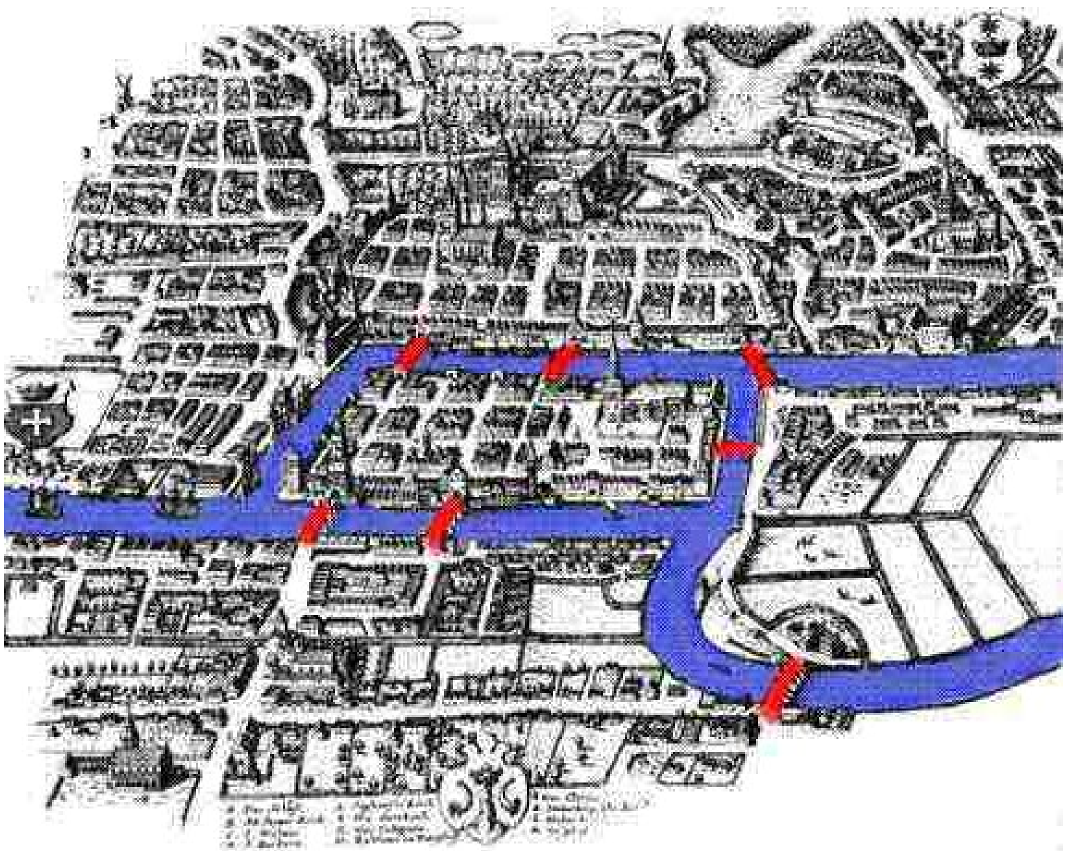
第七章 图与网络

- 图与网络的基本知识
- 树
- 最短路问题
- 最大流问题
- 最小费用流问题

哥尼斯堡七桥问题



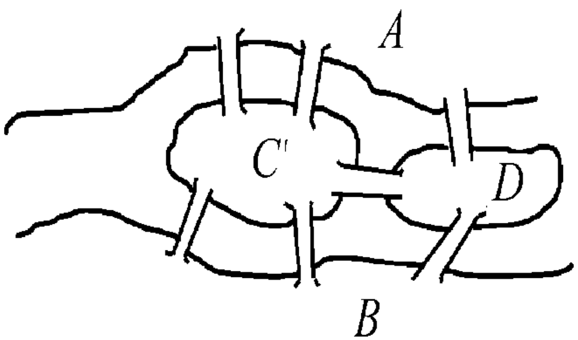
18世纪，欧洲有一个风景秀丽的小城哥尼斯堡（今俄罗斯加里宁格勒）。那里的普莱格尔河上有七座桥，将河中的两个岛和河岸连结。城中的居民经常沿河过桥散步，于是提出了一个问题：一个人怎样才能一次走遍七座桥，每座桥只走过一次，最后回到出发点？大家都试图找出问题的答案，但是谁也解决不了这个问题………

这就是哥尼斯堡七桥问题。

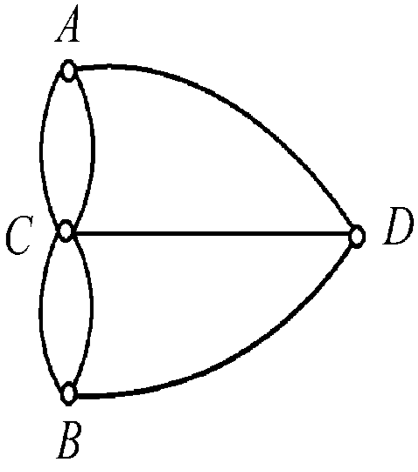
哥尼斯堡七桥问题

Leonhard Euler
(1707-1783)
瑞士数学家

他把这个难题进行了转换：把二岸和小岛缩成点，桥化为边，于是“七桥问题”就等价于下图（**B**）中所画图形的一笔画问题了。



(A)



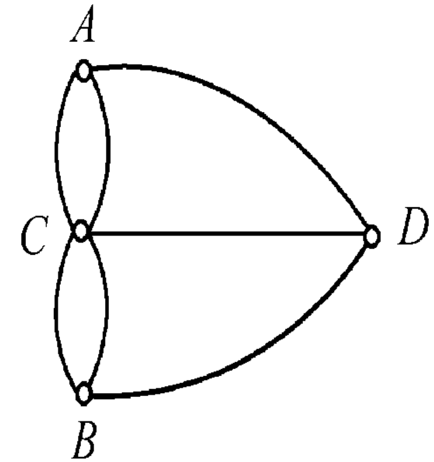
(B)

哥尼斯堡七桥问题

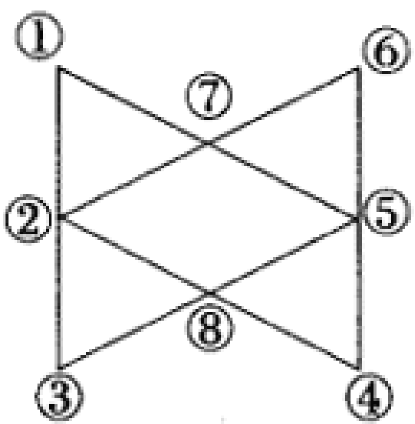
经过研究，欧拉发现了一笔画的规律：

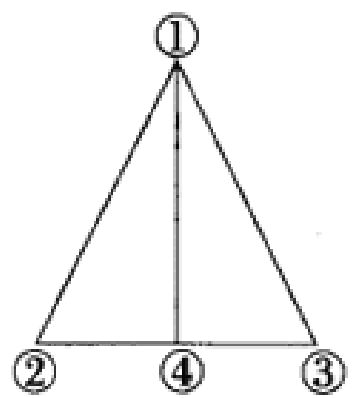
i) 能一笔画的图形必须是连通图

ii) 全部由偶点组成的连通图
　　只有两个奇点其余均为偶点的连通图

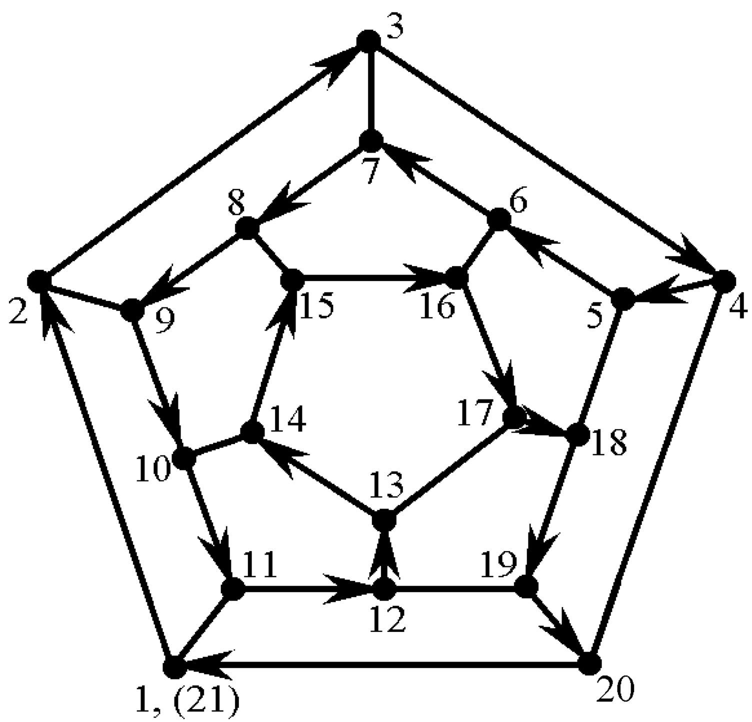


(B)





哈密顿圈问题



1857年，英国数学家Hamiltion发明了一种游戏。他用一个实心正12面体象征地球，正12面体的20个顶点分别表示世界上20座名城，要求游戏者从任一城市出发，寻找一条可经由每个城市一次且仅一次再回到原出发点的路，也称“环球旅行”问题。

农夫过河

有位农夫，携带一匹狼、一只羊和一挑草要过一条小河。河中只有一条小船，一次摆渡农夫只能带狼、羊、草中的一样。当农夫不在场时，狼要吃羊，羊要吃草。试问农夫怎样才能将这三样东西摆渡到河对岸。至少要摆渡几次？

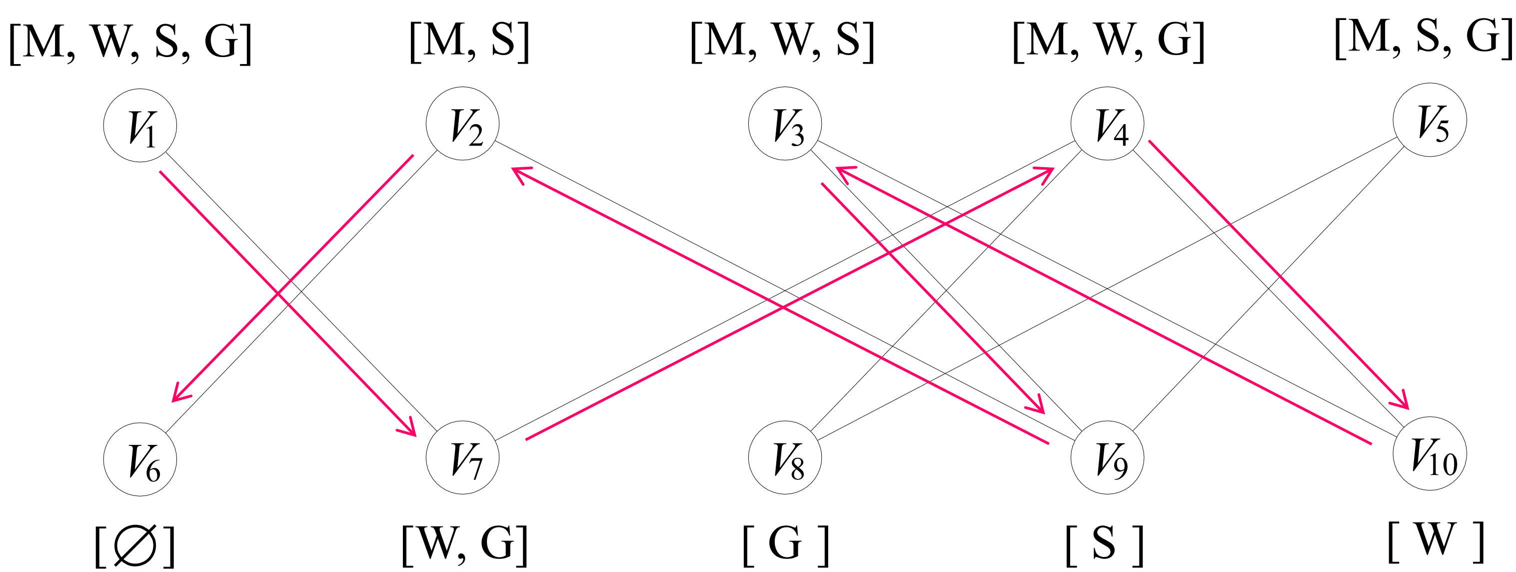
农夫过河

用M、W、S、G分别代表人、狼、羊、草。

①[M, W, S, G] [Ø]　②[M, W] [S, G]　③[M, S] [W, G]　④[M, G] [W, S]

⑤ [M, W, S] [G]　⑥[M, W, G] [S]　⑦[W, S, G] [M]　⑧[M, S, G] [W]

上面组合中②、④、⑦是不允许的。去掉这三个组合中的六个状态，那么可能的状态有10个。按照状态中是否有人存在，将它们分为两组，如图所示。



更多实用意义的问题

- 中国邮递员问题(Chinese Postman Problem, CPP)：一个邮递员负责一个地区的信件投递，他每天要从邮局出发，走遍该地区所有街道，最后返回到邮局，问如何安排送信的路线使所走路程最短？
- 旅行售货商问题(Traveling Salesman Problem, TSP)：卖货郎走遍一个区域内的所有村庄一次，使得所走路程最短?
- 要在若干个城市之间架设电网（或铺设管道），如何选择一个联系所有城市的电网（管网结构）？
- 在一个已有的道路网络（管网、通信网络等），已知每条线路（管道）的容量，网络上可通行的最大流量是多少？
- …………

第七章 图与网络

1. 图与网络的基本知识

1.1 图与网络的基本概念

自然界和人类社会中，大量的事物以及事物之间的关系可以用图形来描述。

- 电路网络、城市规划、交通运输、物资调配 ……

我们生活在一个网络社会中，从某种意义上说，现代社会是一个由各种网络所组成的复杂的网络系统。

- 计算机信息网络
- 电话通信网络
- 运输服务网络
- 能源和物资分派网络
- ……

VS 平面几何中的图

- 关心图中有多少个点
- 关心点与点之间有无连线

这里研究的图是反映对象之间关系的一种工具

- 从形形色色的具体实际问题中抽象而来
- 研究其共同的性质、规律和方法

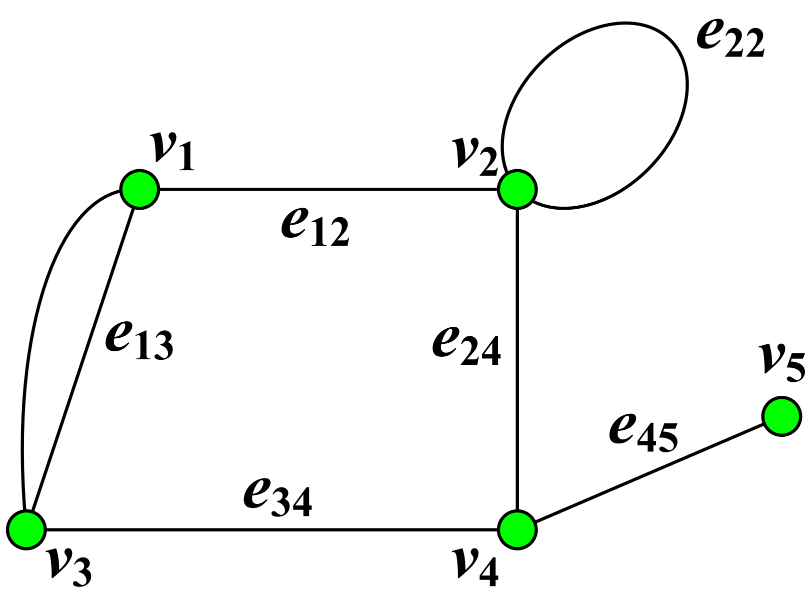
1.1 图与网络的基本概念

图与网络的定义

- 顶点/节点 (*Vertex*)
  - 物理实体、事物、概念
  - 一般用 v_i 表示
- 边 (*Edge*)
  - 节点间的连线，表示有关系
  - 一般用 e_{ij} 表示
- 图 (*Graph*)
  - 节点和边的集合
  - 一般用 $G(V,E)$ 表示
  - 点集 $V=\{v_1,v_2,\dots,v_n\}$
  - 边集 $E=\{e_{ij}\}$
- 网络 (*Network*)

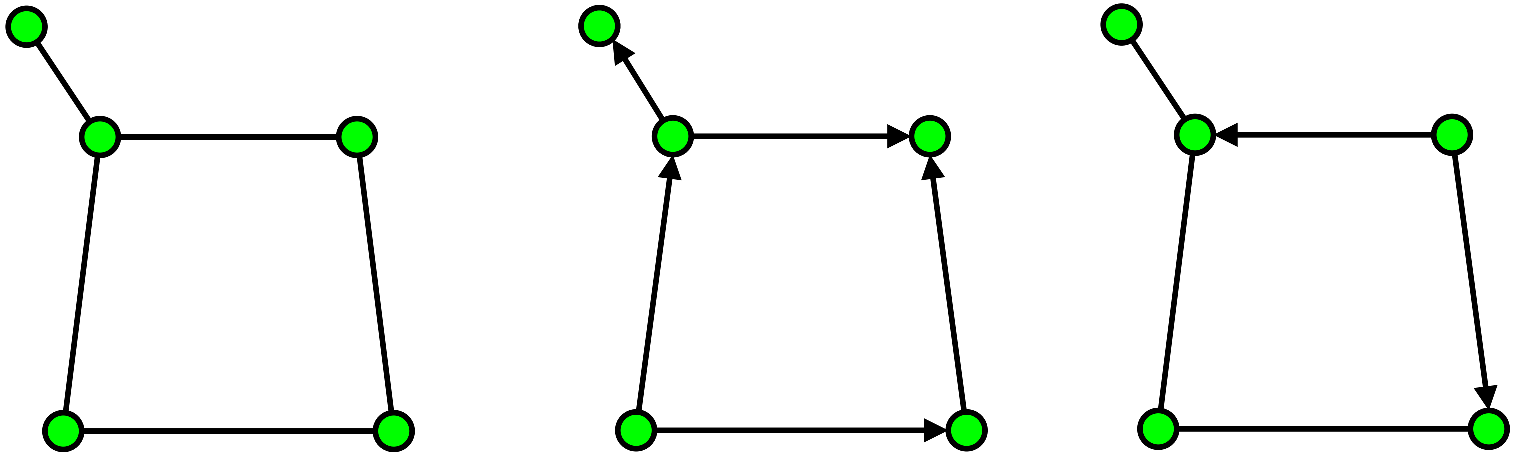
  边或点上带有某种数量指标，称之为“权”，如 w_{ij}

  又称加权图 (*Weighted graph*)



无向图，有向图

- 边都没有方向的图称为无向图，即 $e_{ij} = e_{ji}$ 或 $(v_i, v_j) = (v_j, v_i)$。
- 当边都有方向时，称为有向图，即在有序对(v_i, v_j)中，v_i表示始点，v_j表示终点。
- 在有向图中，有向边又称为弧，用 a_{ij} 表示，i, j 的顺序是不能颠倒的，图中弧的方向用箭头标识。
- 图中既有边又有弧，称为混合图。

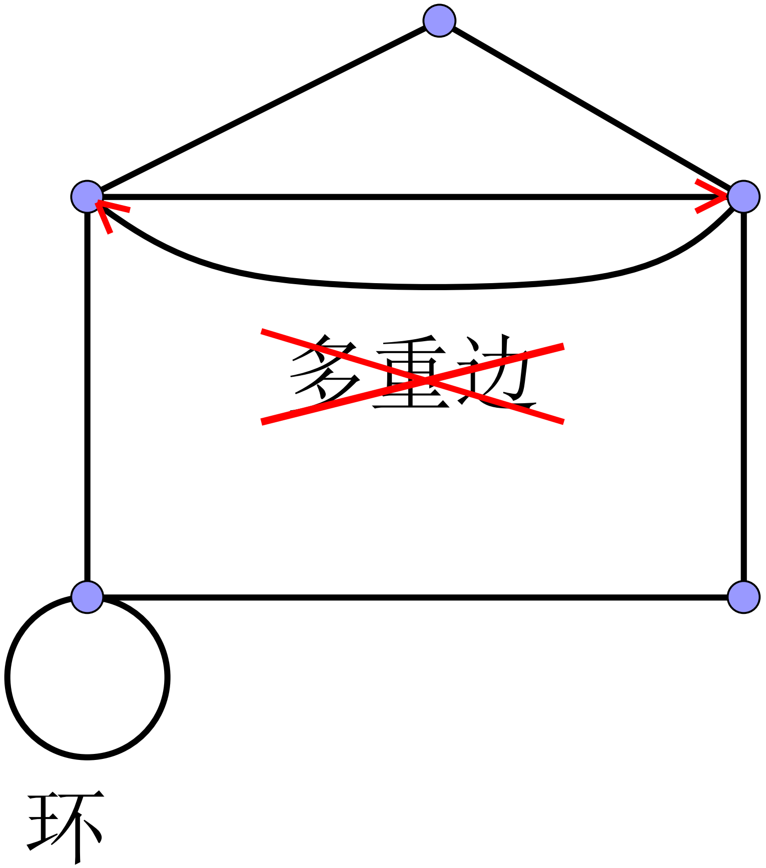


1.1 图与网络的基本概念

简单图，多重图

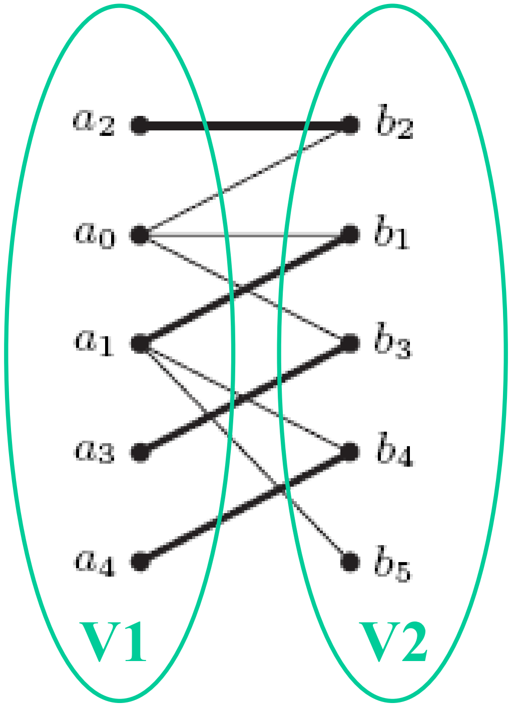
- 一个边的两个端点如果相同，称为环。
- 两个点之间如果多于一条边，称为多重边。
- 不含环和多重边的图称为简单图；含有多重边的图称为多重图。

注：有向图中两点之间有不同方向的两条边，不是多重边。



完全图，偶图

- 一个简单图中，若任意两点之间均有边相连，则称为完全图。
- n个顶点的完全图，边数有 $\frac{n(n-1)}{2}$ 条。
- 若图的顶点能分成两个互不相交的非空集合 V1 和 V2，使在同一集合中的任意两个顶点均不相邻，称为偶图（二分图，bipartition）。



顶点的次（度，vertex degree）

- 无向图中，与节点 v 关联的边数，称为该节点的次或度，记为 $\deg(v)$ 或 $d(v)$。
- 对任意节点 v，若度数 $d(v)$ 为奇数，则称此节点为奇点，若度数 $d(v)$ 为偶数，则称此节点为偶点。
- 次为 1 的点称为悬挂点，所关联的边称为悬挂边。
- 次为 0 的点称为孤立点。
- 有向图中，由节点指向外的弧的数目称为正次数，记为 $d+$，指向该节点的弧的数目称为负次数，记为 $d-$。

顶点的次（度，vertex degree）

定理 1： 图$G=(V,E)$中，所有点的次之和是边数的 2 倍

$$\sum_{v\in V} d(v) = 2m$$

其中$m=|E|$为边数。

定理 2： 任何图中，奇点的个数必为偶数。

$$\sum_{v\in V_1} d(v) + \sum_{v\in V_2} d(v) = \sum_{v\in V} d(v) = 2m$$

定理 3： 有向图中，所有顶点的入次之和等于所有顶点的出次之和。

南京大學 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

链，圈，道路，回路

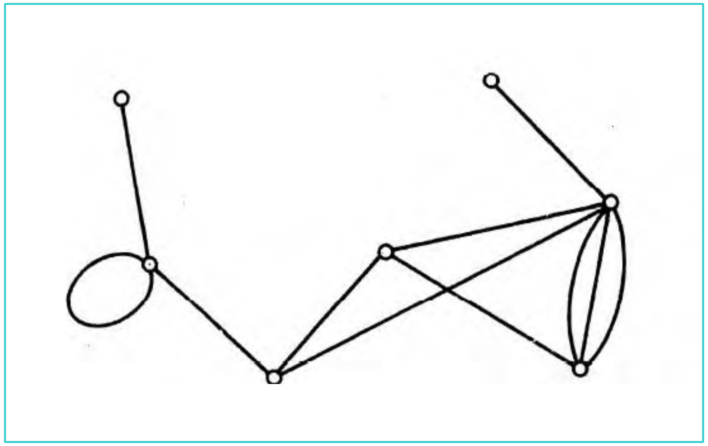
- 设 $G=(V,E)$ 是一个无向图，考虑图 G 中**边的序列**：$[v_0,v_1]$, $[v_1,v_2]$, $[v_2,v_3]$, …, $[v_{k-1},v_k]$，简单地记为 $(v_0,v_1,v_2,v_3,\dots,v_{k-1},v_k)$，在这个序列中，任意一条边的终点都是下一条边的始点，称此序列为图的一条链 link
- 若一条链中 $v_0=v_k$，则称为圈 loop

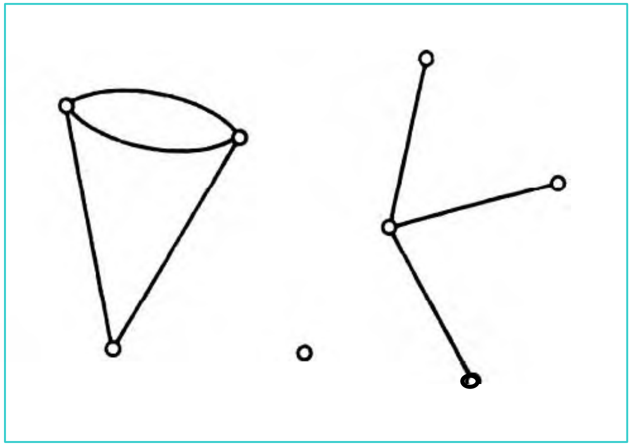
对有向图, 可以类似地定义链和圈

- 对有向图 $G=(V,E)$，当链（圈）上边的**方向相同**时，称为道路 path（回路 circuit）

连通图

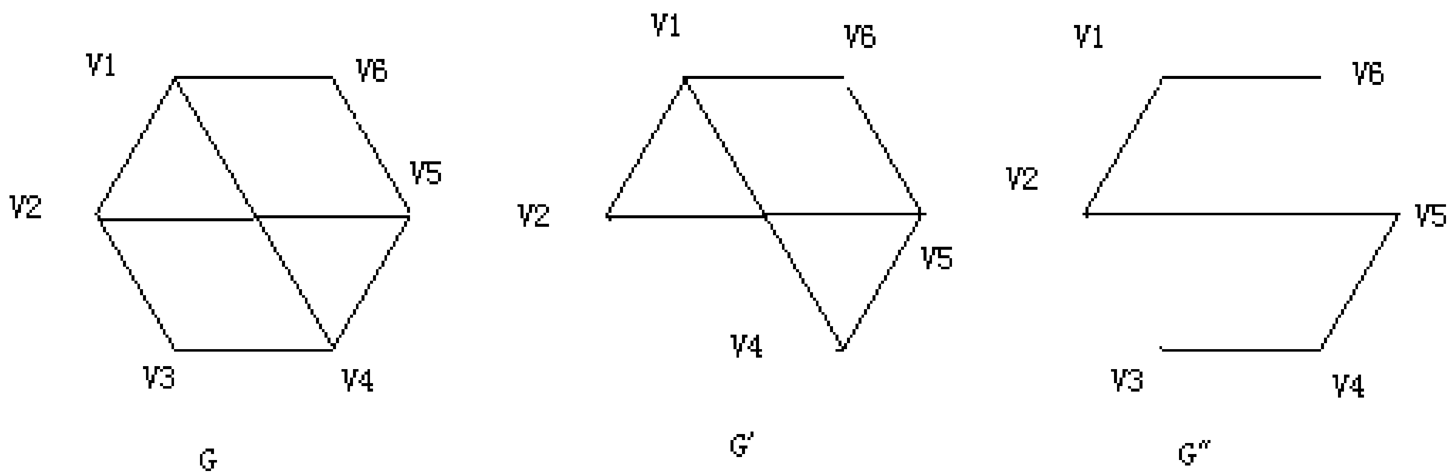
- 连通图（Connected graph）：在一个图中，每一对顶点之间至少存在一条链
- 否则为不连通图





子图

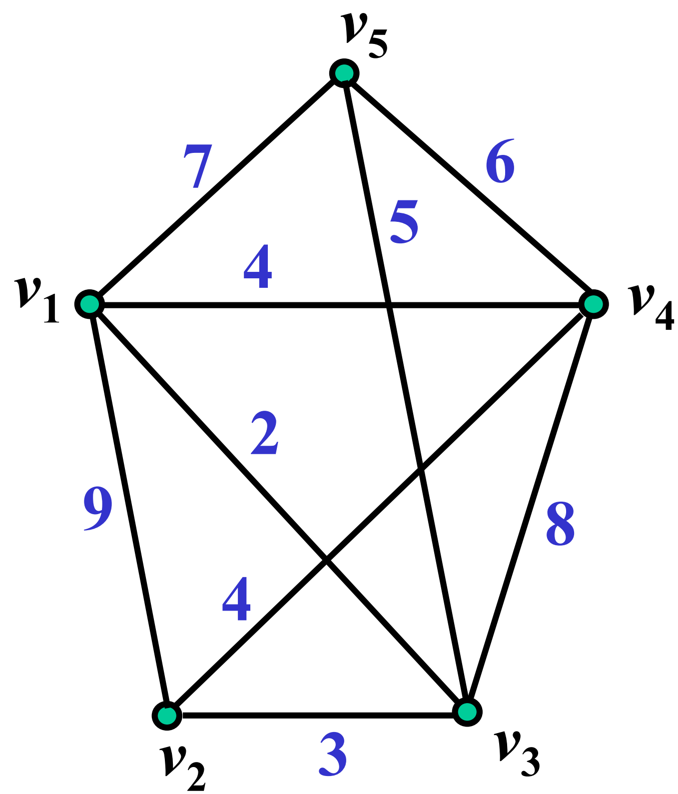
- 设有图 $G=(V,E)$ 和图 $G'=(V',E')$，若 G 和 G' 满足：
  - 若 $V'\subseteq V$，$E'\subseteq E$，则称 G' 是 G 的子图 Subgraph，记为 $G'\subseteq G$；
  - 特别, 若 $V'=V$，$E'\subseteq E$，则称 G' 是 G 的生成子图（支撑图）Spanning Subgraph。



权矩阵 weighted matrix

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

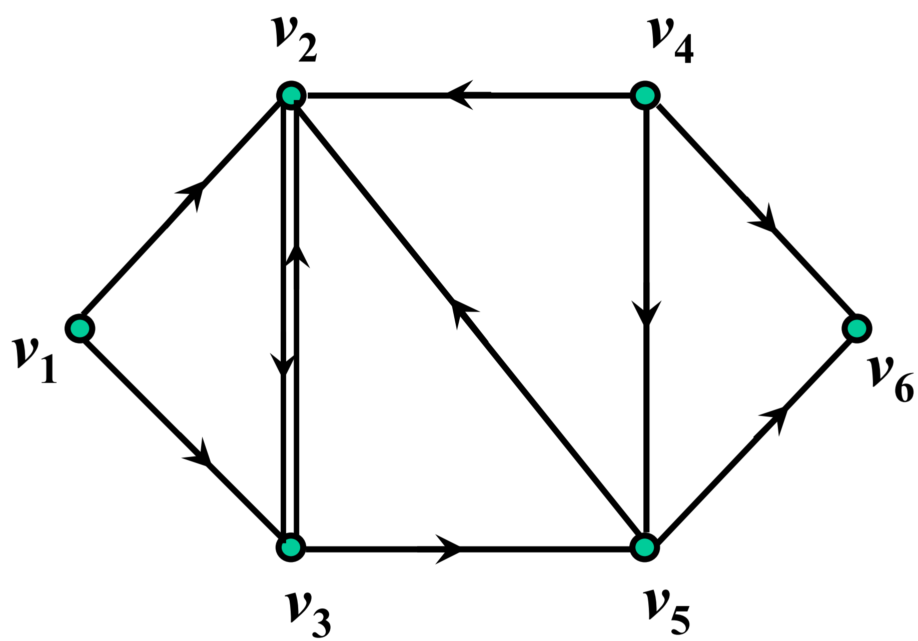
$$a_{ij} = \begin{cases} w_{ij} & (v_i, v_j) \in E \\ 0 & o.w. \end{cases}$$



1.2 图的矩阵表示

邻接矩阵 adjacent matrix

$$A = \begin{array}{cc} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} & \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} \\ \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} & \end{array}$$



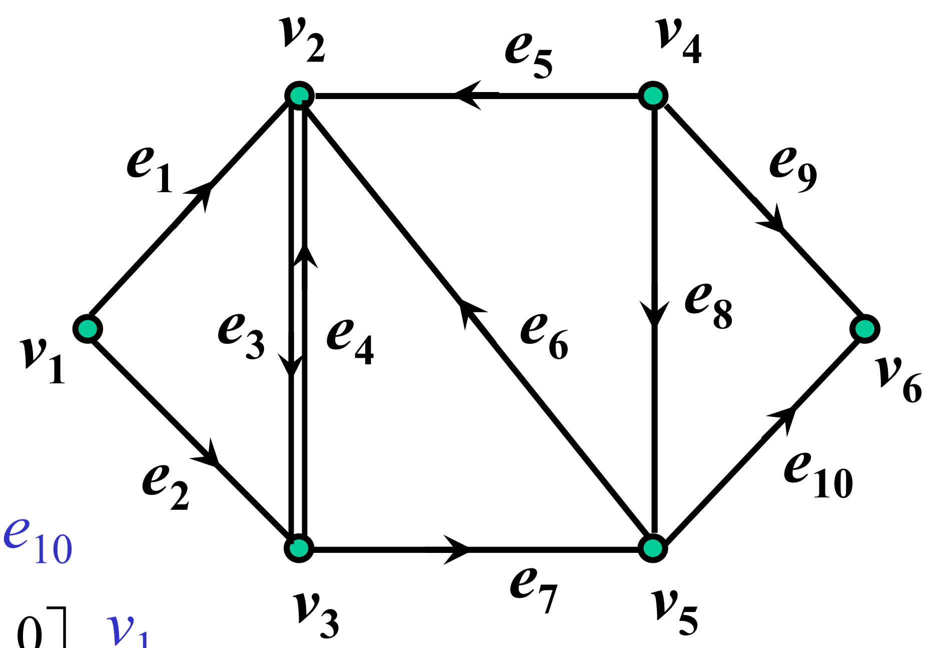
$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & o.w. \end{cases}$$

若G为无向图，则邻接矩阵为对称矩阵。

1.2 图的矩阵表示

关联矩阵 incidence matrix

- A 每一列元素和等于2
- A 每一行元素和等于对应顶点的次数



$$
A = \begin{array}{c}
\begin{array}{cccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array}
$$

$$
a_{ij} = \begin{cases} 1 & v_i \in e_j \\ 0 & o.w. \end{cases}
$$

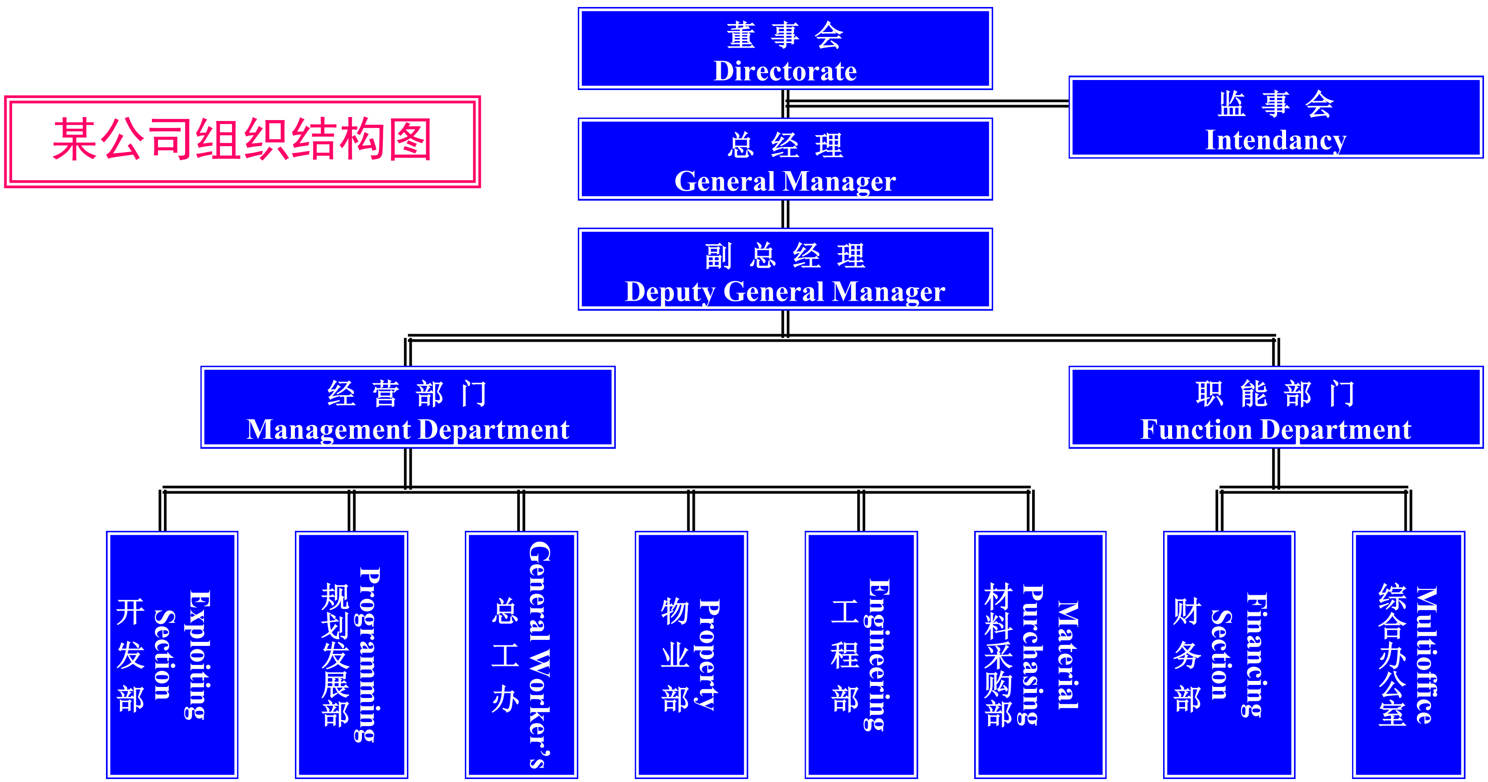
南京大學 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

第七章 图与网络

2. 树

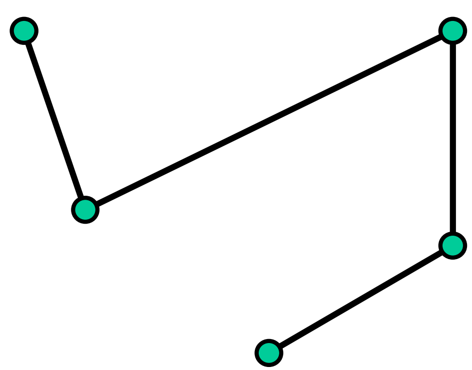
2.1 树的概念

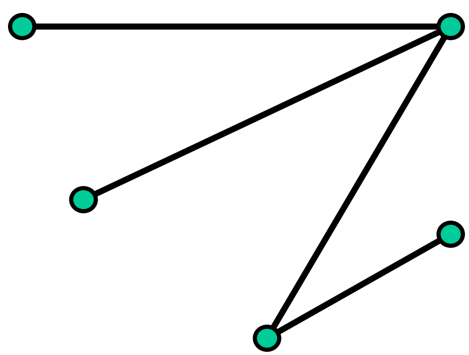
- 树的定义：连通且不含圈的无向图。
- 多级辐射制的电信网络、管理的指标体系、家谱、分类学、组织结构等都是典型的树图。



2.2 树的性质

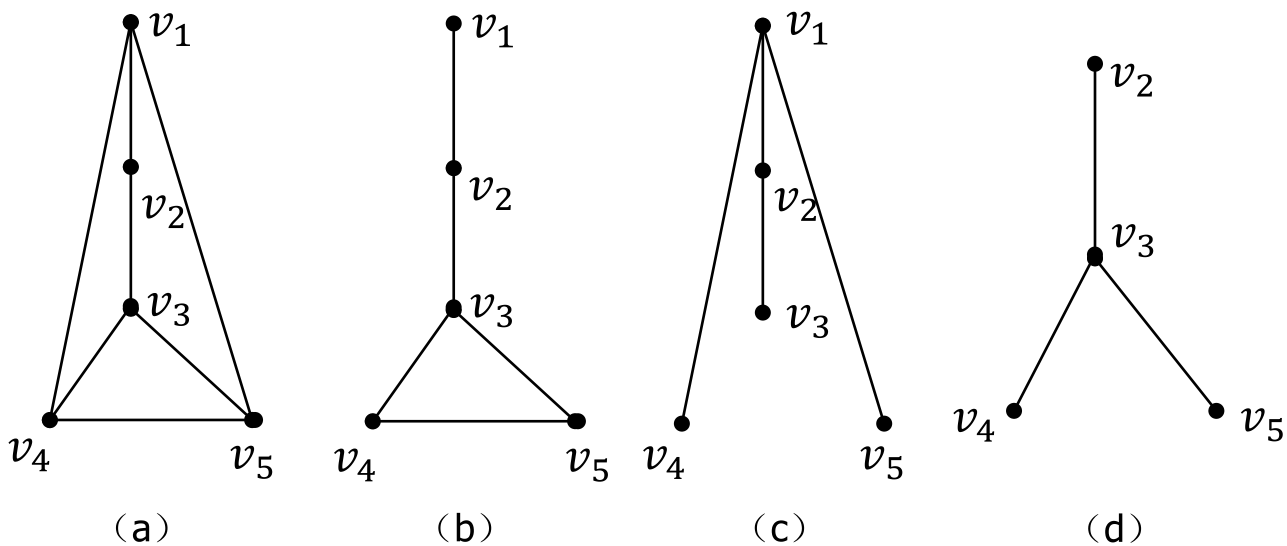
- 连通、无圈；
- 设顶点数为$|V| = n$，边数为$|E| = m$，那么$m = n - 1$；
- 任意两点之间有且仅有一条链；
- 去掉任意一条边，就不连通；
- 增加任意一条边，就得到一个且仅一个圈。





2.3 图的生成树

设 $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ 是一个连通的无向图，若 $\boldsymbol{G}$ 的某个生成子图是一棵树，则称该树为 $\boldsymbol{G}$ 的生成树（支撑树）Spanning Tree，记为 $\boldsymbol{T_G}$。



- 图(c)是图(a)的生成树，而图(d)所示的树则不是图(a)的生成树。
- 图(b)是(a)的生成子图，但不是生成树。

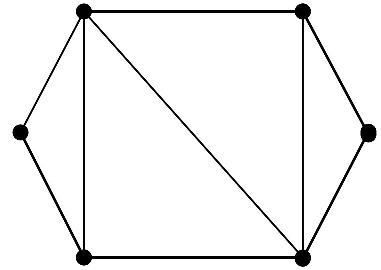
2.3 图的生成树

定理：图 $\boldsymbol{G}$ 有生成树（支撑树）的充要条件是图 $\boldsymbol{G}$ 是连通的。

- 如何找到一棵生成树？
  - 破圈法：每次去掉圈中的一条边，其去掉的边的总数为 $m-(n-1)$；
  - 避圈法：每次选取 G 中的一条边，该边不能与已经选取的边构成回路，此时，选取的边的总数为 $(n-1)$ 。

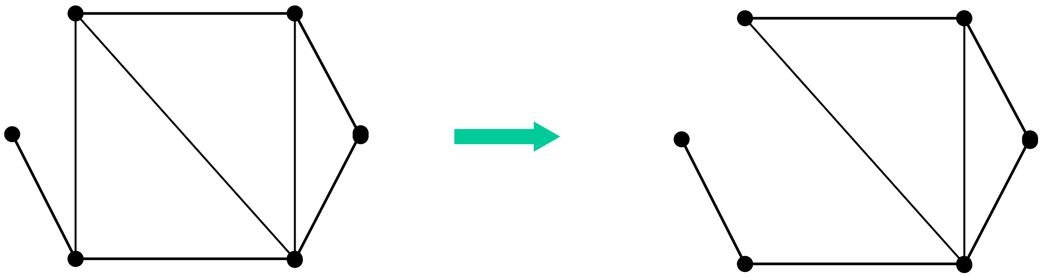
2.3 图的生成树

求右图(a)中的生成子树。

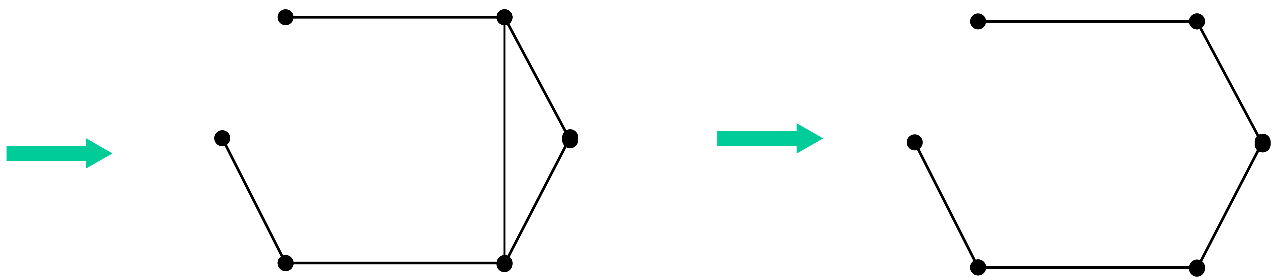


(a)

1. 破圈法:



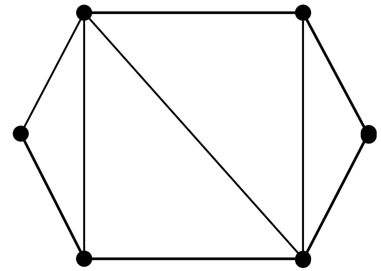
(b) (c)



(d) (e)

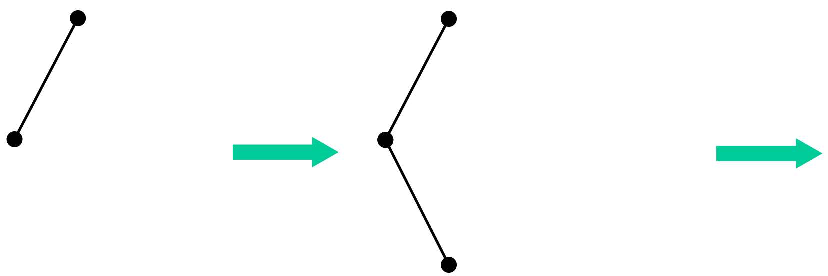
2.3 图的生成树

求右图(a)中的生成子树。

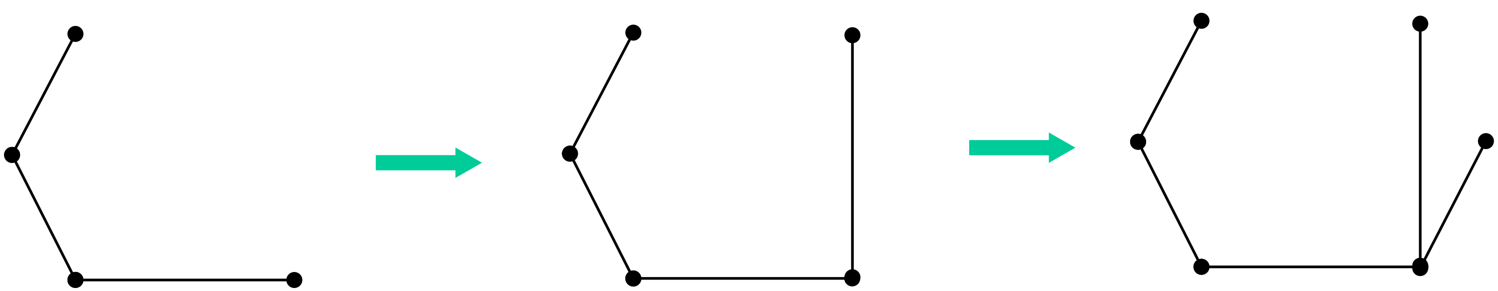


(a)

2. 避圈法:

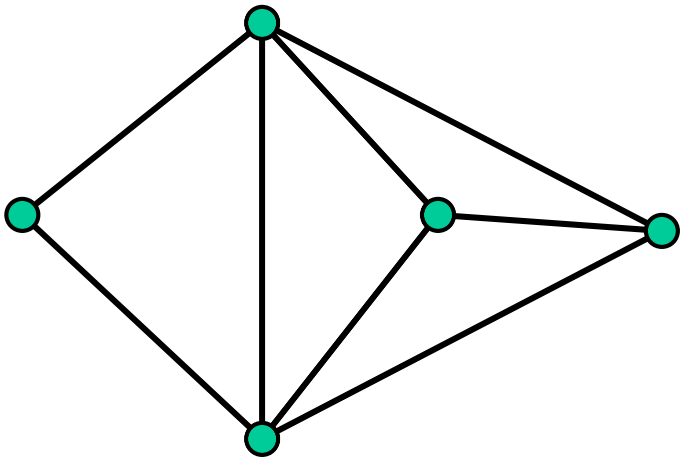


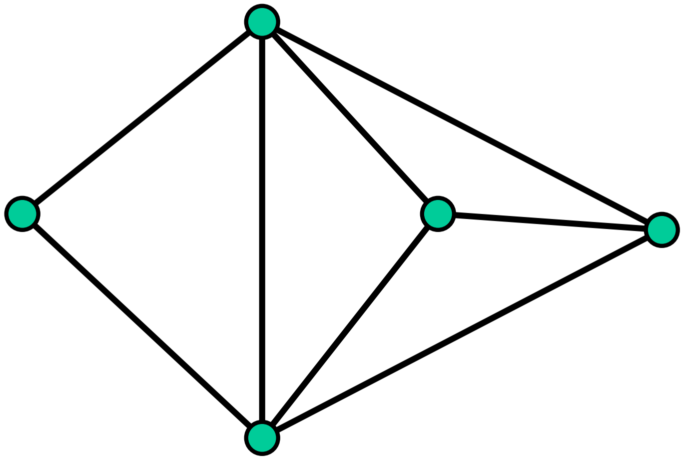
(f) (g)

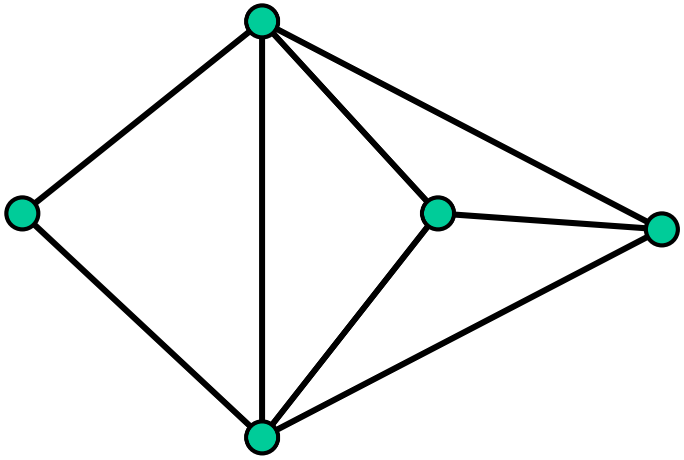


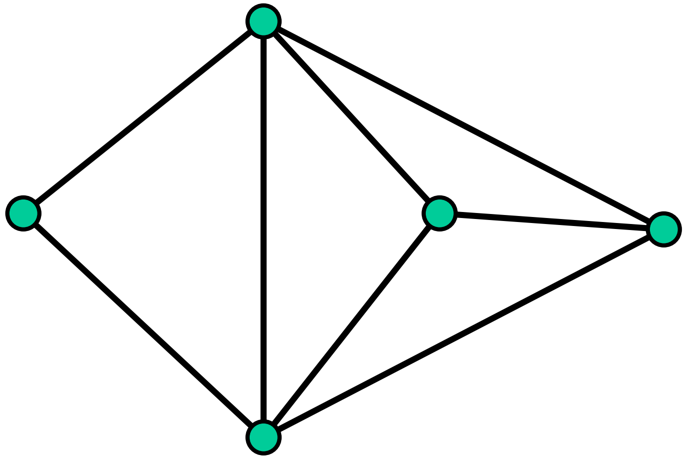
(h) (i) (j)

破圈法



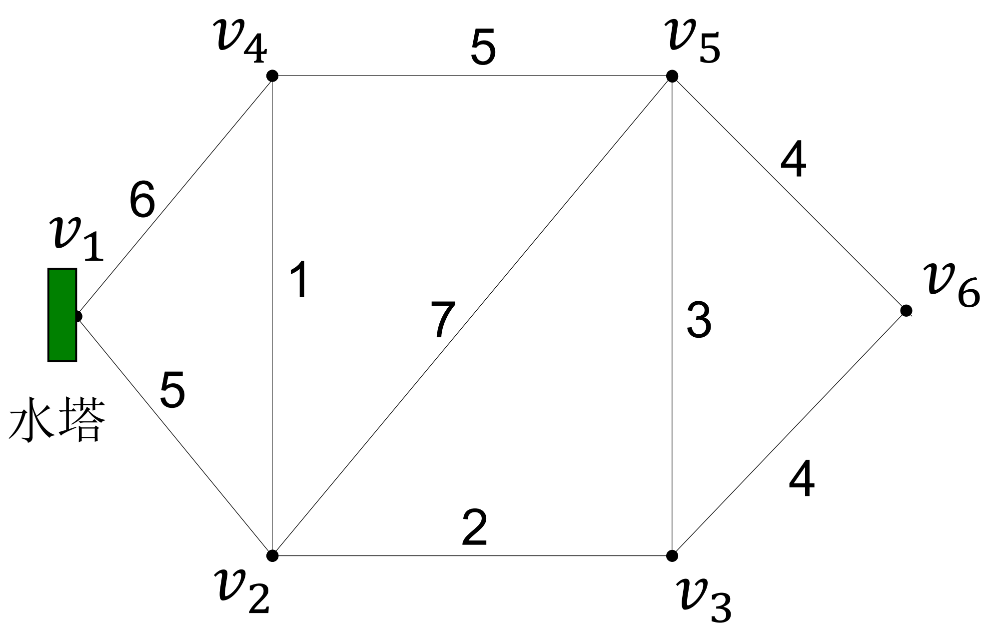






在架设电话线，铺设自来水或暖气管道的工程设计中会遇到如下的优化问题：如何相互连通，使得总线路长度最短？

例：某居民区五栋楼的分布如图。每条边旁的数字表示水塔与各楼间的距离。试求最短的管道铺设方案。



↓

最小生成树问题

2.4 图的最小生成树—数学定义

- 设有一个连通图 $G=(V,E)$，每一边 $e_{ij}=[v_i,v_j]$ 有一个非负权

$$w(e_{ij})=w_{ij}\ (w_{ij}\geq 0)$$

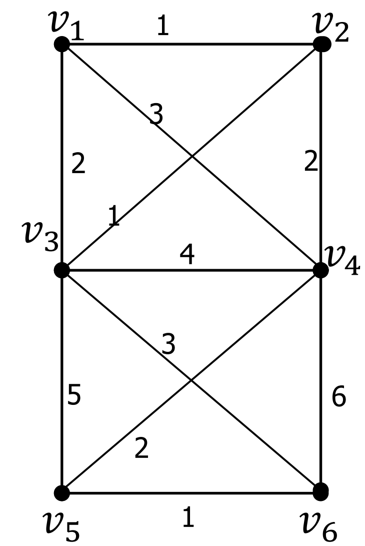
- 如果 $T=(V,E')$，是 G 的一个支撑树，称 E' 中所有边的权之和为支撑树 T 的权，记为 $w(T)$

$$w(T)=\sum_{[v_i,v_j]\in T} w_{ij}$$

- 如果支撑树 T^* 的权 $w(T^*)$ 是 G 的所有支撑树的权中最小者，则称 T^* 是 G 的最小支撑树（最小生成树），即

$$w(T^*)=\min_{T} w(T)$$

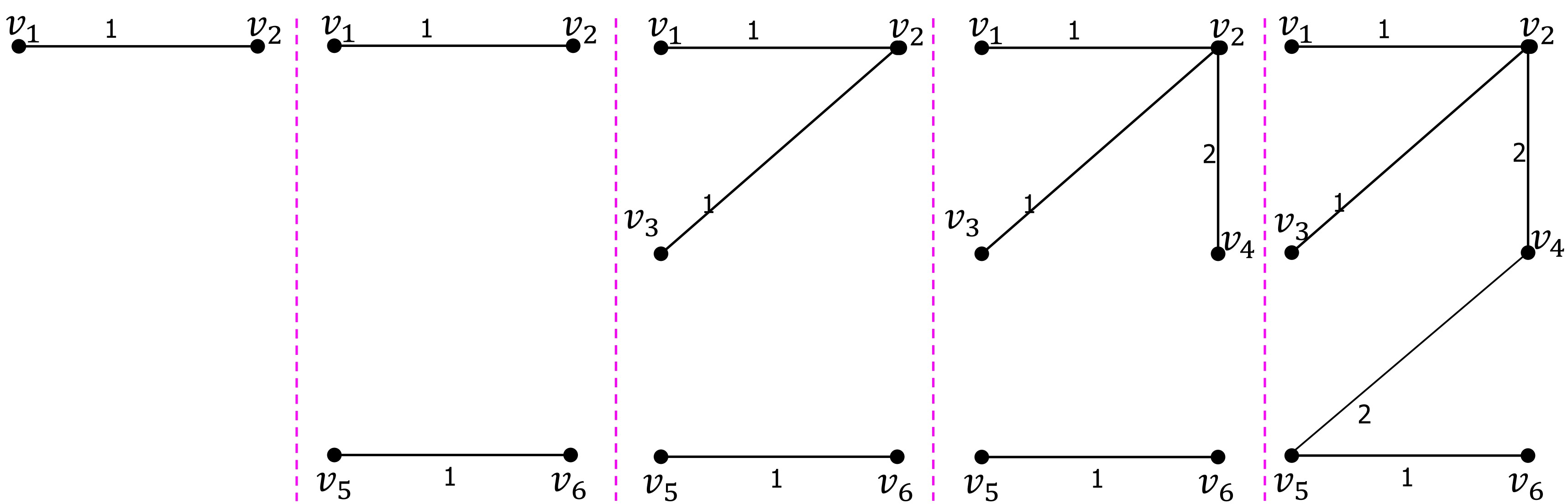
2.4 图的最小生成树



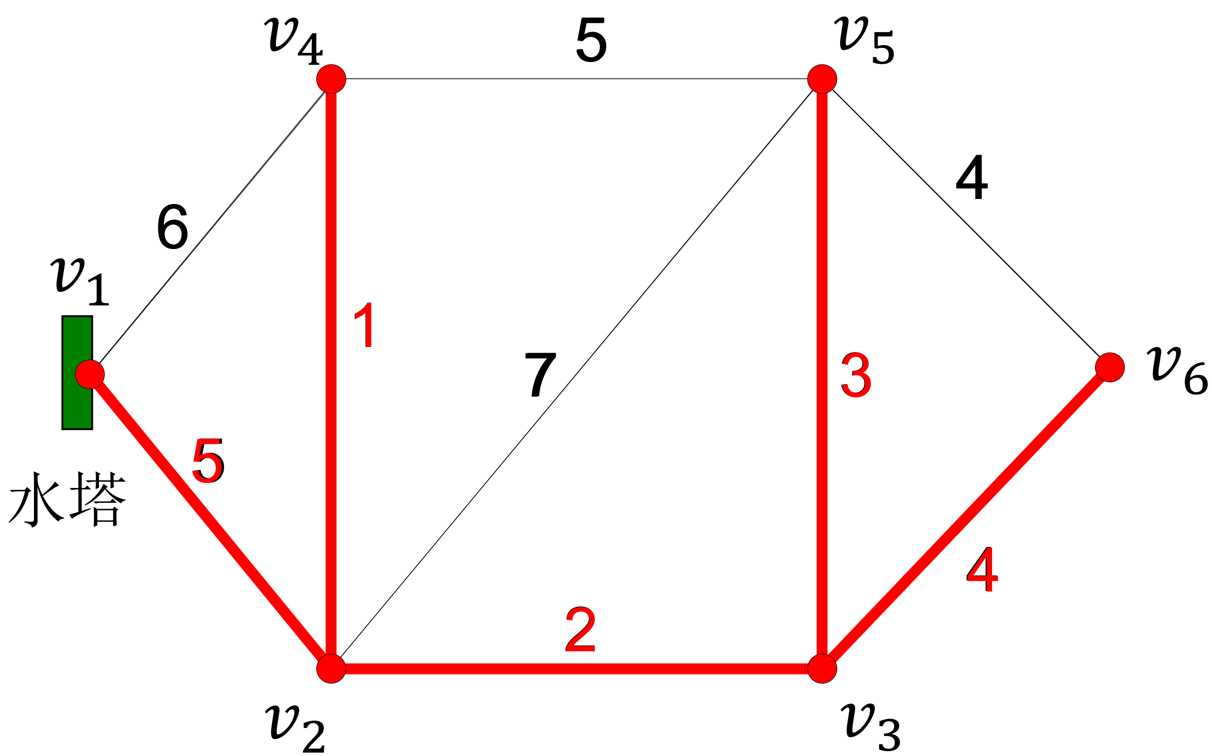
求图 G 的最小生成树的方法有二：

- 避圈法 Kruskal
- 破圈法

避圈法：



练习：试利用破圈法找到下图的最小生成树。



南京大学 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

矩阵计算方法：

Prim 算法

$$
\begin{array}{cc}
 & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{array}{cc} \mathrm{T} & v_1 \\ & v_2 \\ & v_3 \\ & v_4 \\ & v_5 \\ & v_6 \end{array} &
\begin{bmatrix}
0 & 5 & \infty & 6 & \infty & \infty \\
5 & 0 & 2 & 1 & 7 & \infty \\
\infty & 2 & 0 & \infty & 3 & 4 \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
\end{array}
$$

矩阵计算方法：

$$
\begin{array}{cc}
 & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{array}{cc} \mathrm{T} & v_1 \\ \mathrm{T} & v_2 \\ & v_3 \\ & v_4 \\ & v_5 \\ & v_6 \end{array} &
\begin{bmatrix}
0 & \boxed{5} & \infty & 6 & \infty & \infty \\
5 & 0 & 2 & 1 & 7 & \infty \\
\infty & 2 & 0 & \infty & 3 & 4 \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
\end{array}
$$

矩阵计算方法：

$$
\begin{array}{cc|c}
 & & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\mathrm{T} & v_1 & \\
\mathrm{T} & v_2 & \\
 & v_3 & \\
\mathrm{T} & v_4 & \\
 & v_5 & \\
 & v_6 &
\end{array}
\begin{bmatrix}
0 & \boxed{5} & \infty & 6 & \infty & \infty \\
5 & 0 & 2 & \boxed{1} & 7 & \infty \\
\infty & 2 & 0 & \infty & 3 & 4 \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
$$

2.4 图的最小生成树

矩阵计算方法：

$$\begin{array}{cc}
 & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{array}{cc} T & v_1 \\ T & v_2 \\ T & v_3 \\ T & v_4 \\ & v_5 \\ & v_6 \end{array} &
\begin{bmatrix}
0 & \boxed{5} & \infty & 6 & \infty & \infty \\
5 & 0 & \boxed{2} & \boxed{1} & 7 & \infty \\
\infty & 2 & 0 & \infty & 3 & 4 \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
\end{array}$$

矩阵计算方法：

$$
\begin{array}{cc}
 & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{array}{cc} \mathrm{T} & v_1 \\ \mathrm{T} & v_2 \\ \mathrm{T} & v_3 \\ \mathrm{T} & v_4 \\ \mathrm{T} & v_5 \\ & v_6 \end{array} &
\begin{bmatrix}
0 & \boxed{5} & \infty & 6 & \infty & \infty \\
5 & 0 & \boxed{2} & \boxed{1} & 7 & \infty \\
\infty & 2 & 0 & \infty & \boxed{3} & 4 \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
\end{array}
$$

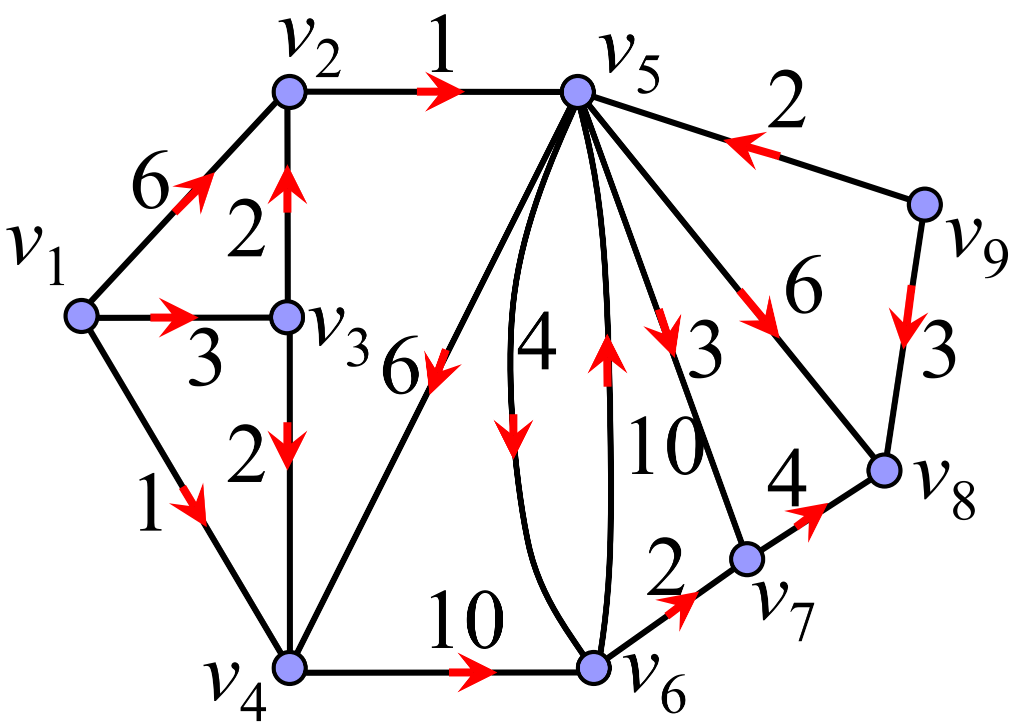
矩阵计算方法：

$$
\begin{array}{cc}
 & \\
\mathrm{T} & v_1 \\
\mathrm{T} & v_2 \\
\mathrm{T} & v_3 \\
\mathrm{T} & v_4 \\
\mathrm{T} & v_5 \\
\mathrm{T} & v_6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{bmatrix}
0 & \boxed{5} & \infty & 6 & \infty & \infty \\
5 & 0 & \boxed{2} & \boxed{1} & 7 & \infty \\
\infty & 2 & 0 & \infty & \boxed{3} & \boxed{4} \\
6 & 1 & \infty & 0 & 5 & \infty \\
\infty & 7 & 3 & 5 & 0 & 4 \\
\infty & \infty & 4 & \infty & 4 & 0
\end{bmatrix}
\end{array}
$$

第七章 图与网络

3. 最短路问题

- **例：**已知如图所示的单行线交通网，每条弧旁的数字表示这条单行线的距离。现在某人要从 v_1 出发，通过这个交通网到达 v_8，求使总距离最短的旅行线路。



3.1 引言

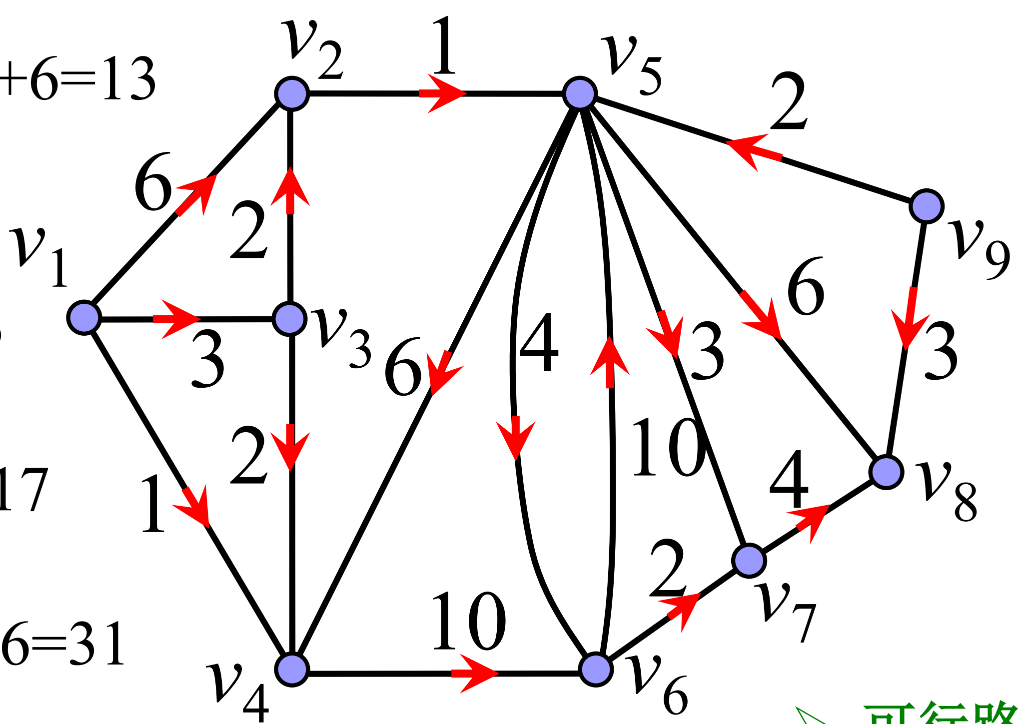
➢ **例：** 已知如图所示的单行线交通网，每条弧旁的数字表示这条单行线的距离。现在某人要从 v_1 出发，通过这个交通网到达 v_8，求使总距离最短的旅行线路。

6+1+6=13

3+2+1+3+4=13

1+10+2+4=17

3+2+10+10+6=31



➢ 可行路线有多条！

➢ 哪一条最短？

3.2 问题的一般提法

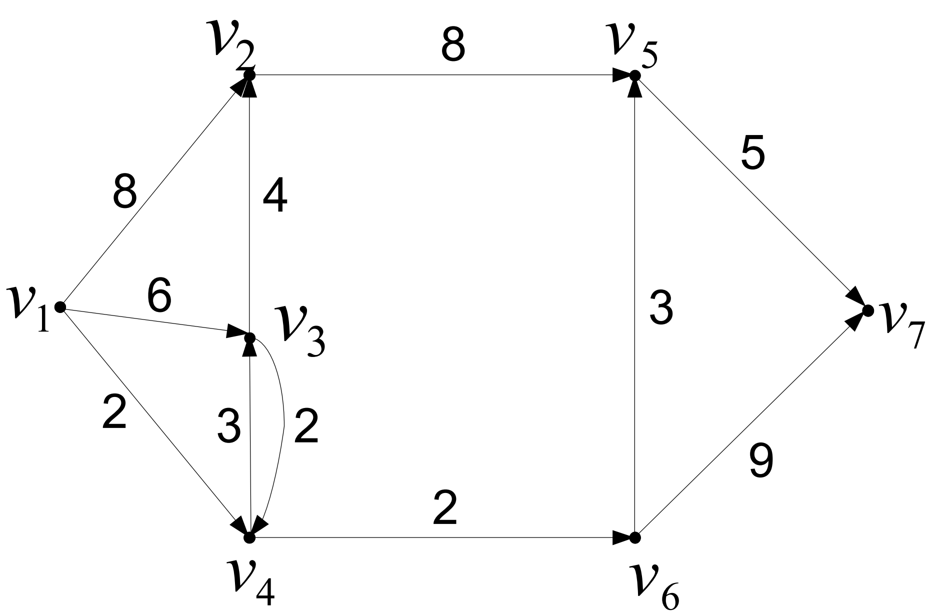
- 给定一个赋权有向图$D=(V,A)$，对每一弧$a=(v_i,v_j)$相应的有权 $w(a)=w_{ij}$。
- 给定 D 中的两个顶点v_s和v_t，设 P 是 D 中从v_s到v_t的一条路，定义路P的权是P中所有弧的权之和，即$w(P)$。
- 最短路问题就是要在所有从v_s到v_t的可行路中，求一条权最小的路。
- 记P_0是从v_s到v_t的最短路，路P_0的权为从v_s到v_t的距离，记为$d(v_s,v_t)$。

3.3 Dijkstra算法

—— 求无负权网络最短路的最好方法之一

基本原理：

若序列$\{v_s, v_1, \ldots, v_{n-1}, v_n\}$是从$v_s$到$v_n$的最短路，则序列$\{v_s, v_1, \ldots, v_{n-1}\}$必为从$v_s$到$v_{n-1}$的最短路



$v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_7$

如果这是v_1到达v_7的最短路，那么也是v_1到达该路径上任意点的最短路。

Dijkstra算法 ⟺ 逐点求最短路

3.3 Dijkstra算法

- 对每个节点，用两种标号：T 和 P (表示从始点到该节点的距离)
  - T 是目前路径的距离 (Tentative Label)
  - P 是最短距离 (Permanent Label)
- 开始时，令始点有P=0，记P标号，其它节点有T=M/∞
- 不断改进T值，当其最小时，将其改为P标号
- 当所有的标号都为P时，找到最短路

3.3 Dijkstra算法

- 标号过程分为两步：

Step 1. 修改T标号。

假定v_i是始点或为新产生的P标号点，考察以v_i为始点的所有弧段v_iv_j

✓ 如果v_j是P标号点，则对v_j不再进行标号;

✓ 如果v_j是T标号点，则进行如下的修改：

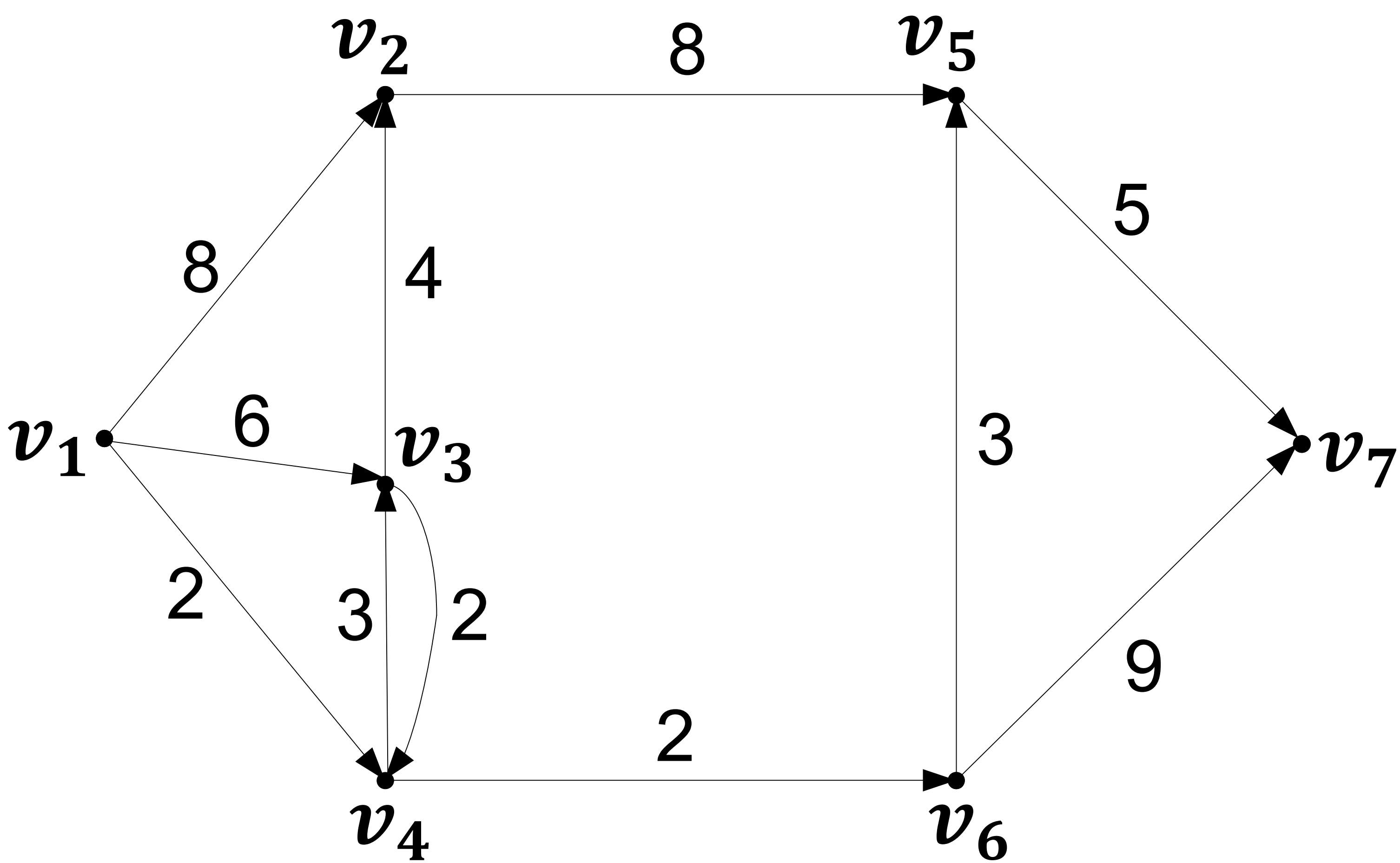
$$T(v_j) = \min\{T(v_j), P(v_i) + w_{ij}\}$$

Step 2. 产生新的P标号。

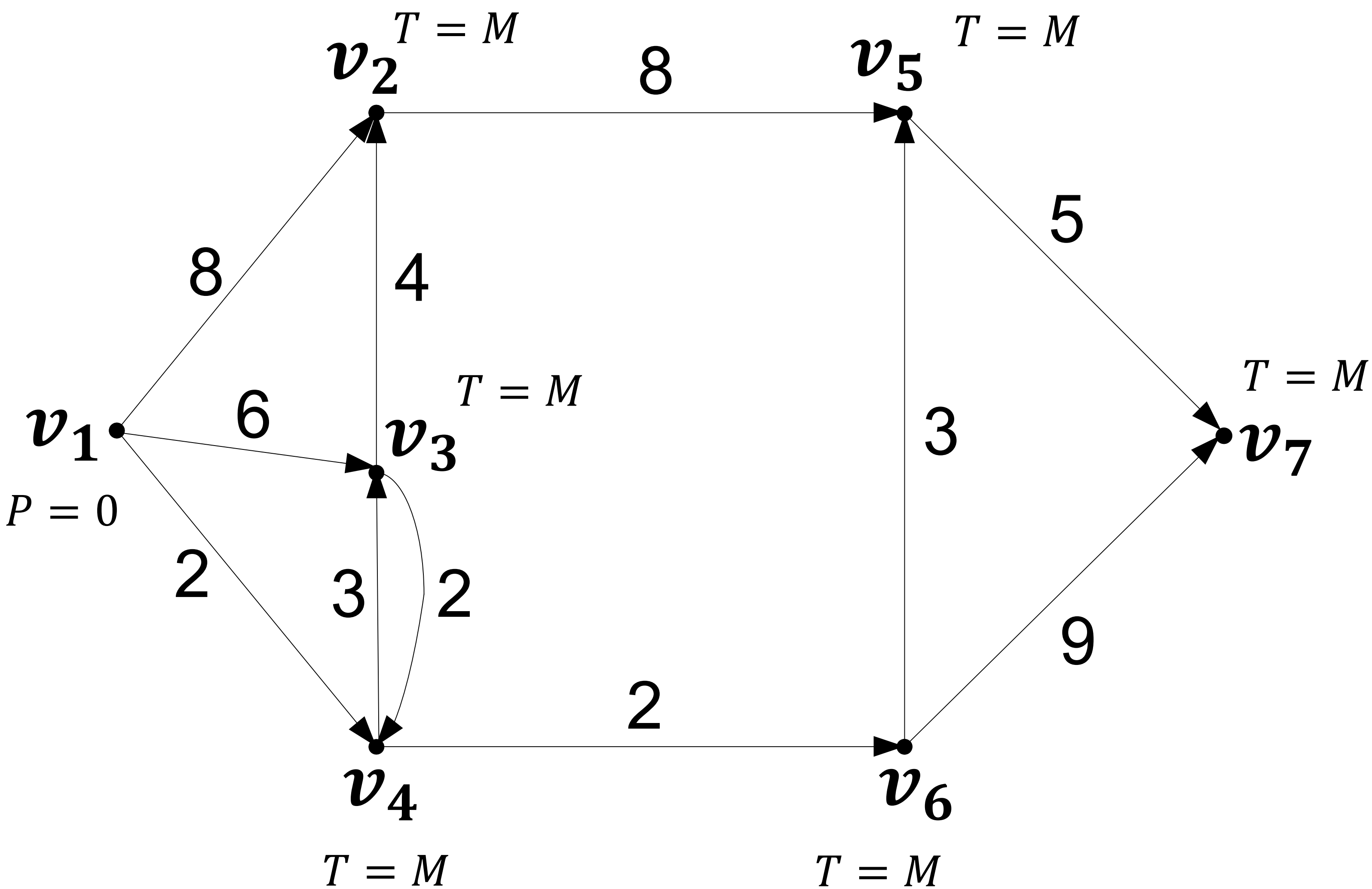
原则：在现有的T标号中将值最小者改为P标号。

3.3 Dijkstra算法—例题

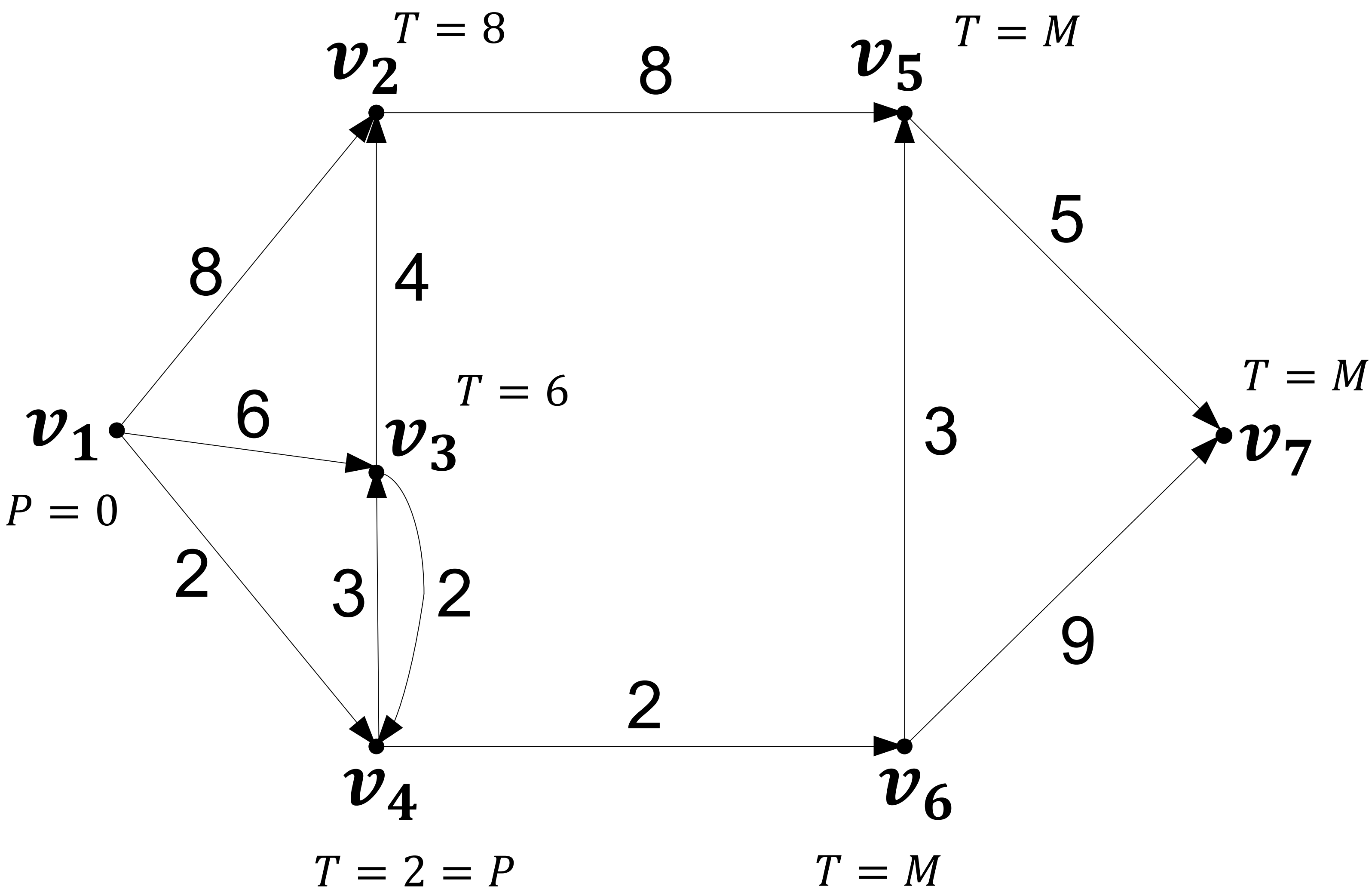
$v_1 \rightarrow v_7$ 的最短路径?



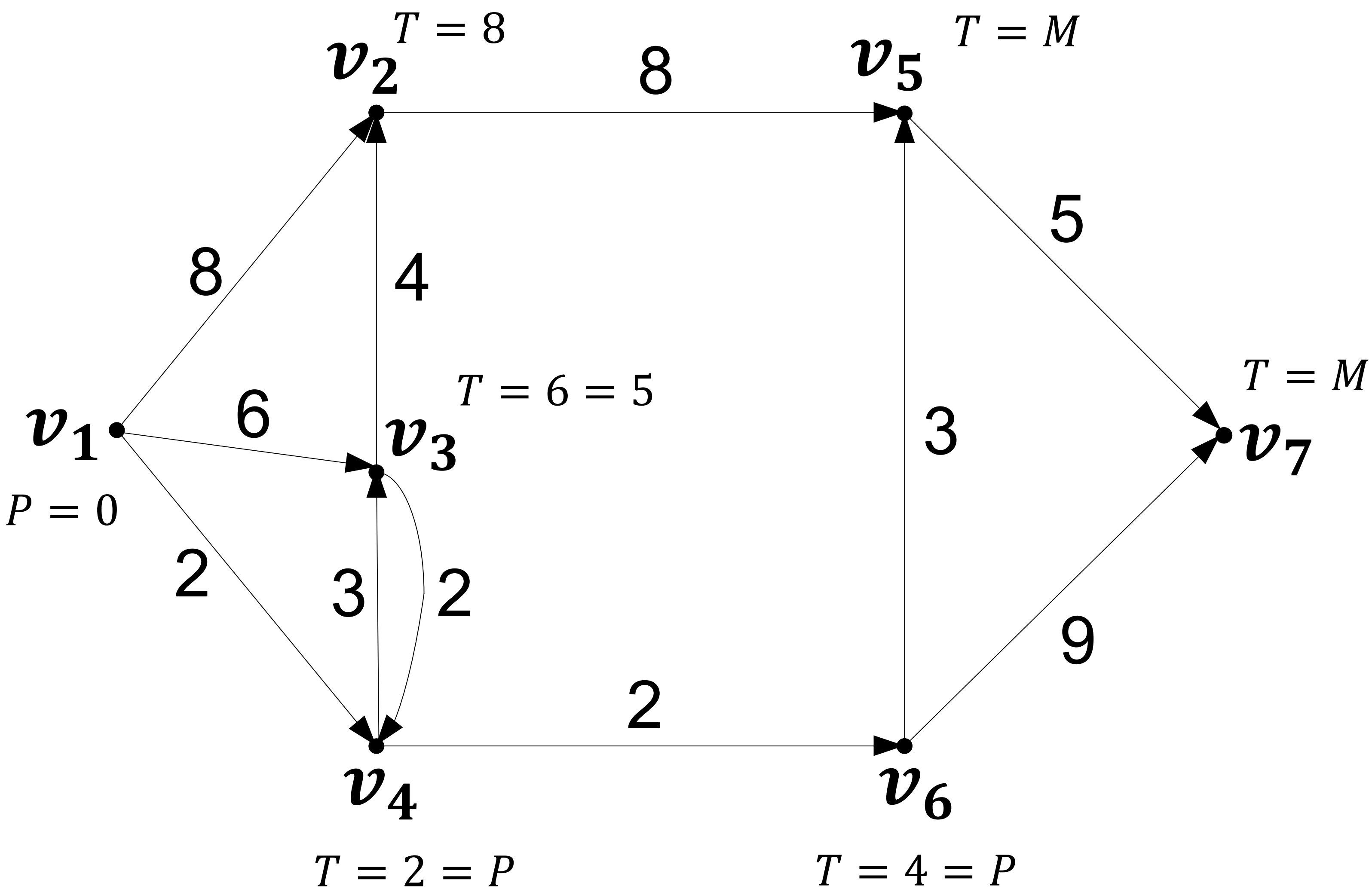
3.3 Dijkstra算法一例题



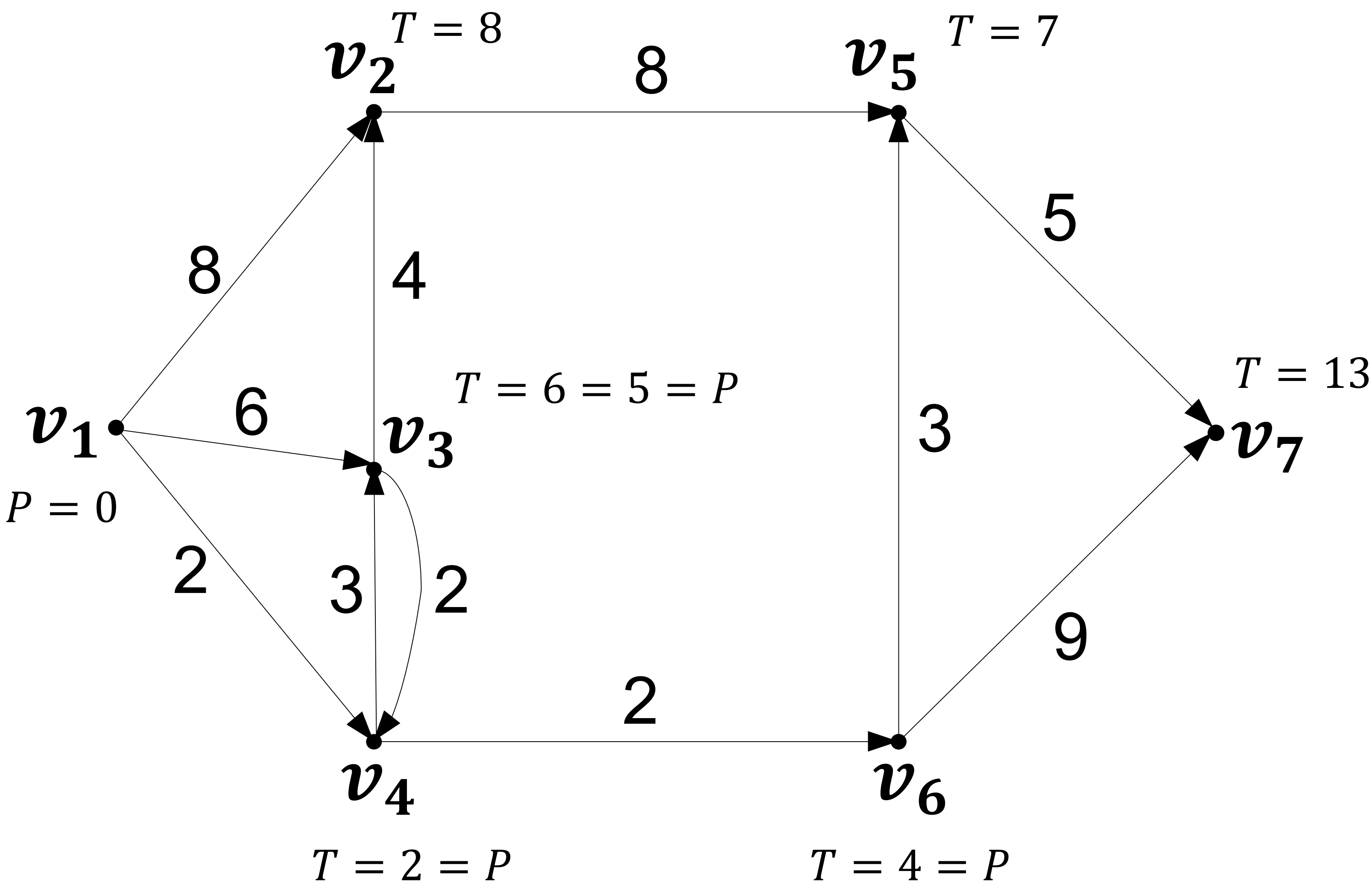
3.3 Dijkstra算法—例题



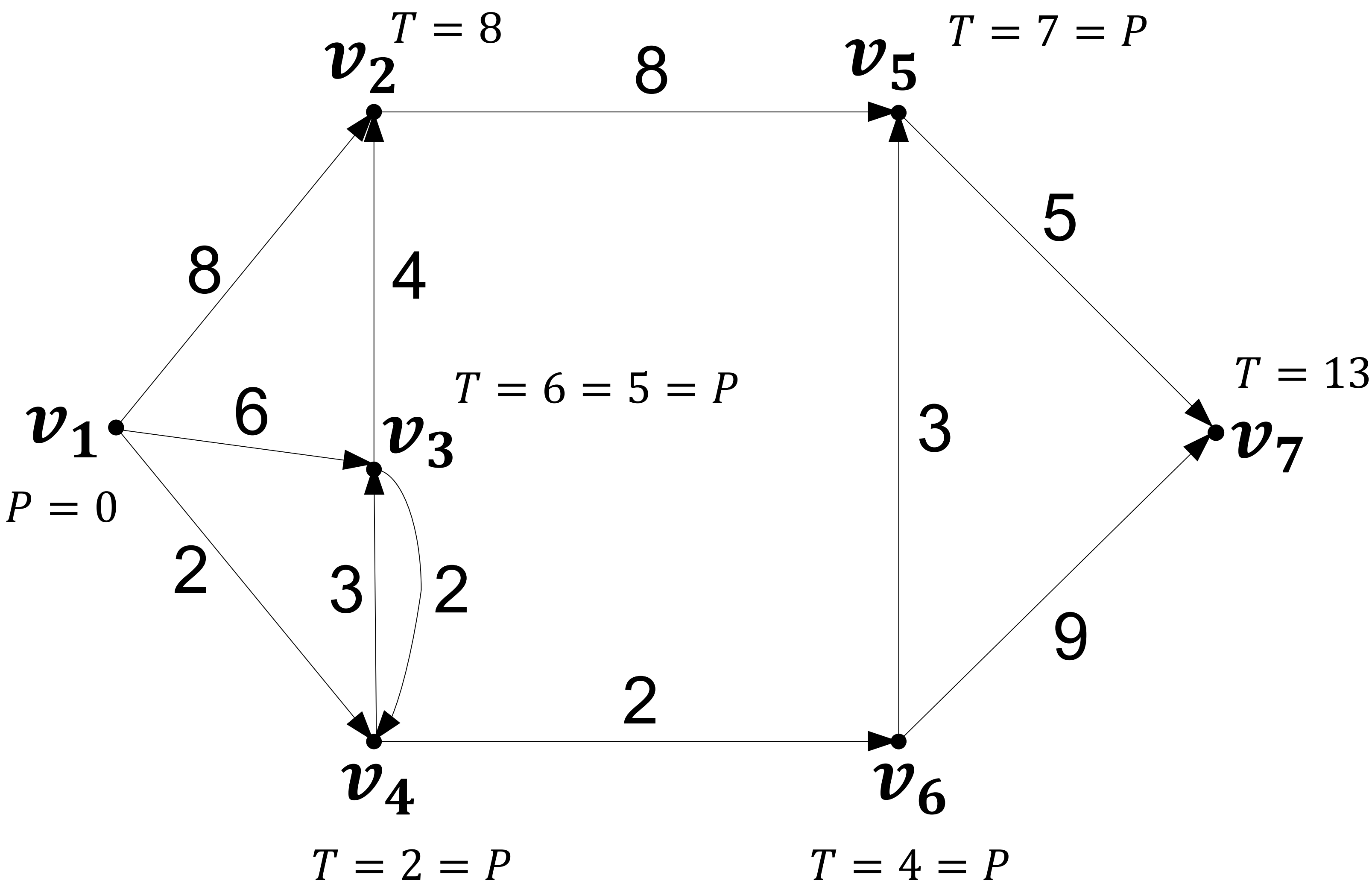
3.3 Dijkstra算法—例题



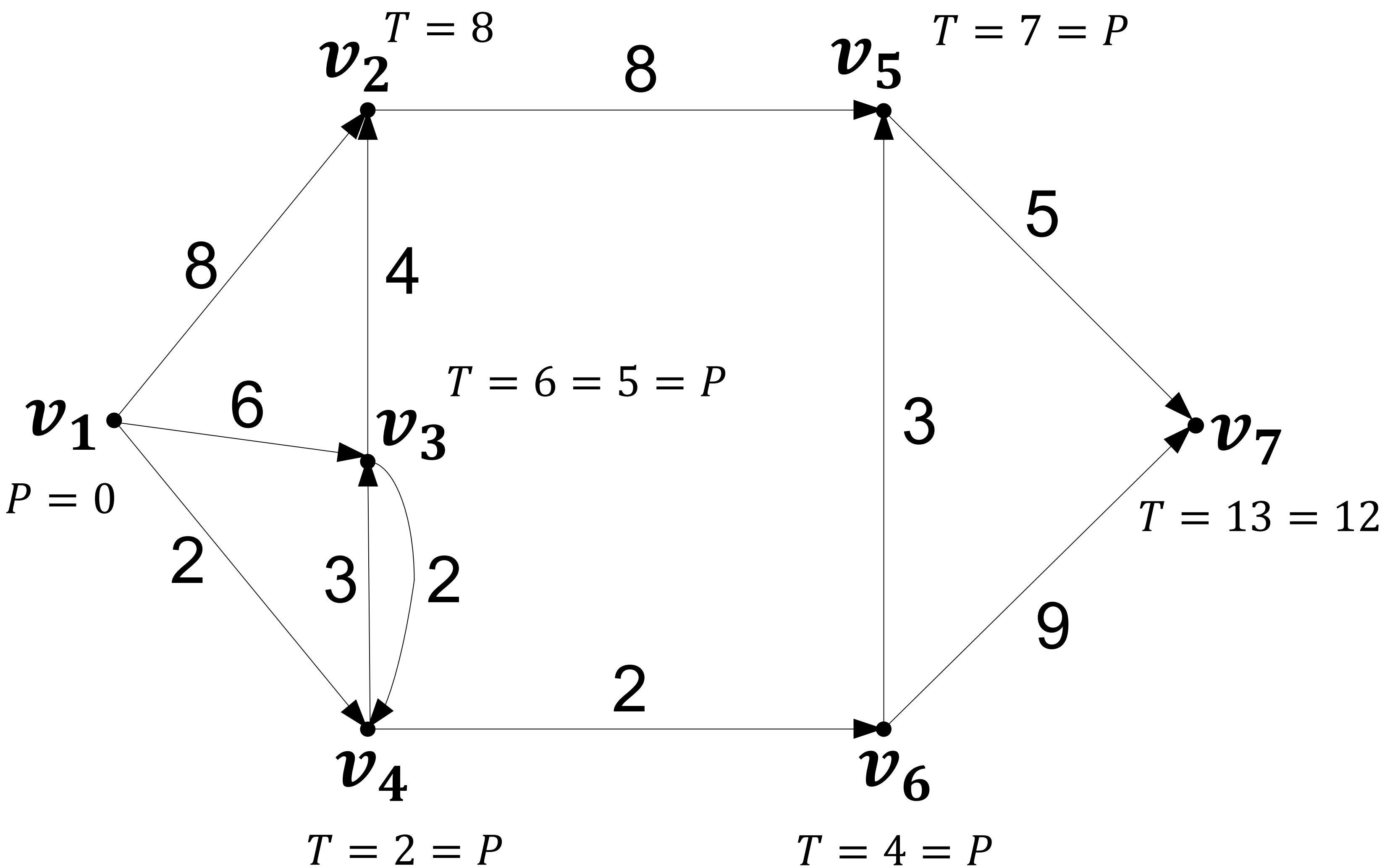
3.3 Dijkstra算法—例题



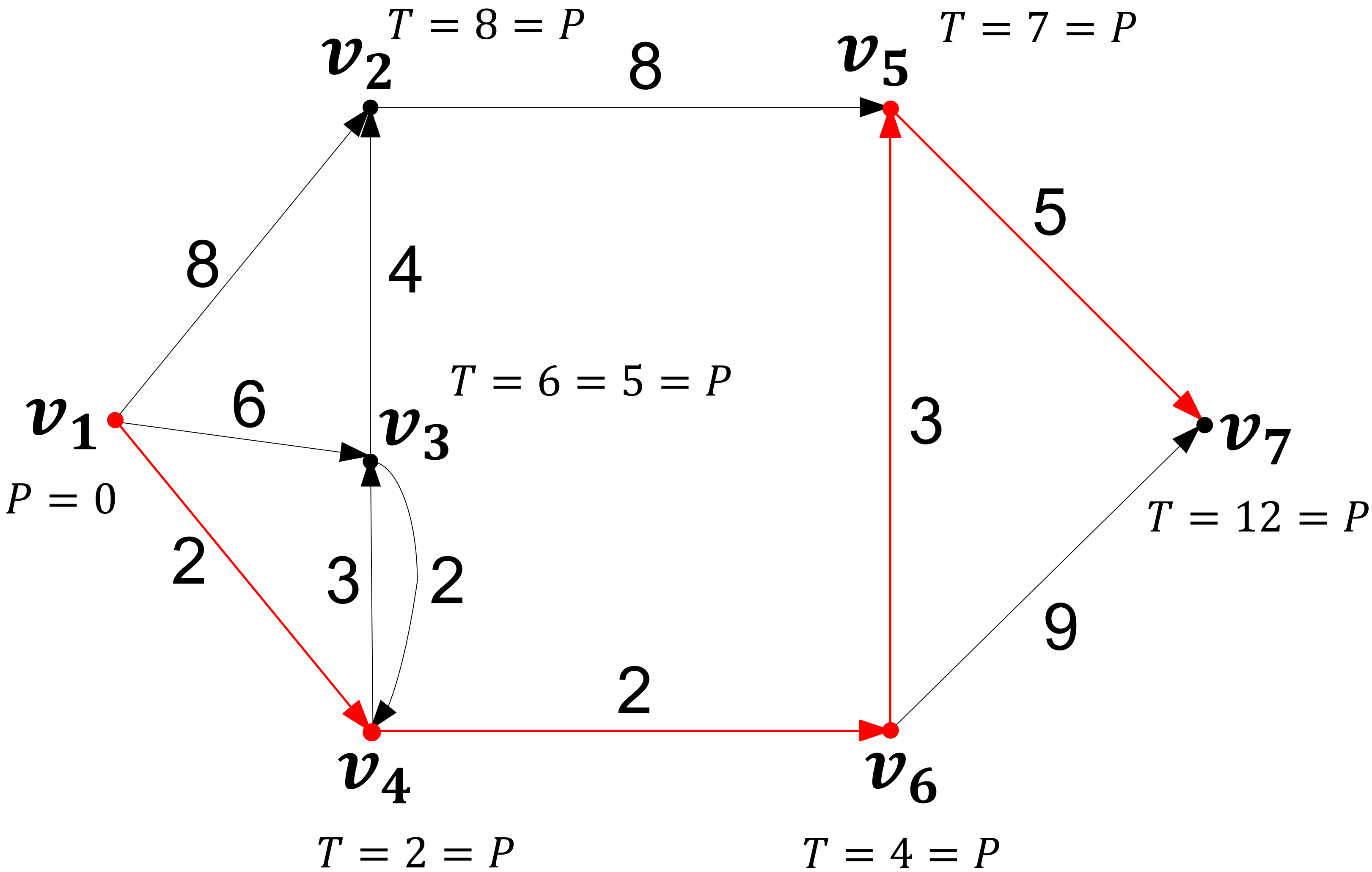
3.3 Dijkstra算法一例题



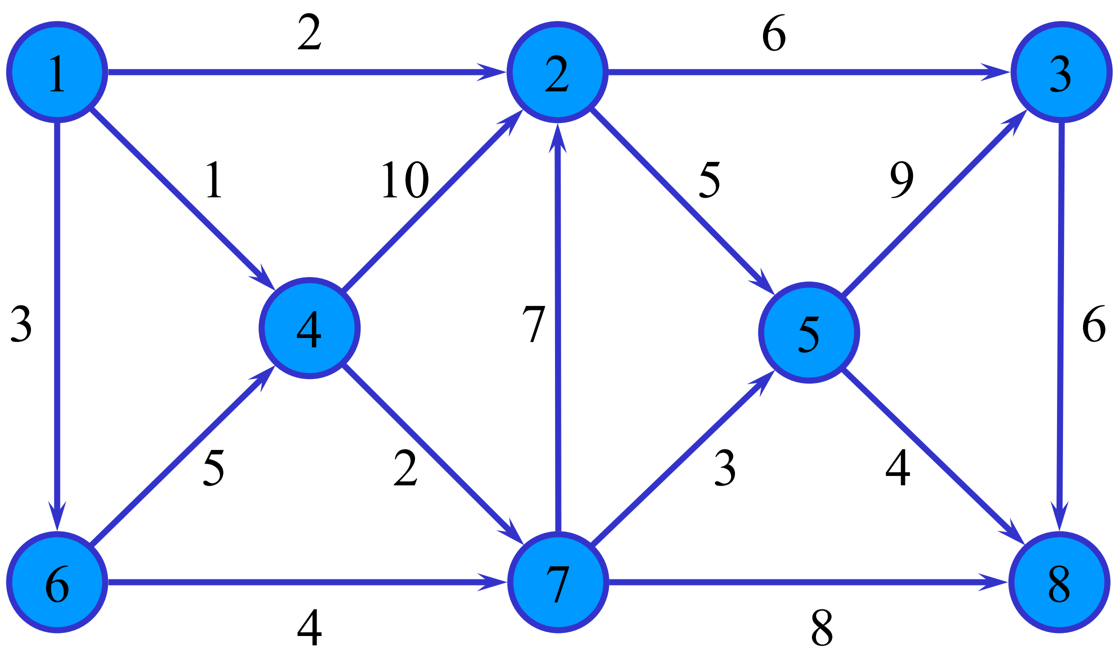
3.3 Dijkstra算法一例题



3.3 Dijkstra算法—例题

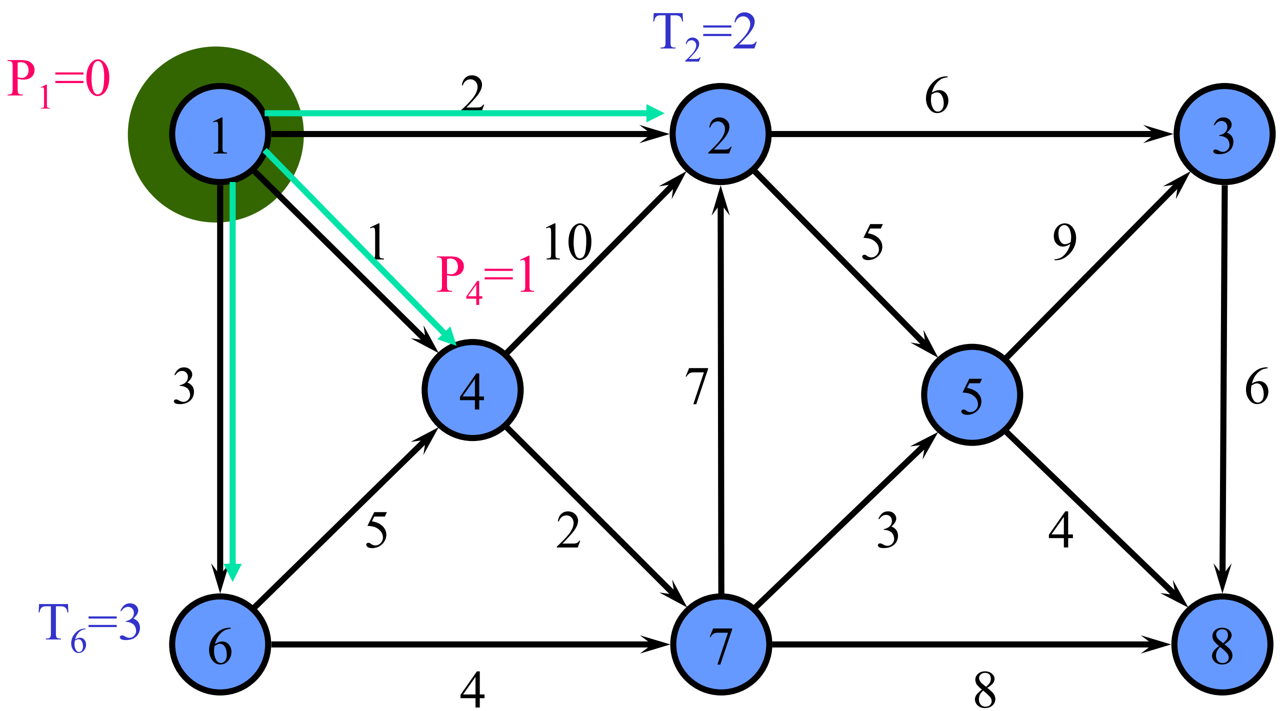


3.3 Dijkstra算法—练习



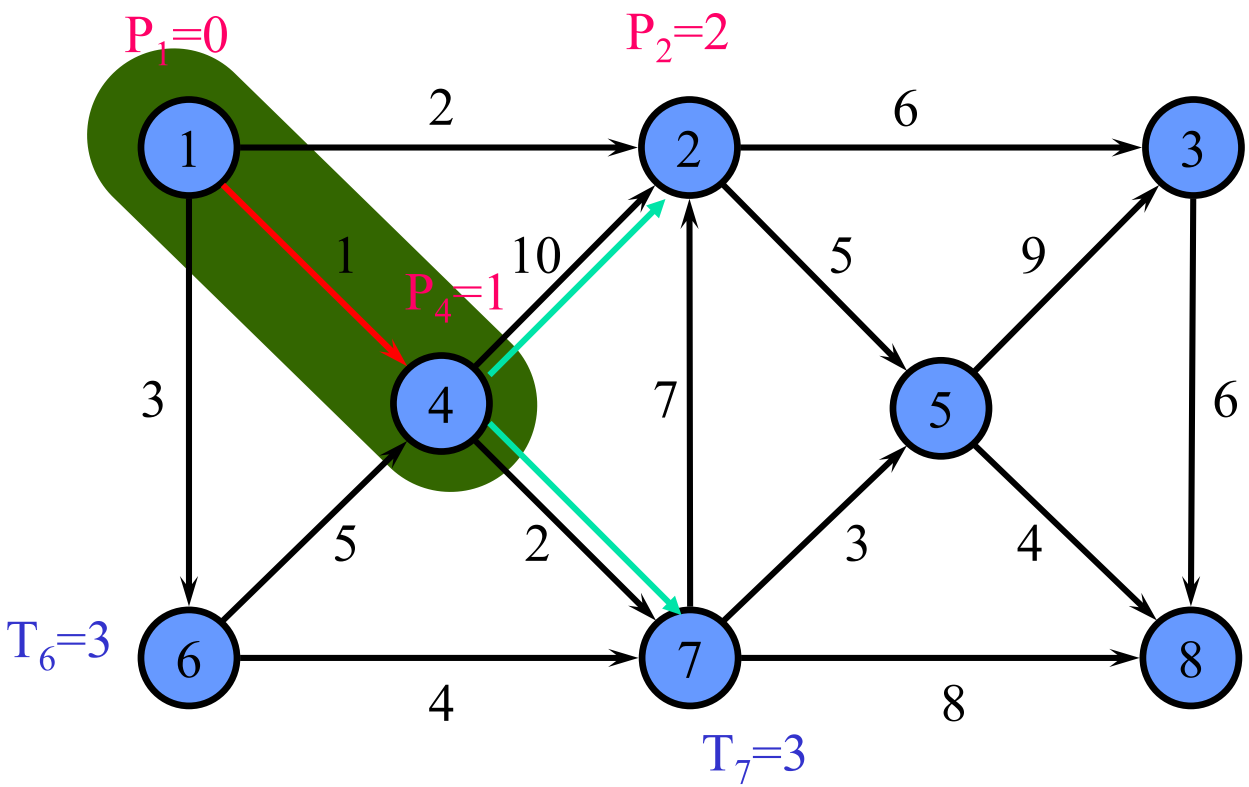
求从1到8的最短路。

f={1}, P_1=0



min {2,1,3}=1

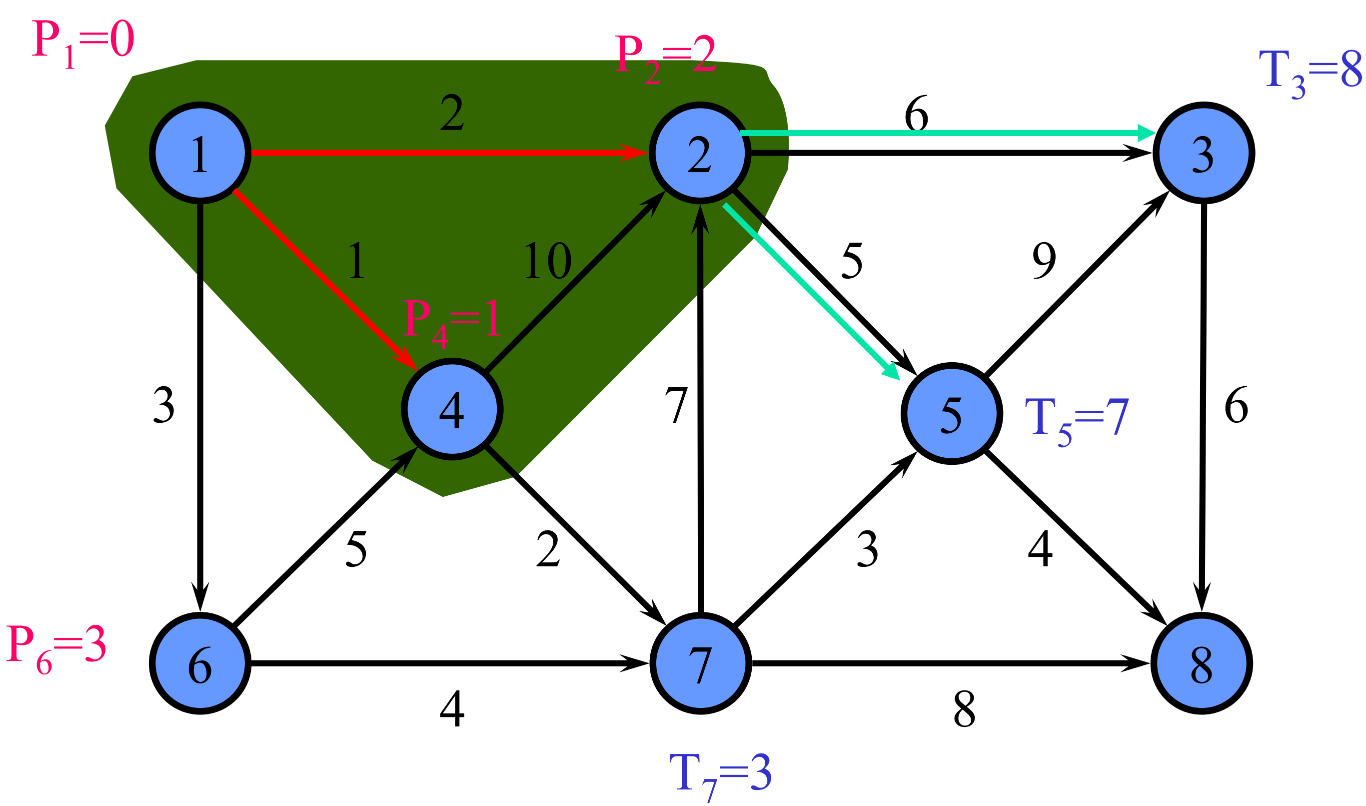
f={1,4}, P_4=1



min {2,3,3}=2

f={1,2,4}, P_2=2

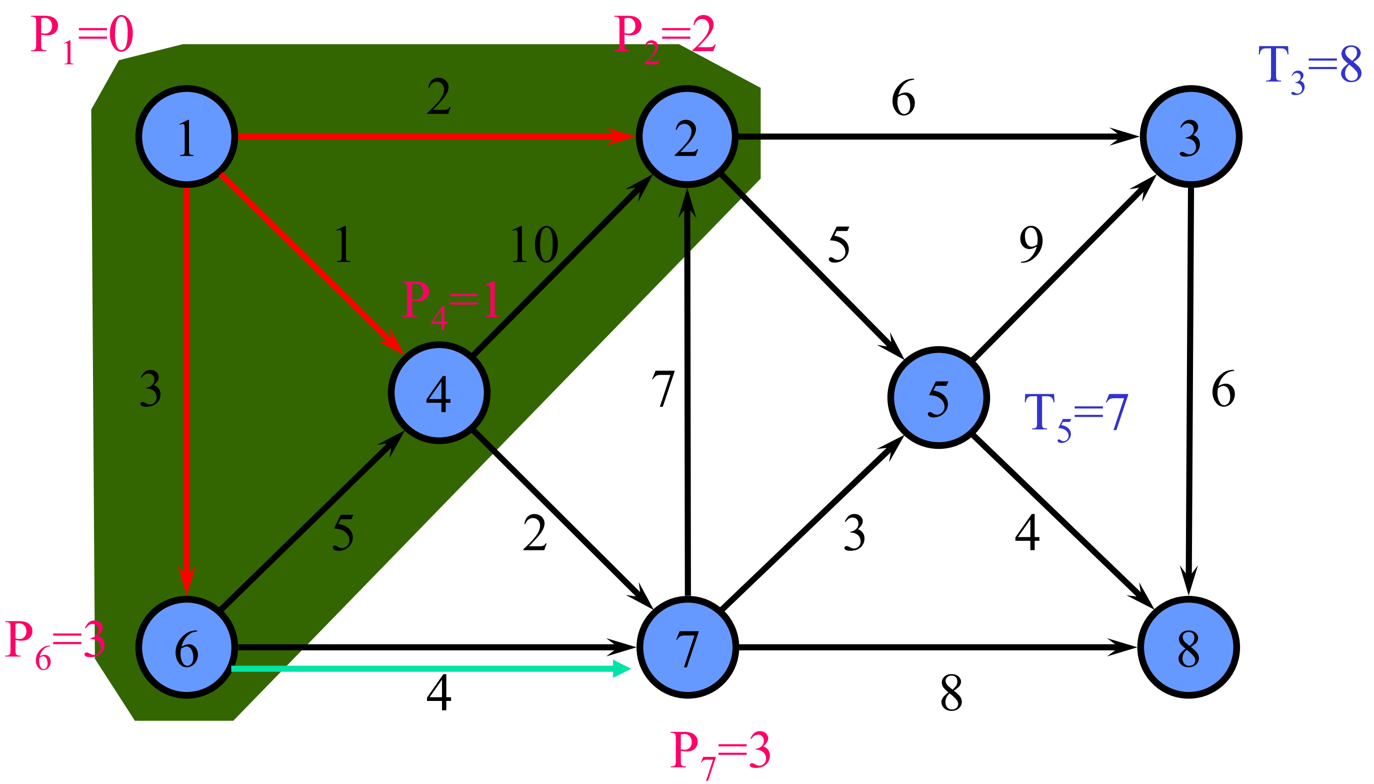
3.3 Dijkstra算法—练习



$\min\{3,8,7,3\}=3$

$f=\{1,2,4,6\}$, $P_6=3$

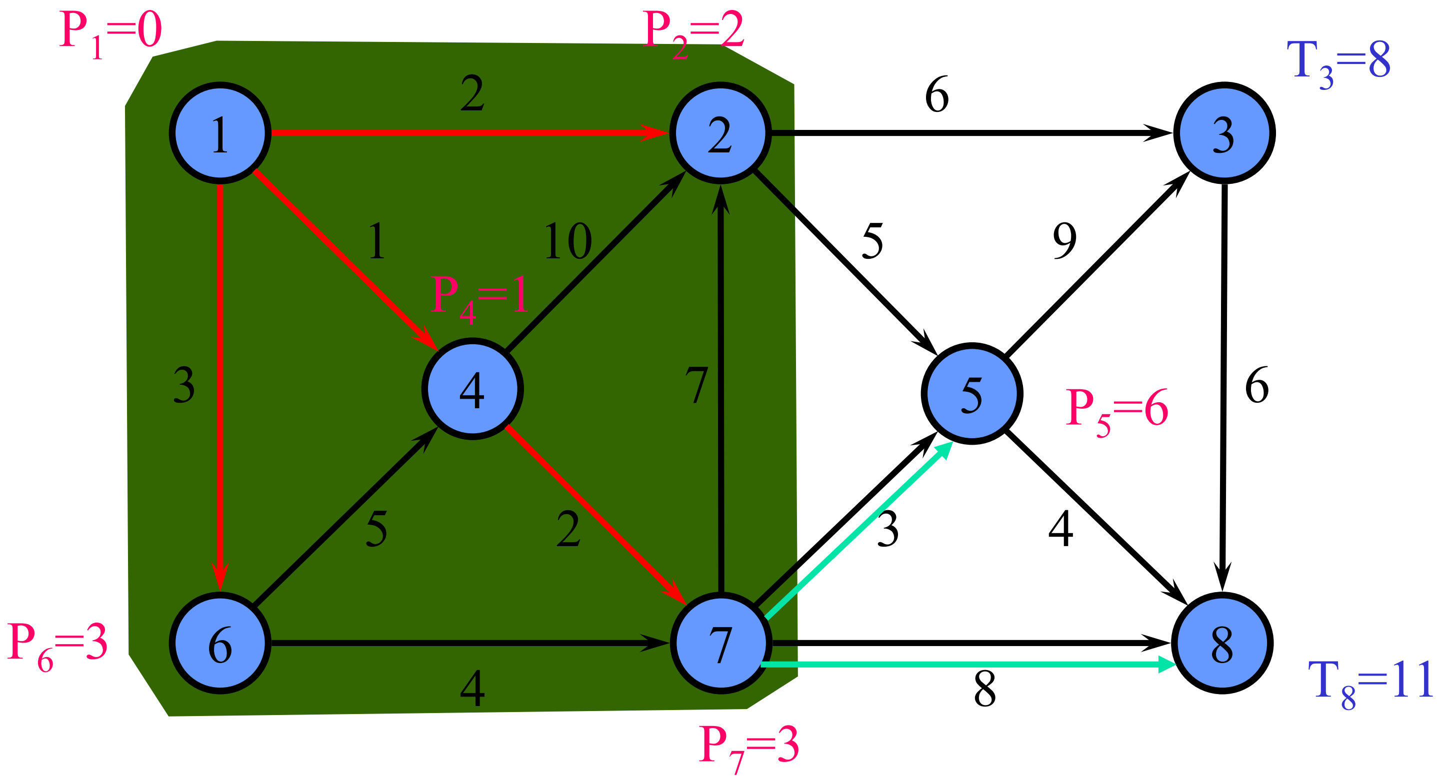
3.3 Dijkstra算法—练习



min {8,7,3}=3

f={1,2,4,6,7}, P_7=3

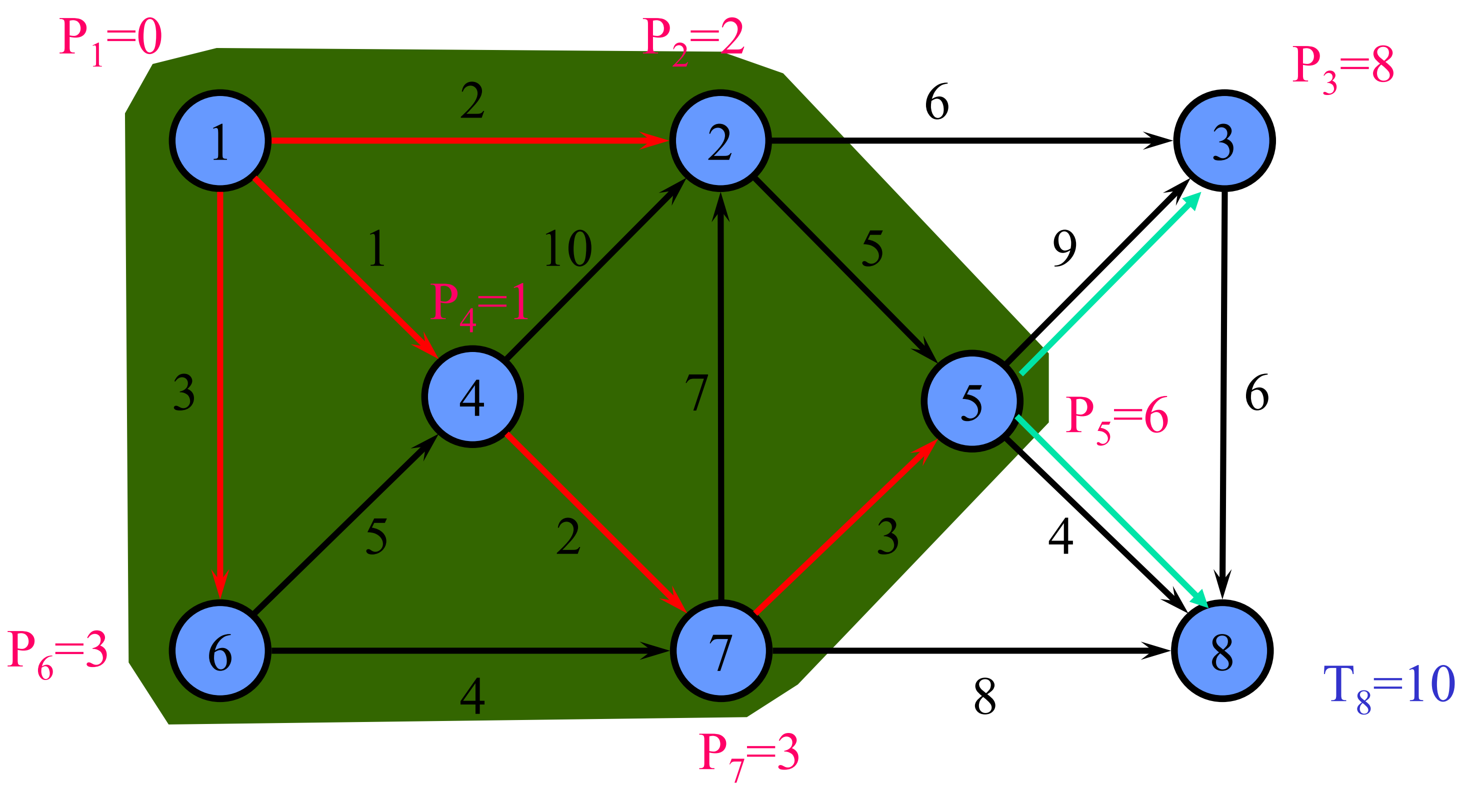
3.3 Dijkstra算法—练习



$\min\{8,6,11\}=6$

$f=\{1,2,4,5,6,7\}$, $P_5=6$

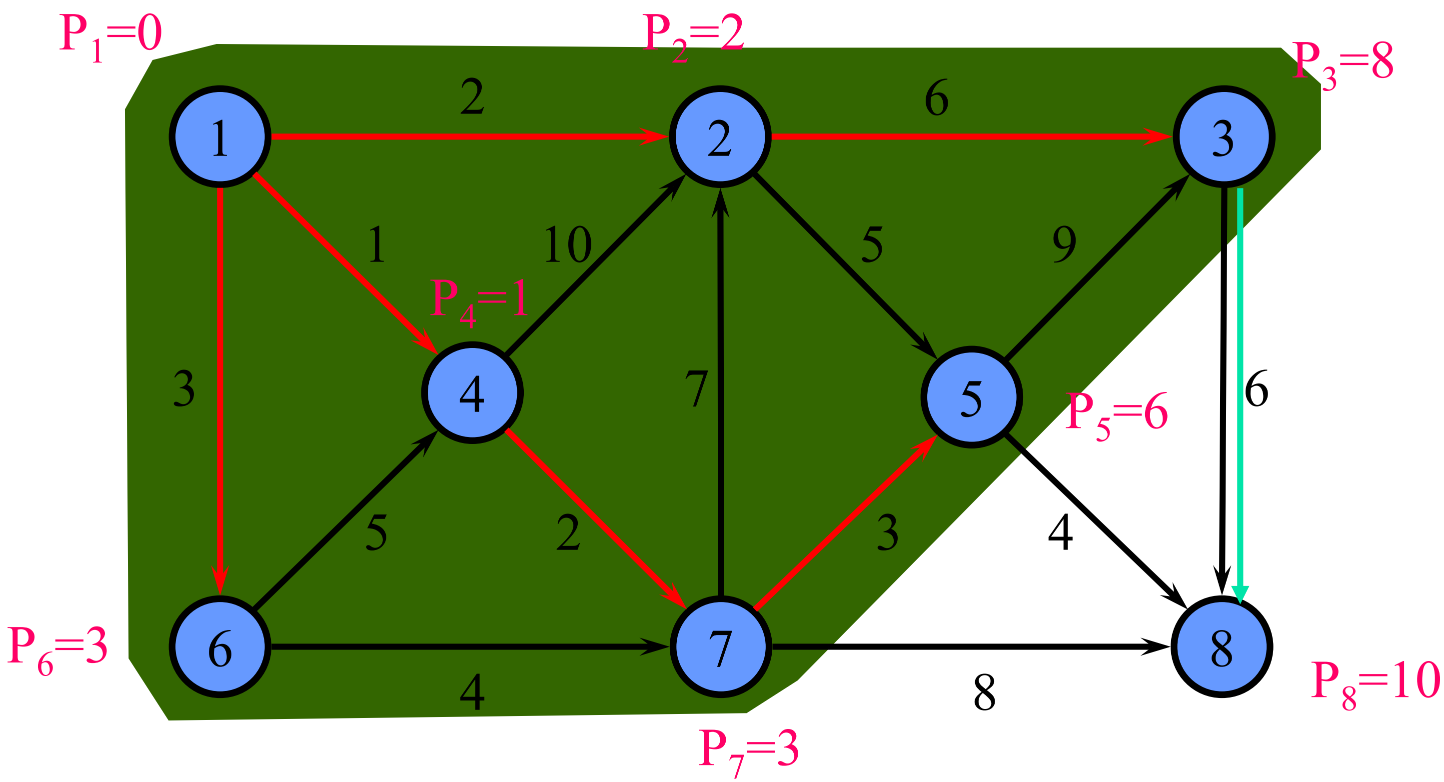
3.3 Dijkstra算法—练习



$\min\{8,10\}=8$

$f=\{1,2,3,4,5,6,7\}$, $P_3=8$

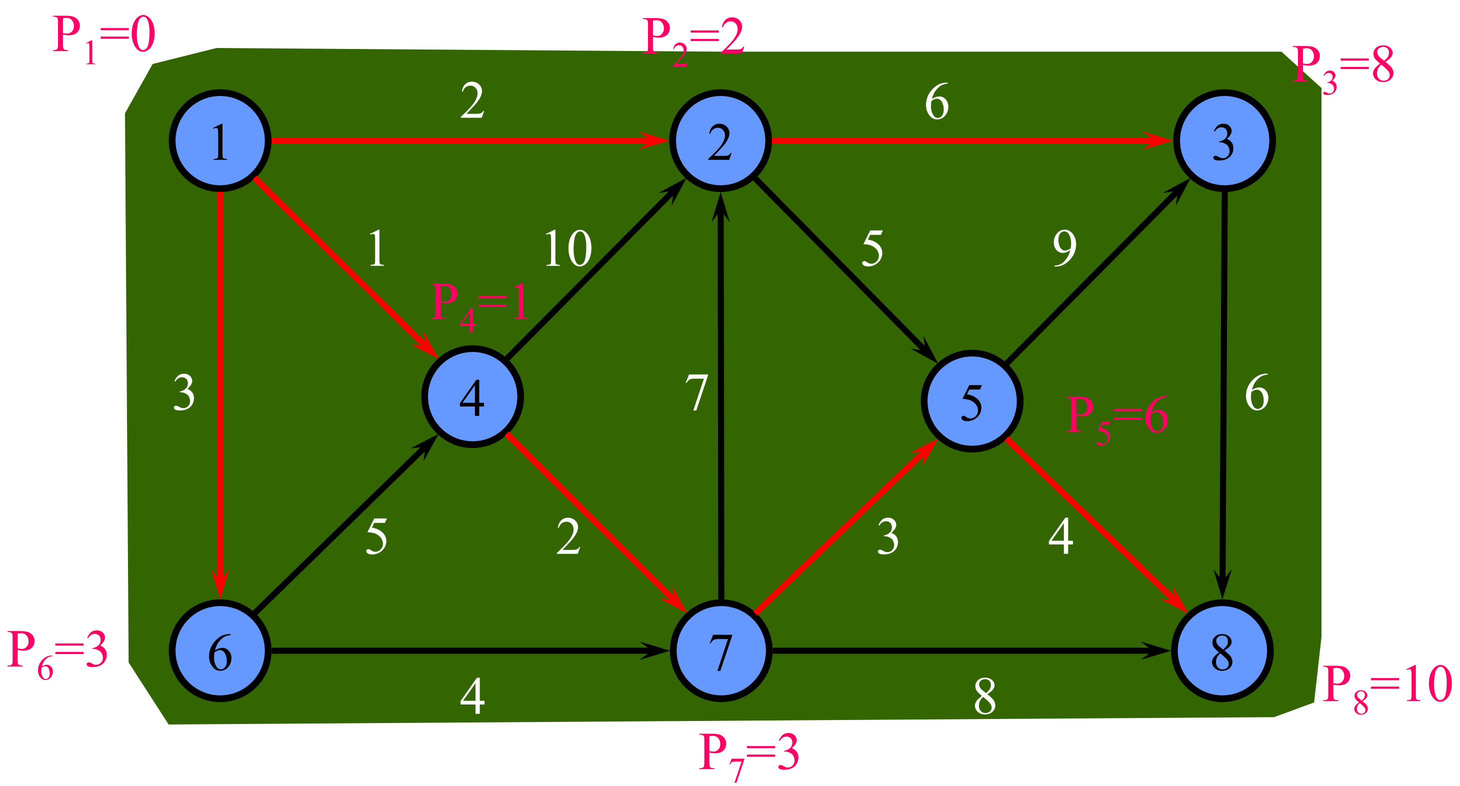
3.3 Dijkstra算法—练习



min {10}=10

f={1,2,3,4,5,6,7,8}, $P_8=10$

3.3 Dijkstra算法—练习



1到8的最短路径为{1，4，7，5，8}，长度为10。

3.4 最短路问题的LP模型

$$\min \sum_{(i,j)\in A} w_{ij}x_{ij}$$

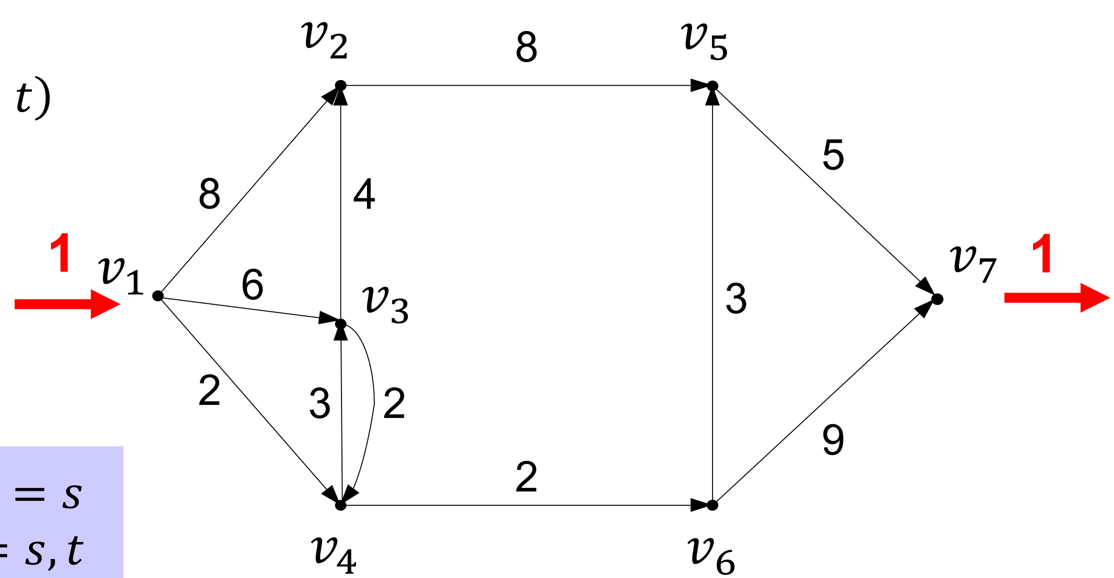
$$s.t. \begin{cases} 1 = \sum_{(i,j)\in A} x_{ij} \quad (i = s) \\ \sum_{(j,i)\in A} x_{ji} = \sum_{(i,j)\in A} x_{ij} \quad (i \neq s,t) \\ \sum_{(j,i)\in A} x_{ji} = 1 \qquad (i = t) \\ x_{ij} \geq 0 \end{cases}$$

对偶问题

$$\max \quad (u_t - u_s)$$

$$s.t. \quad u_j - u_i \leq w_{ij}, \forall (i,j) \in A$$

$$\sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} = \begin{cases} -1, i = s \\ 0, i \neq s,t \\ 1, i = t \end{cases}$$



3.4 最短路问题的LP模型

根据互补松弛条件, 当 $\boldsymbol{x}$ 和 $\boldsymbol{u}$ 分别为原问题和对偶问题的最优解时:

$$x_{ij}(u_j - u_i - w_{ij}) = 0, \qquad \forall (i,j) \in A$$

当某弧 (i,j) 位于最短路上时， 即变量 $x_{ij} > 0$ 时, 一定有 $u_j - u_i = w_{ij}$

- 如果 $\boldsymbol{u}$ 为对偶问题最优解, 易知 $\boldsymbol{u} + \boldsymbol{c}$ ($\boldsymbol{c}$为任意实数) 仍为最优解。
- 不妨令$\boldsymbol{u_s} = \mathbf{0}$, 则 $\boldsymbol{u}$ 的具体数值就可以唯一确定了。在$\boldsymbol{u_s} = \mathbf{0}$时, $\boldsymbol{u_j}$ 正好表示节点 $\boldsymbol{s}$ 到节点 $\boldsymbol{j}$ 的最短路长度。

Bellman 方程(最短路方程、动态规划基本方程)

$$\begin{cases} u_s = 0 \\ u_j = \min\limits_{i \neq j}\{u_i + w_{ij}\} \end{cases}$$

3.5 逐次逼近算法

用迭代法解这个方程：$\begin{cases} u_s = 0 \\ u_j = \min\limits_i \{u_i + w_{ij}\} \end{cases}$

- 开始时令：$u_j^{(1)} = w_{sj}$

  若v_s与v_j之间无边，则记v_s与v_j间的最短路长为$+\infty$

- 从第二步起，使用迭代公式：

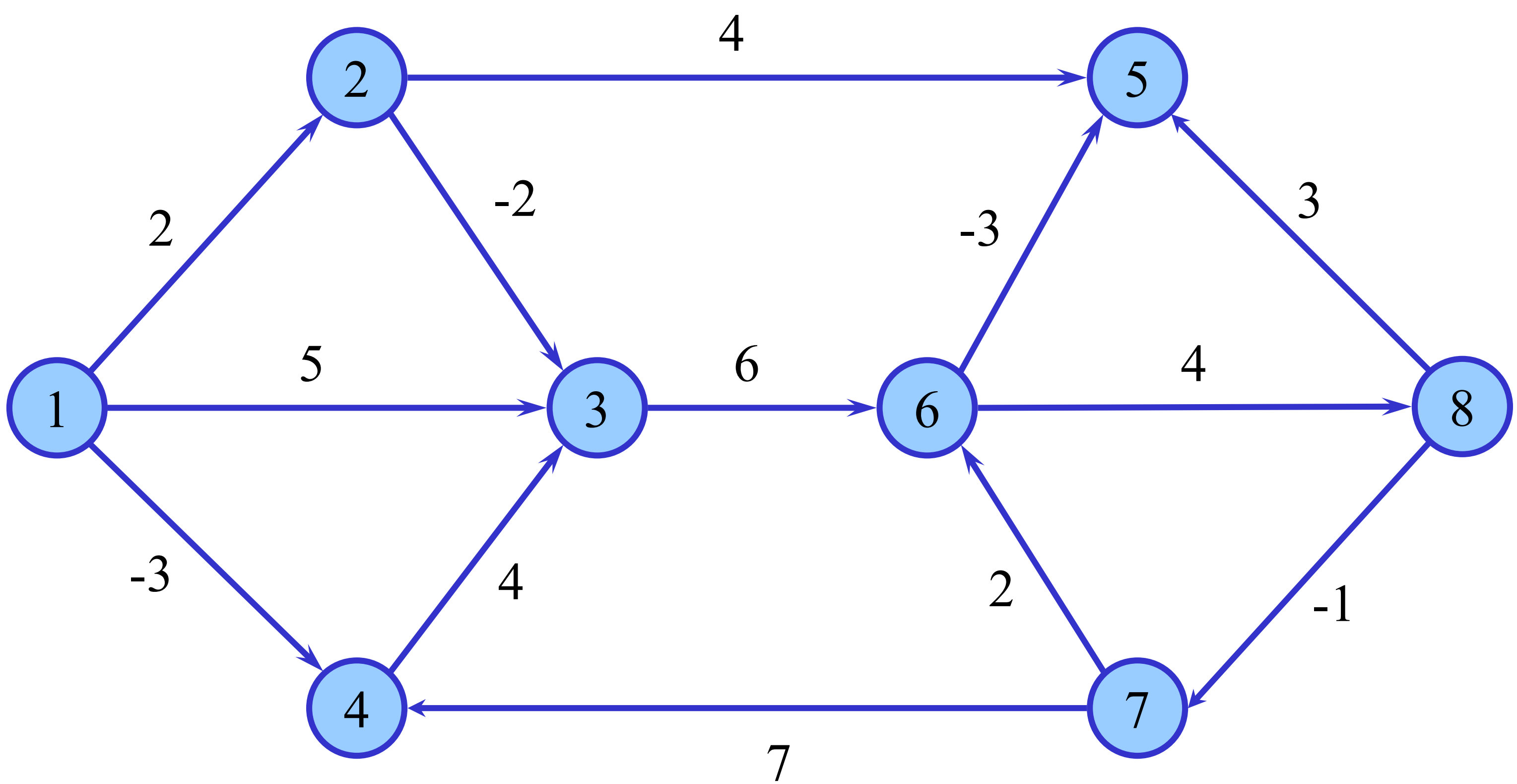
$$u_j^{(k)} = \min_i \{u_i^{(k-1)} + w_{ij}\}$$

- 当第 t 步时出现：

$$u_j^{(t)} = u_j^{(t-1)}$$

即停止，$u_j^{(t)}$就是v_s到各点v_j的最短路长。

3.5 逐次逼近算法一例题



$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_8$

$$\begin{cases} u_j^{(1)} = w_{sj} \\ u_j^{(k)} = \min\limits_i \{u_i^{(k-1)} + w_{ij}\} \end{cases}$$

3.5 逐次逼近算法

$$u_j^{(k)} = \min_i \{u_i^{(k-1)} + w_{ij}\}$$

$u_j^{(k)}$的实际意义是从v_s到v_j之间，至多含有$k-1$个中间点的最短路权。

因此，在含有 n 个点的图中，如果不含有总权小于零的回路，那么该算法最多经过 $n-1$ 次迭代必定收敛。

如果图中含有总权小于零的回路，最短路权没有下界。

Dijkstra算法

南京大学 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

标号设定算法（Label-Setting Algorithm）—— 在通过迭代过程对标号进行逐步修正的过程中，每次迭代将一个顶点从临时标号集合中移入永久标号集合中.

逐次逼近算法 (Successive Approximation Method)

标号修正算法（Label-Correcting Algorithm）—— 每次迭代时并不一定将任何顶点标号从临时标号转变为永久标号，只是对临时标号进行一次修正，所有顶点标号仍然都是临时标号；

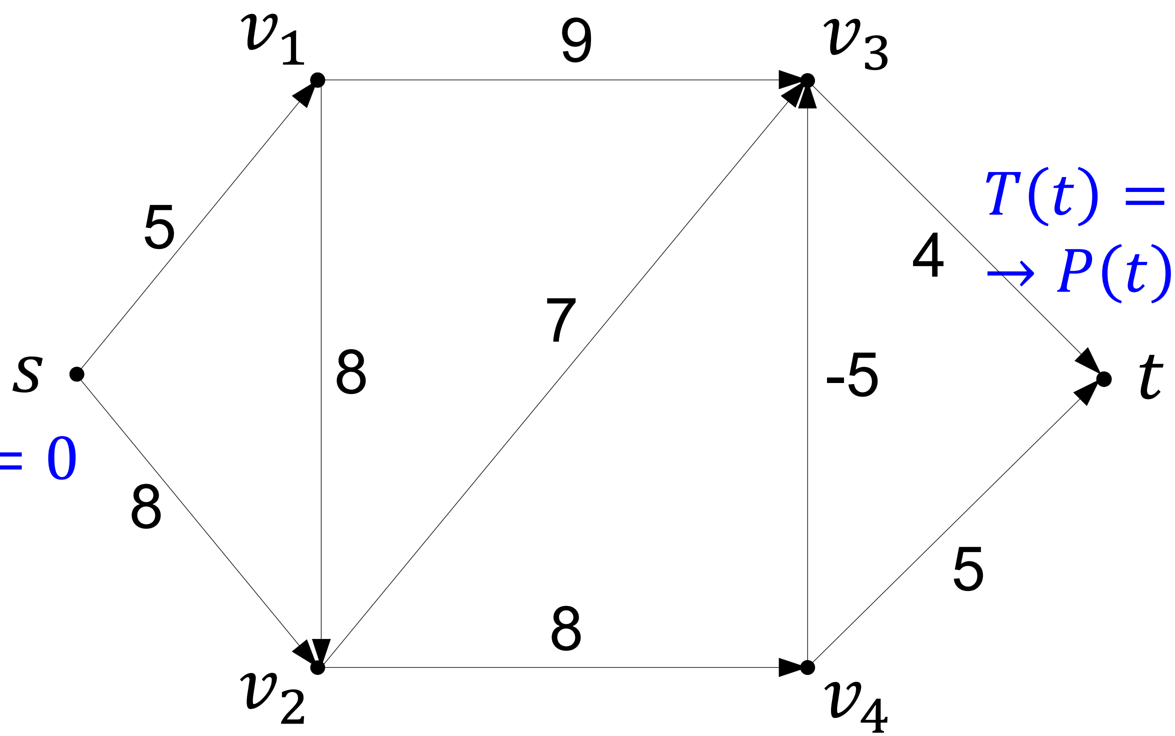
只有在所有迭代终止时，所有顶点标号同时转变为永久标号。

标号设定算法可以看成是标号修正算法的特例，因为在算法终止之前，任何永久标号都可以看成是一种特殊的临时标号。

Ford算法算例

$T(v_3) = 14 \rightarrow P(v_3) = 14$
$\rightarrow T(v_3) = 11 \rightarrow P(v_3) = 11$

$T(v_1) = 5 \rightarrow P(v_1) = 5$



$T(t) = 18 \rightarrow T(t) = 15$
$\rightarrow P(t) = 15$

$P(s) = 0$

$T(v_2) = 8 \rightarrow P(v_2) = 8$

$T(v_4) = 16 \rightarrow P(v_4) = 16$

3.6 Floyd算法

- Dijkstra算法提供了从网络图中某一点到其它点的最短路。
- 但实际问题中往往要求网络任意两点之间的最短距离，用Dijkstra算法就要对每个点分别计算，很麻烦。
- Floyd算法（基于权矩阵）

n 个顶点：$D = \left(d_{ij}\right)_{n\times n}$ $\quad d_{ij} = \begin{cases} w_{ij} & (v_i, v_j) \in E \\ \infty & o.w. \end{cases}$

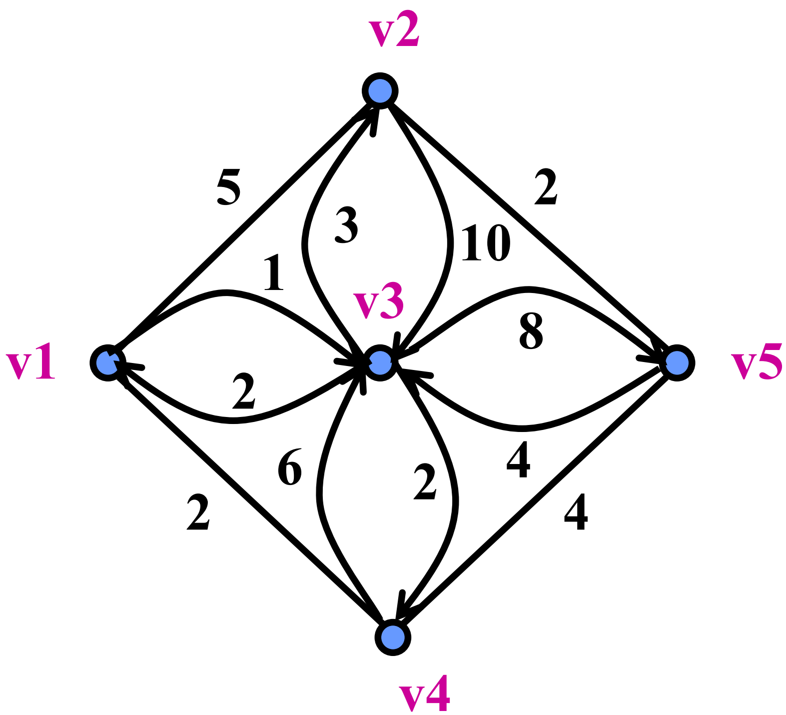
$$D^{(0)} = D \Longrightarrow D^{(k)} = \left(d_{ij}^{(k)}\right)_{n\times n}$$

$$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$$

✓ $D^{(n)} = \left(d_{ij}^{(n)}\right)_{n\times n}$ 的元素就是 v_i 到 v_j 的最短路权。

3.6 Floyd算法—例题

例：求图中各点之间的最短距离。



$$D^{(0)}$$

$$
\begin{array}{c}
\\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5
\end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix}
0 & 5 & 1 & 2 & \infty \\
5 & 0 & 10 & \infty & 2 \\
2 & 3 & 0 & 2 & 8 \\
2 & \infty & 6 & 0 & 4 \\
\infty & 2 & 4 & 4 & 0
\end{pmatrix}
\end{array}
$$

3.6 Floyd算法—例题

南京大學 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

$D^{(0)}$

$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline v_1 & 0 & 5 & 1 & 2 & \infty \\ v_2 & 5 & 0 & 10 & \infty & 2 \\ v_3 & 2 & 3 & 0 & 2 & 8 \\ v_4 & 2 & \infty & 6 & 0 & 4 \\ v_5 & \infty & 2 & 4 & 4 & 0 \end{array}$$

$D^{(1)}$

$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline v_1 & 0 & 5 & 1 & 2 & \infty \\ v_2 & 5 & 0 & 6 & 7 & 2 \\ v_3 & 2 & 3 & 0 & 2 & 8 \\ v_4 & 2 & 7 & 3 & 0 & 4 \\ v_5 & \infty & 2 & 4 & 4 & 0 \end{array}$$

$D^{(2)}$

$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline v_1 & 0 & 5 & 1 & 2 & 7 \\ v_2 & 5 & 0 & 6 & 7 & 2 \\ v_3 & 2 & 3 & 0 & 2 & 5 \\ v_4 & 2 & 7 & 3 & 0 & 4 \\ v_5 & 7 & 2 & 4 & 4 & 0 \end{array}$$

$D^{(3)}$

$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hline v_1 & 0 & 4 & 1 & 2 & 6 \\ v_2 & 5 & 0 & 6 & 7 & 2 \\ v_3 & 2 & 3 & 0 & 2 & 5 \\ v_4 & 2 & 6 & 3 & 0 & 4 \\ v_5 & 6 & 2 & 4 & 4 & 0 \end{array}$$

3.6 Floyd算法—例题

$$D^{(3)}$$

$$\begin{array}{c|c} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 4 & 1 & 2 & 6 \\ 5 & 0 & 6 & 7 & 2 \\ 2 & 3 & 0 & 2 & 5 \\ 2 & 6 & 3 & 0 & 4 \\ 6 & 2 & 4 & 4 & 0 \end{pmatrix} \end{array}$$

$$D^{(4)}$$

$$\begin{array}{c|c} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 4 & 1 & 2 & 6 \\ 5 & 0 & 6 & 7 & 2 \\ 2 & 3 & 0 & 2 & 5 \\ 2 & 6 & 3 & 0 & 4 \\ 6 & 2 & 4 & 4 & 0 \end{pmatrix} \end{array}$$

$$D^{(5)}$$

$$\begin{array}{c|c} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{pmatrix} 0 & 4 & 1 & 2 & 6 \\ 5 & 0 & 6 & 6 & 2 \\ 2 & 3 & 0 & 2 & 5 \\ 2 & 6 & 3 & 0 & 4 \\ 6 & 2 & 4 & 4 & 0 \end{pmatrix} \end{array}$$

第七章 图与网络

4. 最大流问题

4.1 网络流的基本概念

容量网络

- 对网络流的研究是在容量网络上进行的
- 容量网络是指网络上的每一条弧（v_i, v_j）都有一个最大的通过能力，称为该弧的容量 $c(v_i, v_j)$，c_{ij}
- 在网络图中通常规定一个发点 (s) 和一个收点 (t)，其余为中间点
- 网络最大流是指从发点到收点间允许通过的最大流量
- 对有多个发点或收点的网络，另虚设一个总发点或总收点

流与可行流

- 流是加在网络各条弧上的一组负载量
- 对加在弧 (v_i, v_j) 上的负载量记作 $f(v_i, v_j)$，f_{ij}
- 若网络上所有的 $f_{ij} = 0$，则这个流称为零流
- 满足容量限制条件和平衡条件的一组流为可行流
  - ✓ 容量限制条件：对所有弧有 $0 \le f_{ij} \le c_{ij}$
  - ✓ 中间点平衡条件：$\sum f_{ij} - \sum f_{ji} = 0 \ (i \ne s, t)$
- 若以W表示网络中从 s → t 的流量，则有

$$W = \sum_j f(v_s, v_j) = \sum_j f(v_j, v_t)$$ ✓ 收发点平衡条件

4.1 网络流的基本概念

- 任何网络上一定存在可行流（零流就是可行流）
- 所谓求网络最大流，是指满足容量限制条件和中间点平衡条件下，使W值达到最大
- 这是一个线性规划问题，但图论方法更简单

$$\max \ W$$

$$\sum_{(s,j)\in A} f_{sj} - \sum_{(j,s)\in A} f_{js} = W$$

$$\sum_{(t,j)\in A} f_{tj} - \sum_{(j,t)\in A} f_{jt} = -W$$

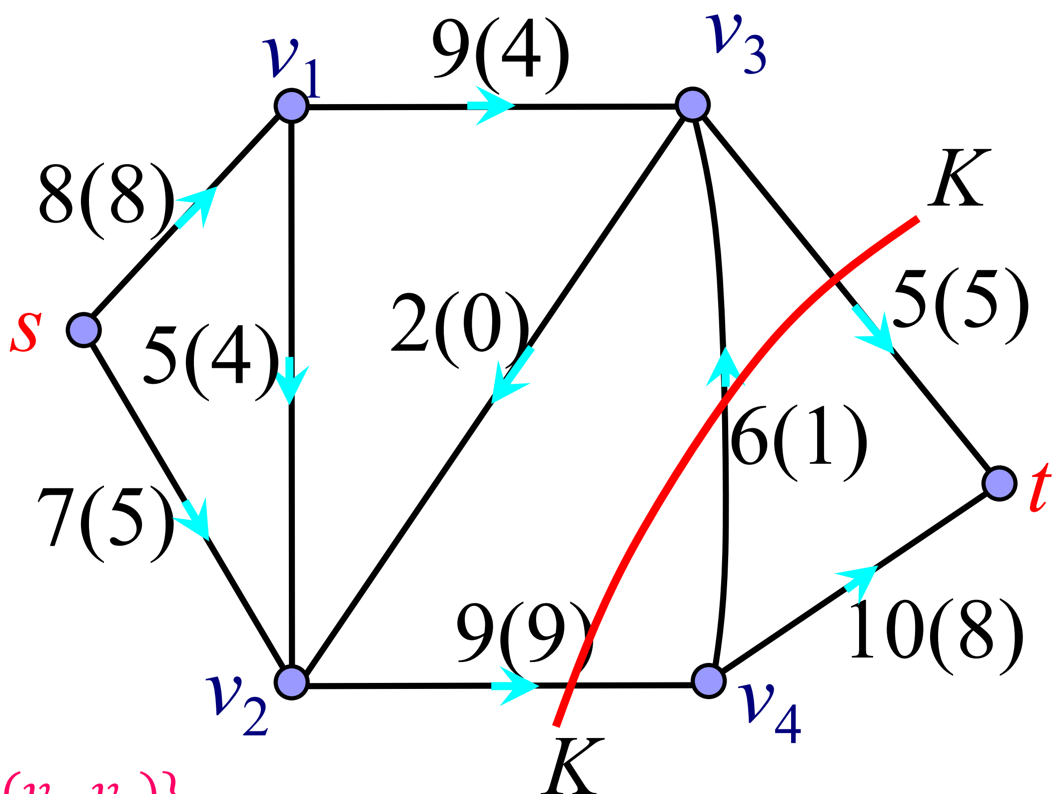
$$\sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = 0, i \neq s, t$$

$$0 \leq f_{ij} \leq C_{ij}$$

4.2 割和流量

- 割：将容量网络中的发点和收点分开，并使发点到收点的流中断的**一组边的集合**。

KK 将网络上的点分割成 V 和 V' 两个集合，且有 $s \in V, t \in V'$。



边的集合

$$(V, V') = \{(v_3, t), (v_4, v_3), (v_2, v_4)\}$$

是一个割。

4.2 割和流量

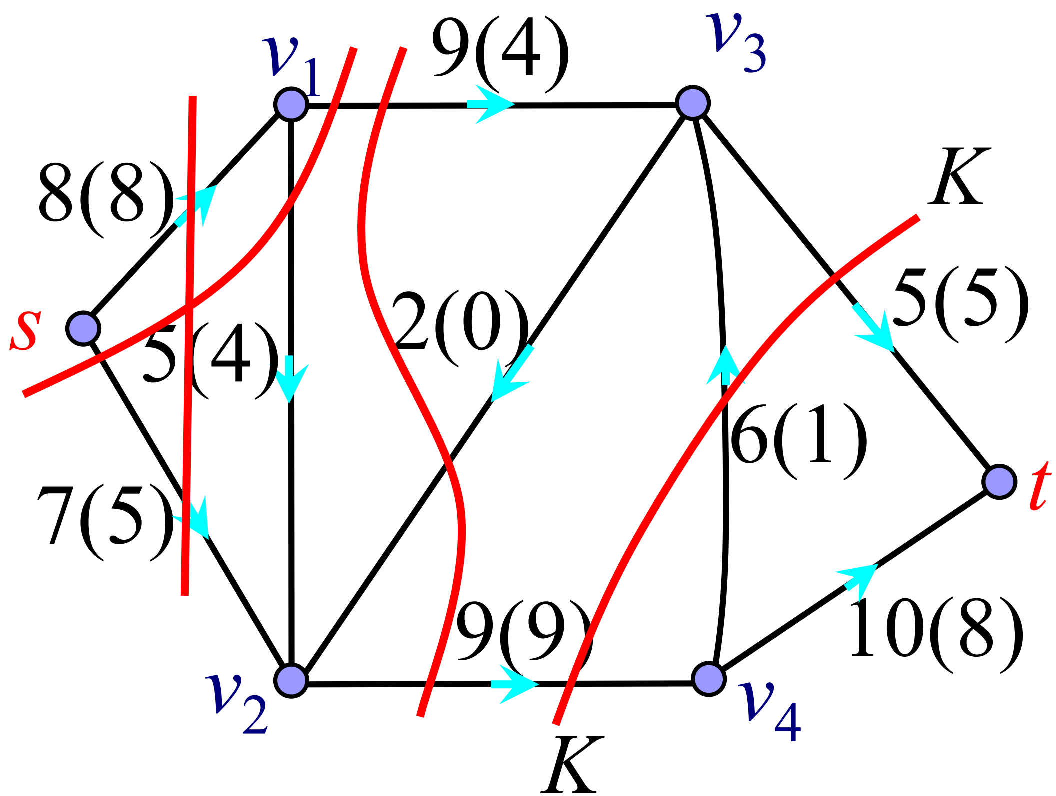
- 割集容量是指所有始点在 V，终点在 V' 的弧的容量之和

$7+8=15$

$9+5+7=21$

$9+9=18$

$5+9=14$

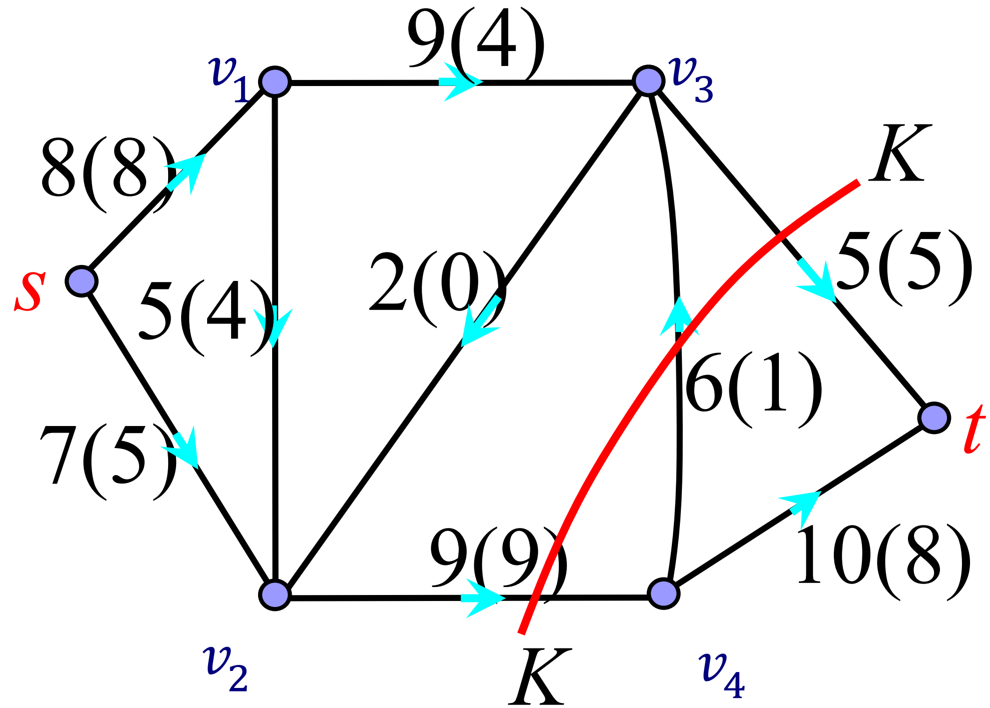


割集有多个，其中割集容量最小者称为最小割。

4.2 割和流量

$$f(V,V') = \sum_{(i,j)\in(V,V')} f(v_i,v_j)$$

$$f(V',V) = \sum_{(j,i)\in(V',V)} f(v_j,v_i)$$



若用

- $f(V,V')$表示通过割 (V,V') 中所有 $V \to V'$ 方向弧的流量的总和
- $f(V',V)$表示通过割 (V,V') 中所有 $V' \to V$ 方向弧的流量的总和

➢ 从 $s \to t$ 的流量实际上等于通过割的从$V \to V'$的流量减去$V' \to V$的流量，即

$$W = f(V,V') - f(V',V)$$

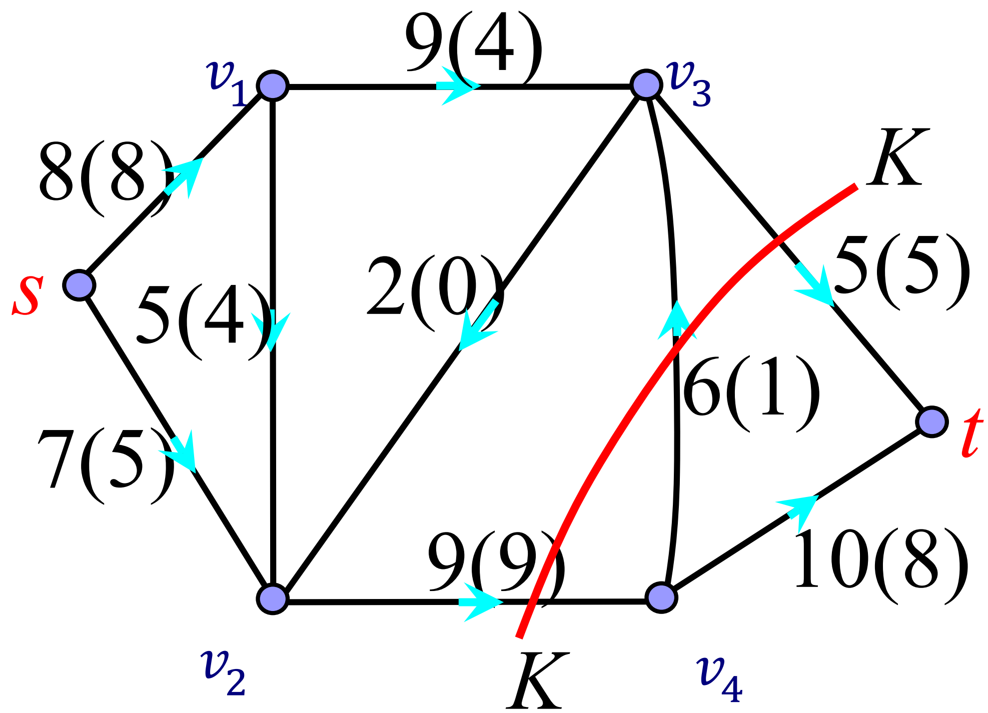
4.2 割和流量

- 割集是从 $s \to t$ 的必经之路
- 任何一个可行流的流量不会超过任一割集的容量

$$W \le C(V, V')$$

$$\Downarrow$$

$$\max W \le \min C(V, V')$$



- 若用 W^* 代表网络中从 $s \to t$ 的最大流，$C^*(V, V')$ 代表网络中最小的一个割的容量，则

$$W^* = C^*(V, V')$$

—— 最大流-最小割定理

前向弧，后向弧

如果在网络的发点和收点之间存在一条链，那么在这条链上

- 所有指向为“发点到收点”的弧称为前向弧，记为μ^+
- 所有指向为相反的弧称为后向弧，记为μ^-

增广链

如果

- 在所有的前向弧上，流量 $<$ 容量；
- 在所有的后向弧上，流量 >0；

则称这条链为增广链(不饱和链)

4.3 增广链(不饱和链)

- 当有增广链存在时，找出

$$\theta = \min\left\{\begin{matrix} (c_{ij} - f_{ij}), & for\ all\ \mu^{+} \\ f_{ij}, & for\ all\ \mu^{-} \end{matrix}\right\}, \quad \theta > 0$$

再令 $$f'_{ij} = \begin{cases} f_{ij} + \theta, & for\ all\ \mu^{+} \\ f_{ij} - \theta, & for\ all\ \mu^{-} \\ f_{ij}, & otherwise \end{cases}$$

- 显然 f' 仍然是一个可行流，但比原来的可行流 f 在 $s \to t$ 方向上增大了一个正的 θ 值；
- 因此只有当网络上找不到增广链时，$s \to t$ 的流才不可能再增大。

4.3 增广链(不饱和链)

- **定理：**可行流 f 为最大流的充分必要条件是当且仅当网络不存在增广链。

- ✓ 给出一初始可行流，例如 $f_{ij}=0$；
- ✓ 寻找增广链，若存在，则通过该增广链调整、增加网络流；
- ✓ 若不存在增广链，则网络流不可再增加，求得最大流。

□ 标号算法 Ford-Fulkerson

1. 首先给发点标号$(0, \delta_s)$

使这个点得到标号的前一个点的代号。因为s是发点，故记为0

δ_s是从上一标号点到这个标号点的流量的最大允许调整值。因为s是发点，不允许调整，故$\delta_s = +\infty$

2. 列出与已标号点相邻的所有未标号点：

(1) 考虑从标号点 i 出发的所有弧(i, j)

- 如果 $f_{ij}= c_{ij}$，不给点 j 标号
- 若 $f_{ij} < c_{ij}$ 对 j 标号，记为$(+v_i, \delta_j)$, 其中 $\delta_j = \min\{\delta_i, c_{ij} - f_{ij}\}$

(2) 考虑所有指向点 i 的弧(j, i)

- 如果 $f_{ji} = 0$，不给点 j 标号
- 若 $f_{ji} > 0$ 对 j 标号，记为$(-v_i, \delta_j)$, 其中 $\delta_j = \min\{\delta_i, f_{ji}\}$

如果某未标号点 k 有两个以上相邻的标号点，为减少迭代次数，可按(1)(2)分别计算出 δ_k 值，并取其中最大一个标记。

4.4 求网络最大流的标号算法

3. 重复第2步，可能出现两种结局：

(1) 标号过程中断，点 t 得不到标号，说明该网络中不存在增广链，给定的流量即为最大流

- 记已标号点的集合为 V，未标号点集合为 V'
- (V, V') 即为网络的最小割

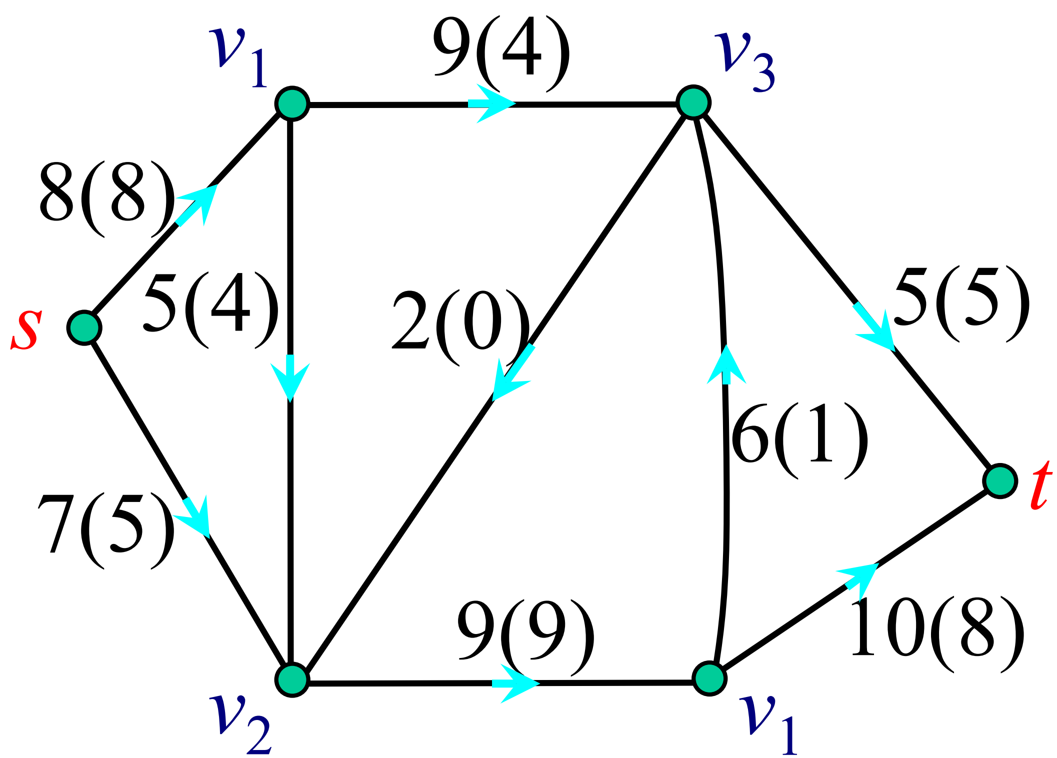
(2) 点 t 得到标号，这时可用反向追踪法在网络中找出一条从 $s \to t$ 的由标号点及相应弧连接而成的增广链

4. 修改流量。设图中原有可行流为 f，令

$$f' = \begin{cases} f + \delta(t), & \text{对增广链上所有前向弧} \\ f - \delta(t), & \text{对增广链上所有后向弧} \\ f, & \text{对所有非增广链上的弧} \end{cases}$$

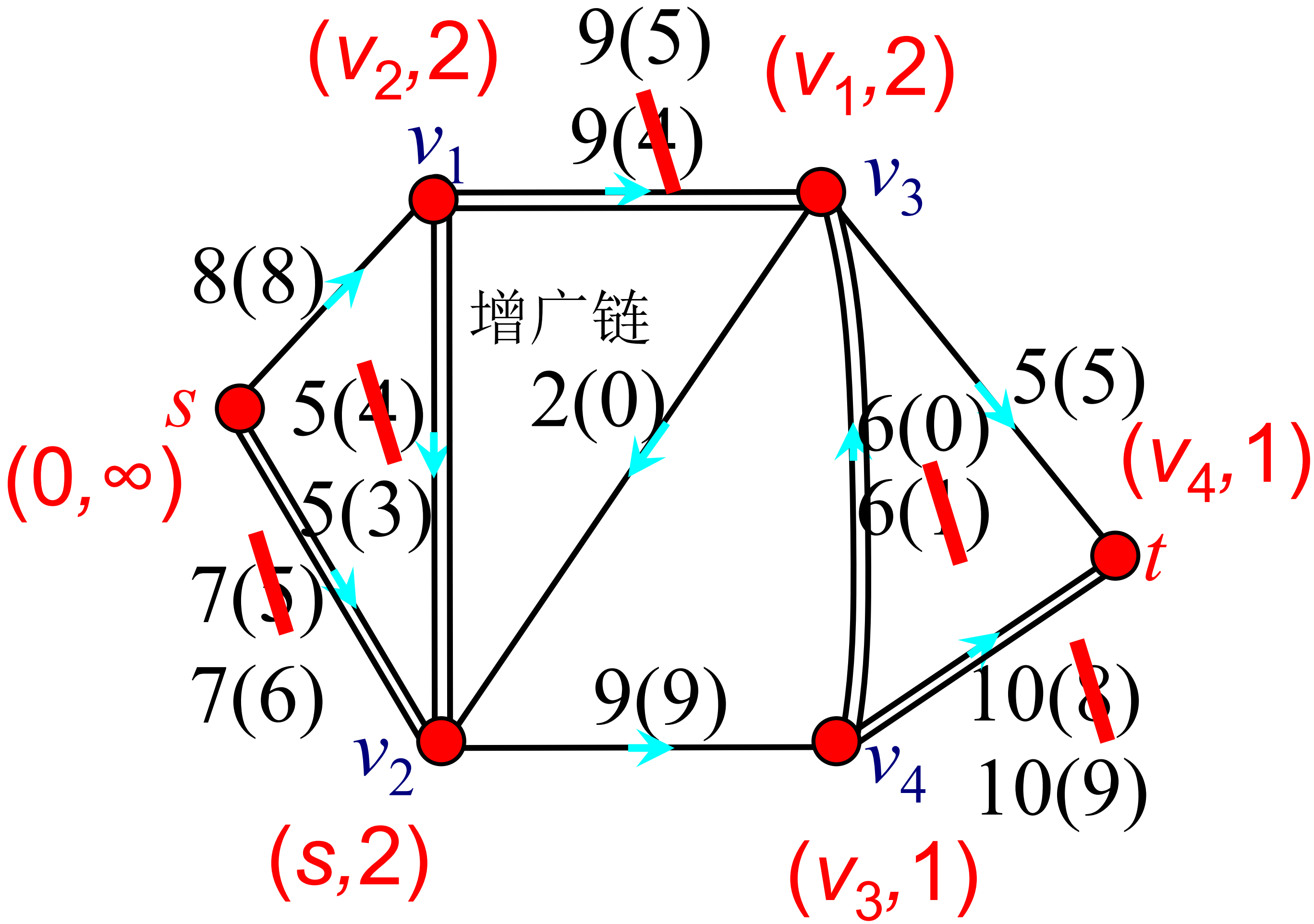
5. 抹掉图上所有标号。

重复第1到第4步，直到图中找不出任何增广链，即第3步中的(1)为止，这时网络图中的流量即为最大流。



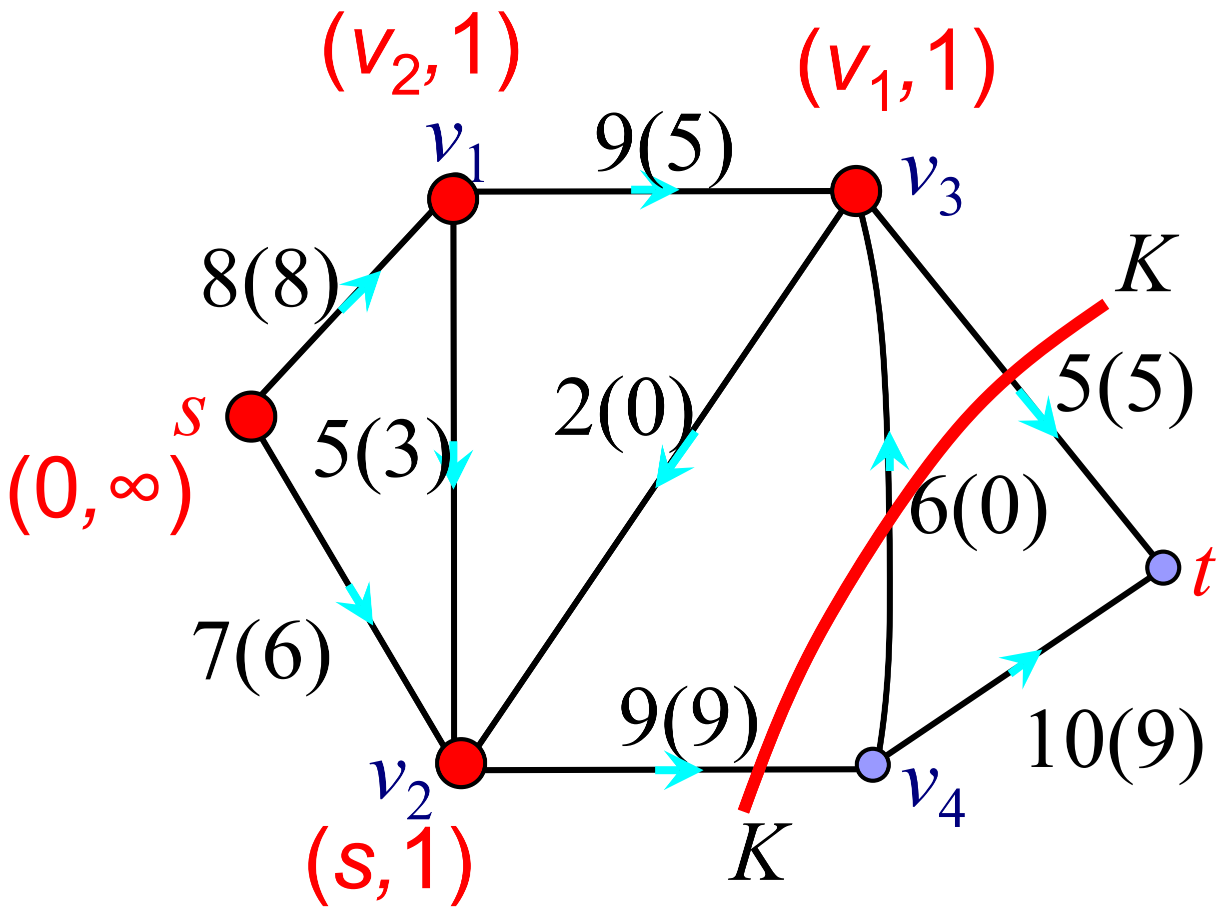
4.4 求网络最大流的标号算法

例1：



例1：

重复上述标号过程

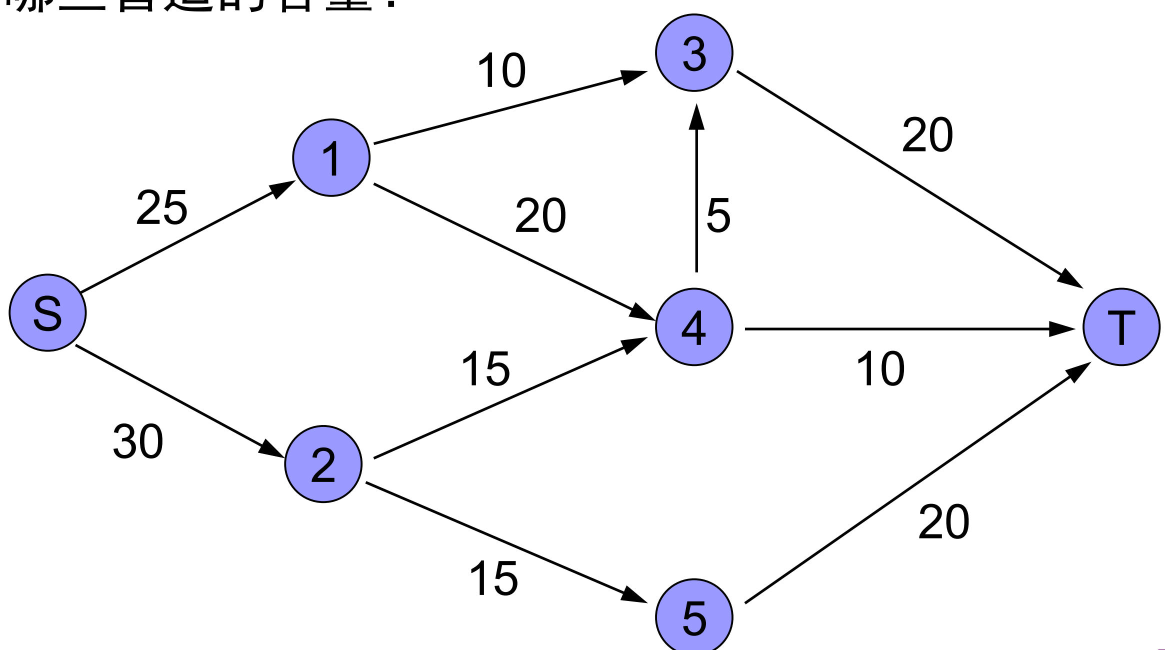


已再无增广链，故可行流即为最大流 $W^* = 14$

例2:

在下面的输油管线网络图中，弧代表油管，接点代表管道连接处的油泵，各弧上的数字代表相应的管道容量，问:

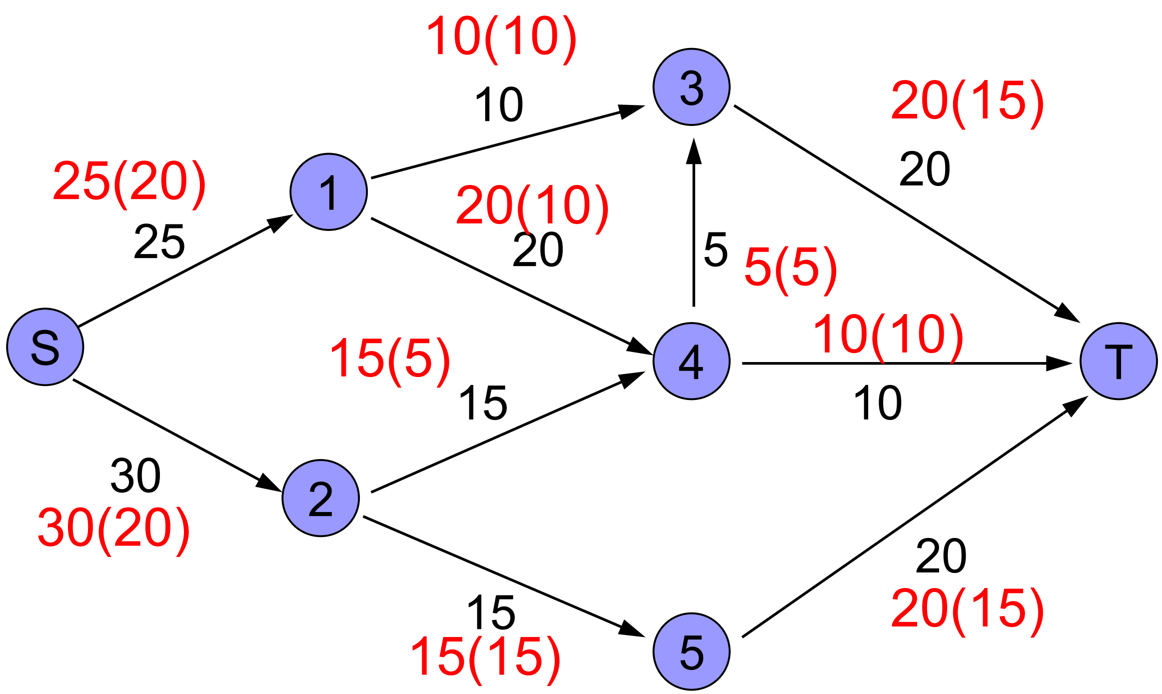
1. 从源S到终点T的最大流量是多少?
2. 为了增加从S到T的流量，同时又希望尽可能少地改变网络中的管道，应当改变哪些管道的容量?



4.4 求网络最大流的标号算法

1. 先找可行流，再求最大流

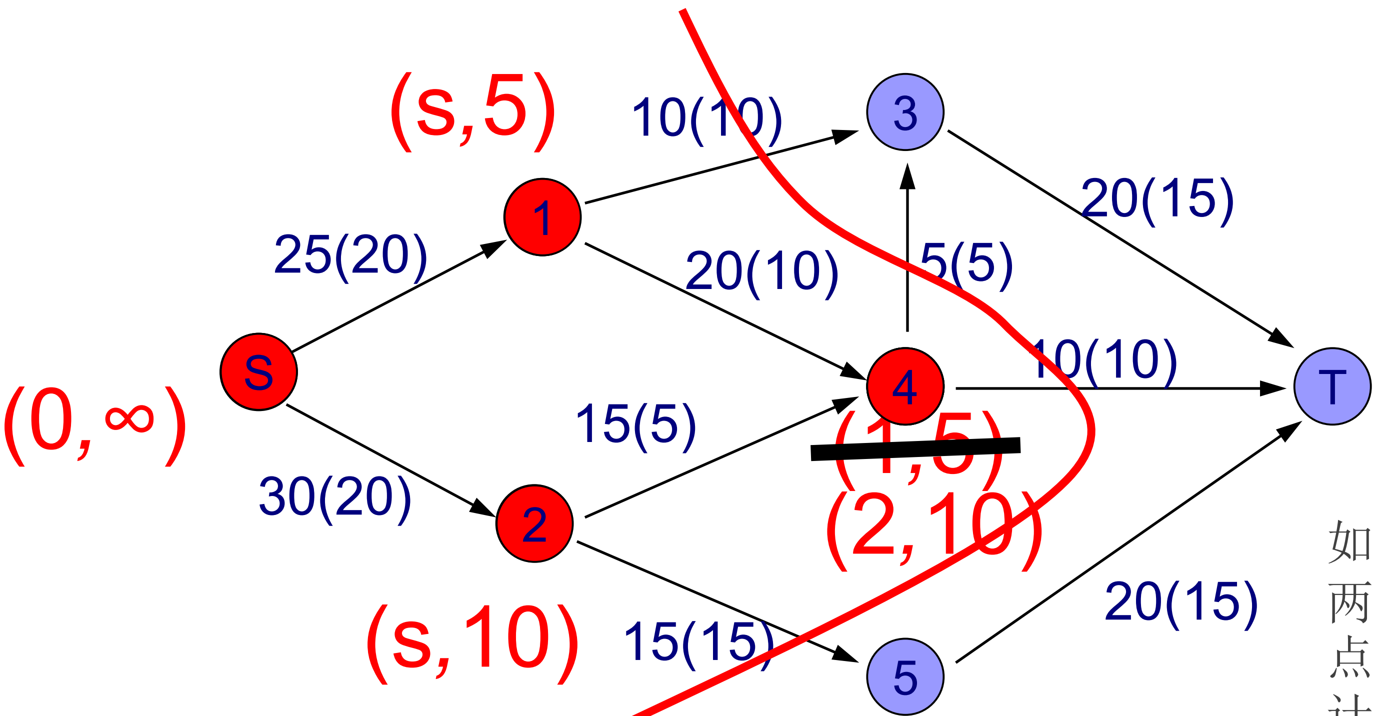
✓ 从后往前，注意中间点平衡



现在可行流量为多少？

- 40
- 看流进的或流出的

✓ 标号法求最大流



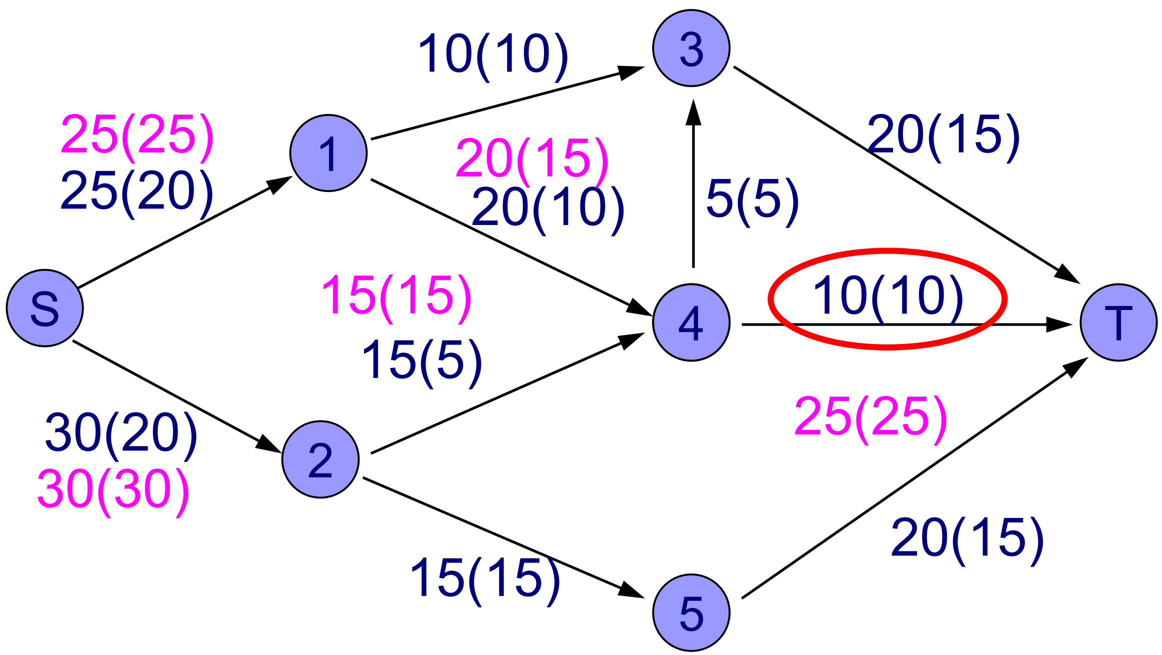
如果某未标号点k有两个以上相邻的标号点，可按(1)(2)分别计算出δ_k值，并取其中最大一个标记。

现在最大流为多少？

- 40
- 看网络的最小割

4.4 求网络最大流的标号算法

2. 为了增加从S到T的流量，同时又希望尽可能少地改变网络中的管道，应当改变哪些管道的容量？ 即只改变一条弧的容量。

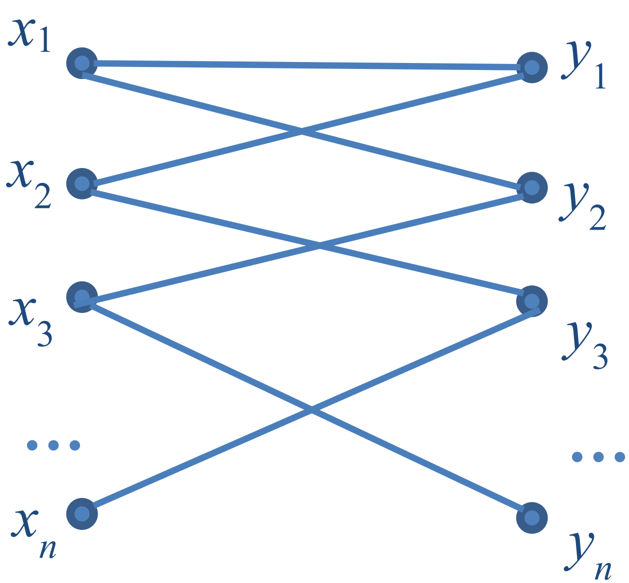


- 先看流到T的三条弧
- 找满弧

现在最大流为多少？

55

工作分配问题： 有 n 个工人，m 件工作，每个人能力不同，各能胜任其中某几项工作。假设每件工作只需一个人做，每人只做一件工作，怎样分配才能使尽量多的工作有人做，更多的人有工作？

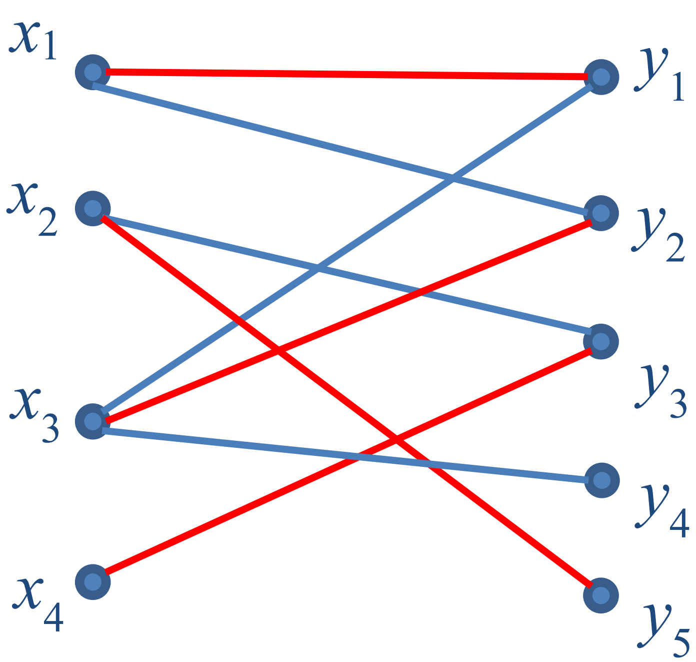


二部图 $G=(X,Y,E)$

在图G中找一个边集E的子集，使得集中任何两条边没有公共端点，最好的方案就是要使此边集的边数尽可能多。

↑

最大匹配



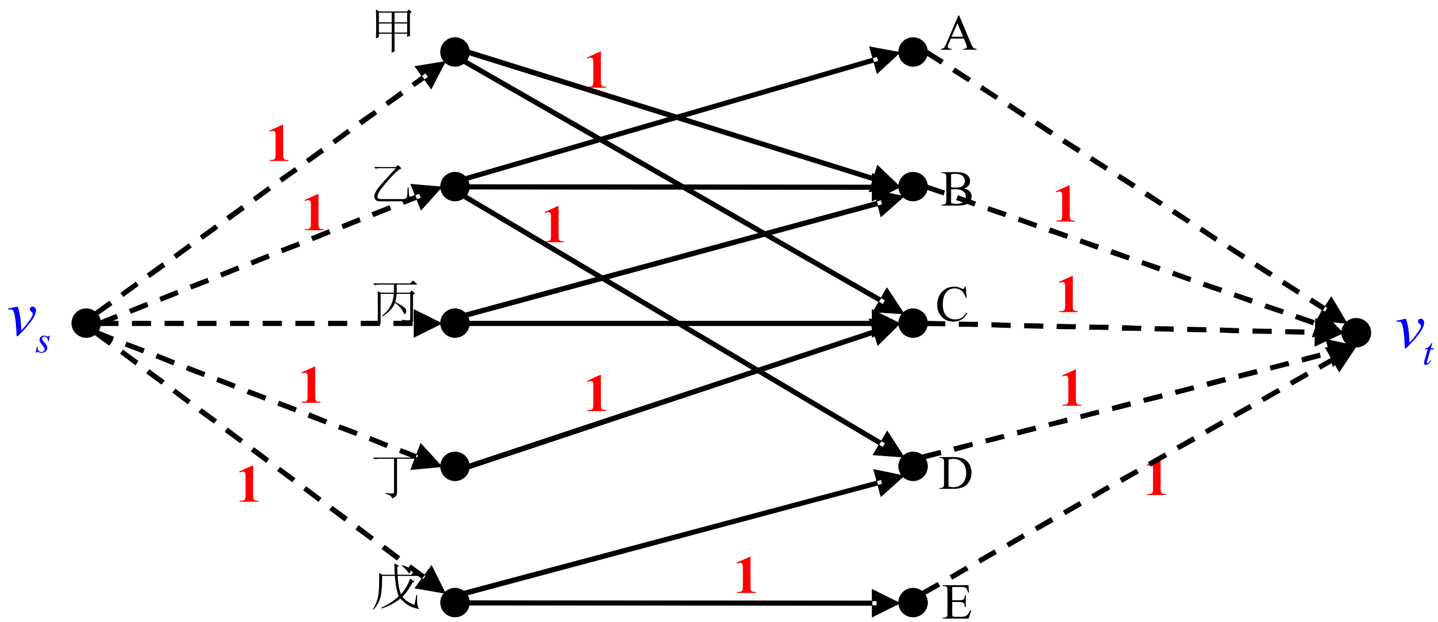
一个图的最大匹配中所含边数是确定的，但匹配方案可以不同！

$$M = \{(x_1, y_1), (x_2, y_5), (x_3, y_2), (x_4, y_3)\}$$

$$M' = \{(x_1, y_1), (x_2, y_5), (x_3, y_4), (x_4, y_3)\}$$

4.5 最大匹配问题

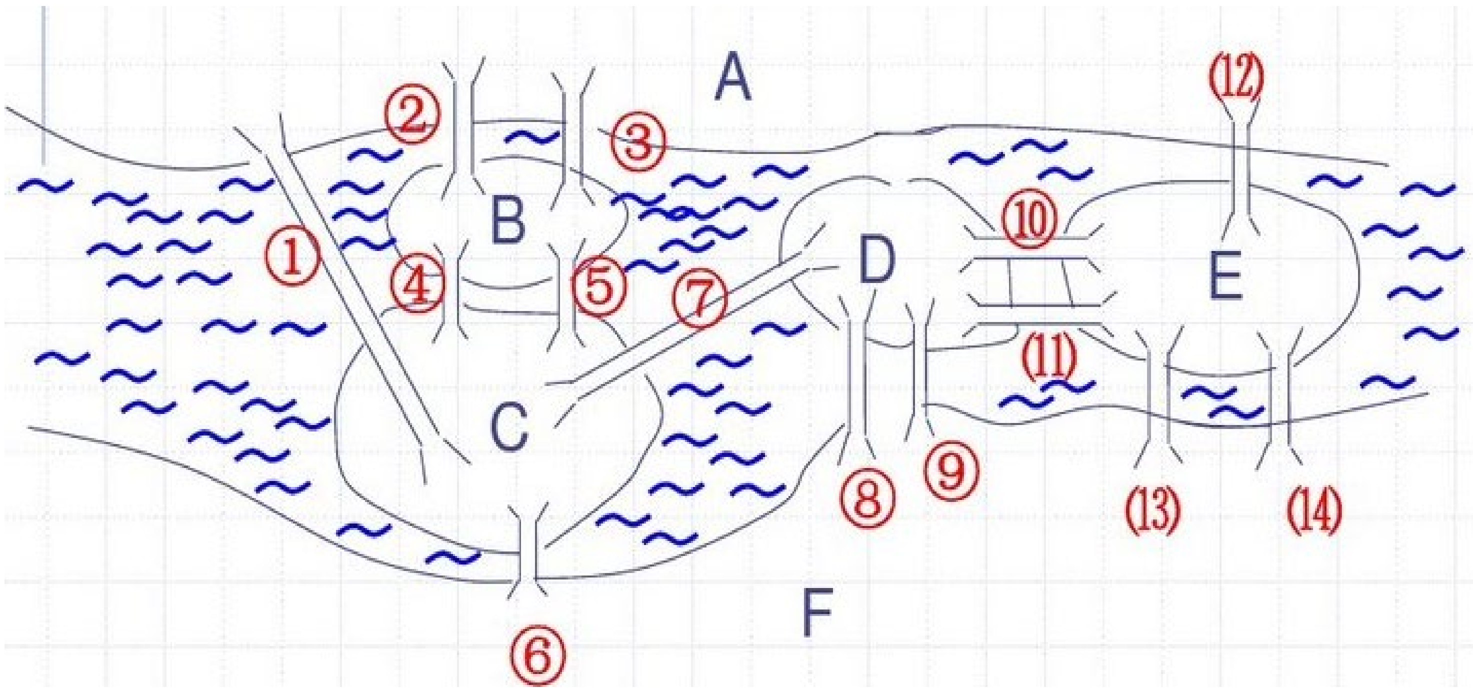
二部图中最大匹配问题，可以化为最大流问题求解。



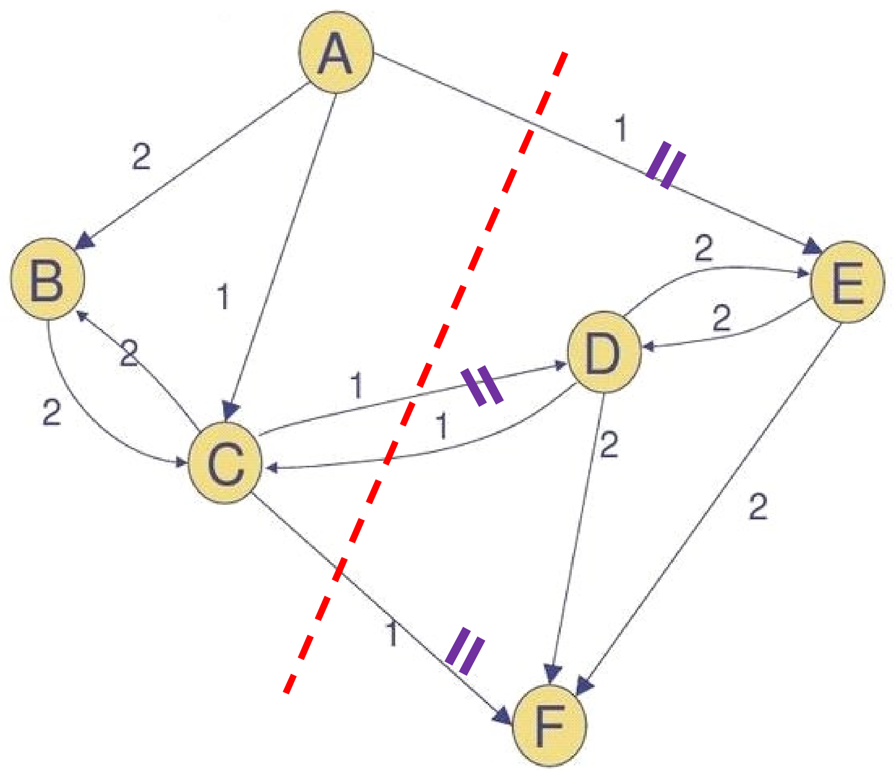
令全部边上的容量均为 1，那么当这个网络的流达到最大时，就得到最大匹配方案。

4.6 其它问题

图中A、B、C、D、E、F分别表示陆地和岛屿，1，2，…，14表示桥梁及其编号。河两岸分别互为敌对的双方部队占领，问至少应切几座桥梁（具体指出编号）才能达到阻止对方部队过河的目的。试用图论方法进行分析。



南京大学 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING



最小割为 $\{AE, CD, CF\}$

4.6 其它问题

在美国职业棒球例行赛中，每个球队都要打162场比赛（对手包括但不限于同一分区里的其他队伍，和同一支队伍也往往会有多次交手），所胜场数最多者为该分区的冠军；如果出现并列第一，则用加赛决出冠军。在比赛过程中，如果发现某支球队无论如何都已经不可能以第一名或者并列第一名的成绩结束比赛，那么这支球队就提前被淘汰了（虽然它还要继续打下去）。

4.6 其它问题

Team	胜	负	剩余	纽约	巴尔的摩	波士顿	多伦多	底特律
纽约	75	59	28	0	3	8	7	3
巴尔的摩	72	62	28	3	0	2	7	4
波士顿	69	66	27	8	2	0	0	0
多伦多	60	75	27	7	7	0	0	0
底特律	49	86	27	3	4	0	0	0

该表是某次美国联盟东区比赛的结果。在该小组分区中，纽约队暂时排名第一，总共胜 75 场，负 59 场，剩余 28 场比赛没打，其中和巴尔的摩还有 3 场比赛，和波士顿还有 8 场比赛，和多伦多还有 7 场比赛，和底特律还有 3 场比赛（还有 7 场与不在此分区的其他队伍的比赛）。

4.6 其它问题

Team	胜	负	剩余	纽约	巴尔的摩	波士顿	多伦多	底特律
纽约	75	59	28	0	3	8	7	3
巴尔的摩	72	62	28	3	0	2	7	4
波士顿	69	66	27	8	2	0	0	0
多伦多	60	75	27	7	7	0	0	0
底特律	49	86	27	3	4	0	0	0

底特律暂时只有 49 场比赛获胜，剩余 27 场比赛没打。如果剩余的 27 场比赛全都获胜的话，是有希望超过纽约队的；即使只有其中 26 场比赛获胜，也有希望与纽约队战平，并在加赛中取胜。然而，根据表里的信息已经足以判断，其实底特律已经没有希望夺冠了，请你不妨来分析推导一下。

第七章 图与网络

5. 最小费用流问题

5.1 最小费用流—模型

- 给定网络$G=(V,A,C,d)$和经过网络的流量$W(f)=v$，求流在网络上的最佳分布，使总费用最小。

$$\min \quad d(f)=\sum_{(i,j)\in A} d_{ij}f_{ij}$$

$$\sum_{(s,j)\in A} f_{sj}-\sum_{(j,s)\in A} f_{js}=v$$

$$\sum_{(t,j)\in A} f_{tj}-\sum_{(j,t)\in A} f_{jt}=-v$$

$$\sum_{(i,j)\in A} f_{ij}-\sum_{(j,i)\in A} f_{ji}=0, i\neq s,t$$

$$0\leq f_{ij}\leq C_{ij}$$

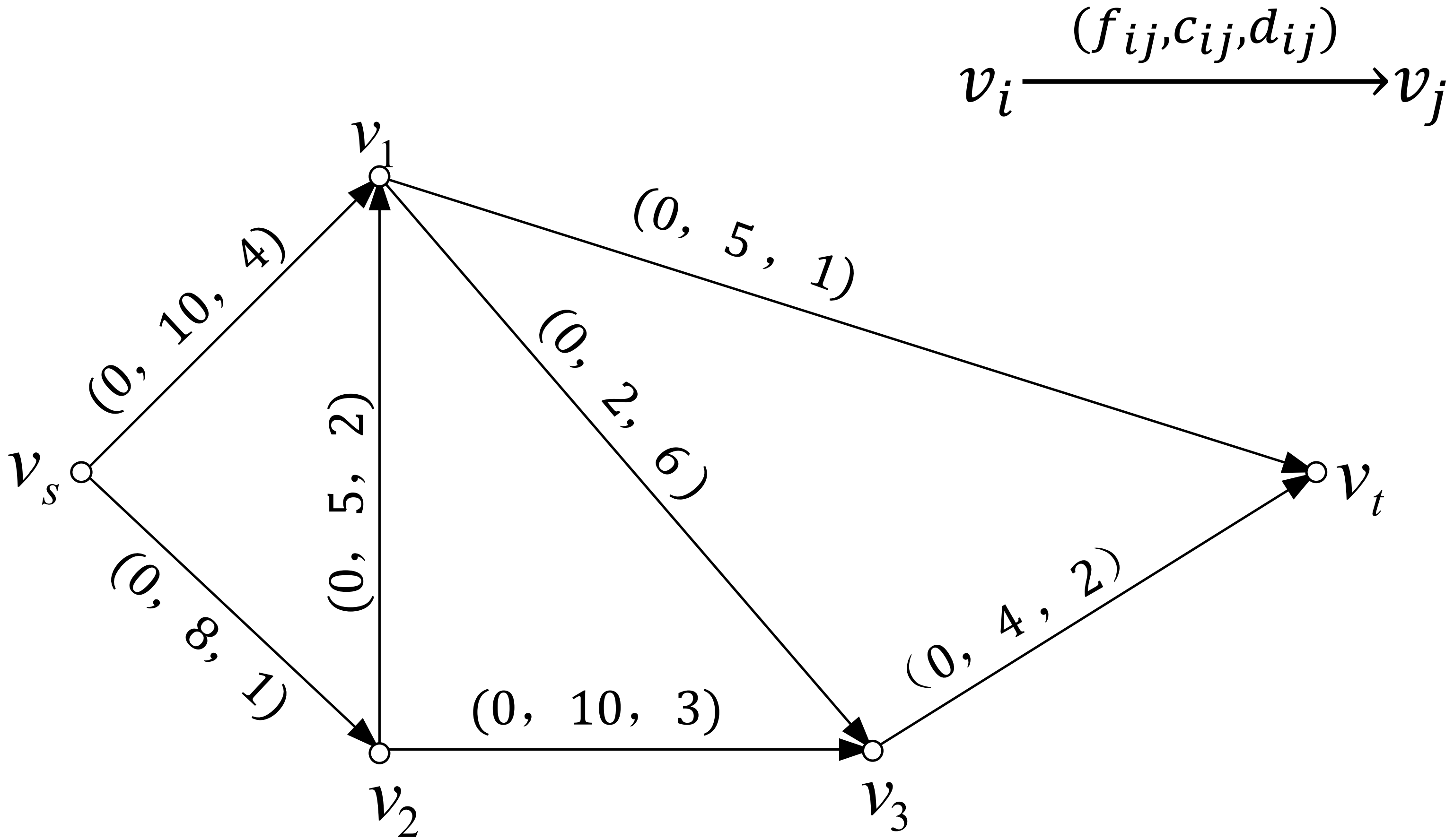
5.2 最小费用增广链

- 增广链费用，最小费用增广链。
- 对于最小费用可行流，沿最小费用增广链调整流，可使流增加，并保持流费用最小。
- 给定初始最小费用可行流，求最小费用增广链，若存在，则沿该增广链调整网络流，直到达到给定的网络流或不存在增广链为止，后一种情况为最小费用最大流。
- 若给定网络流超过最大流，则不可能实现。

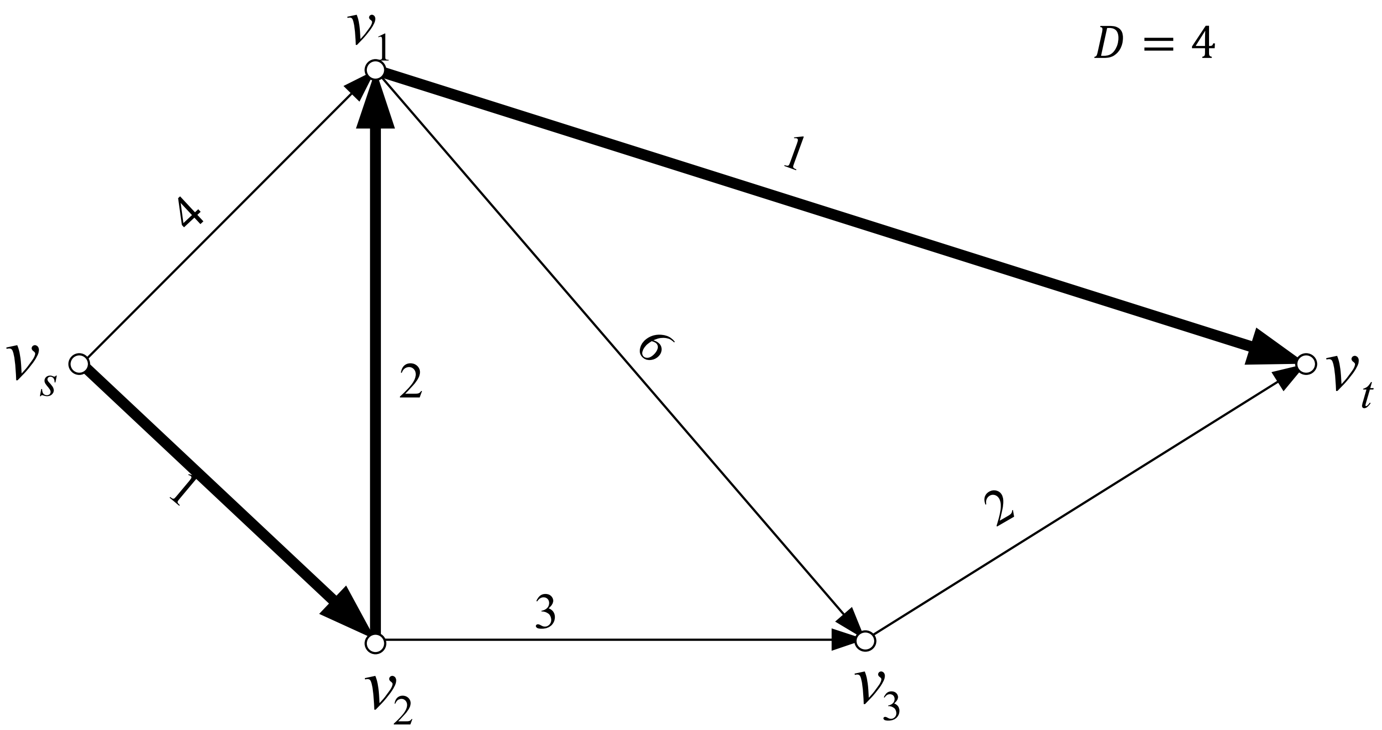
5.2 最小费用增广链

- 生成最小费用可行流的剩余网络：
  - 将饱和弧反向
  - 将非饱和非零流弧加一反向弧
  - 零流弧不变
  - 所有前向弧的权为该弧的费用，后向弧的权为该弧费用的相反数
- 剩余网络又叫长度网络
- 最小费用增广链对应剩余网络的最短路

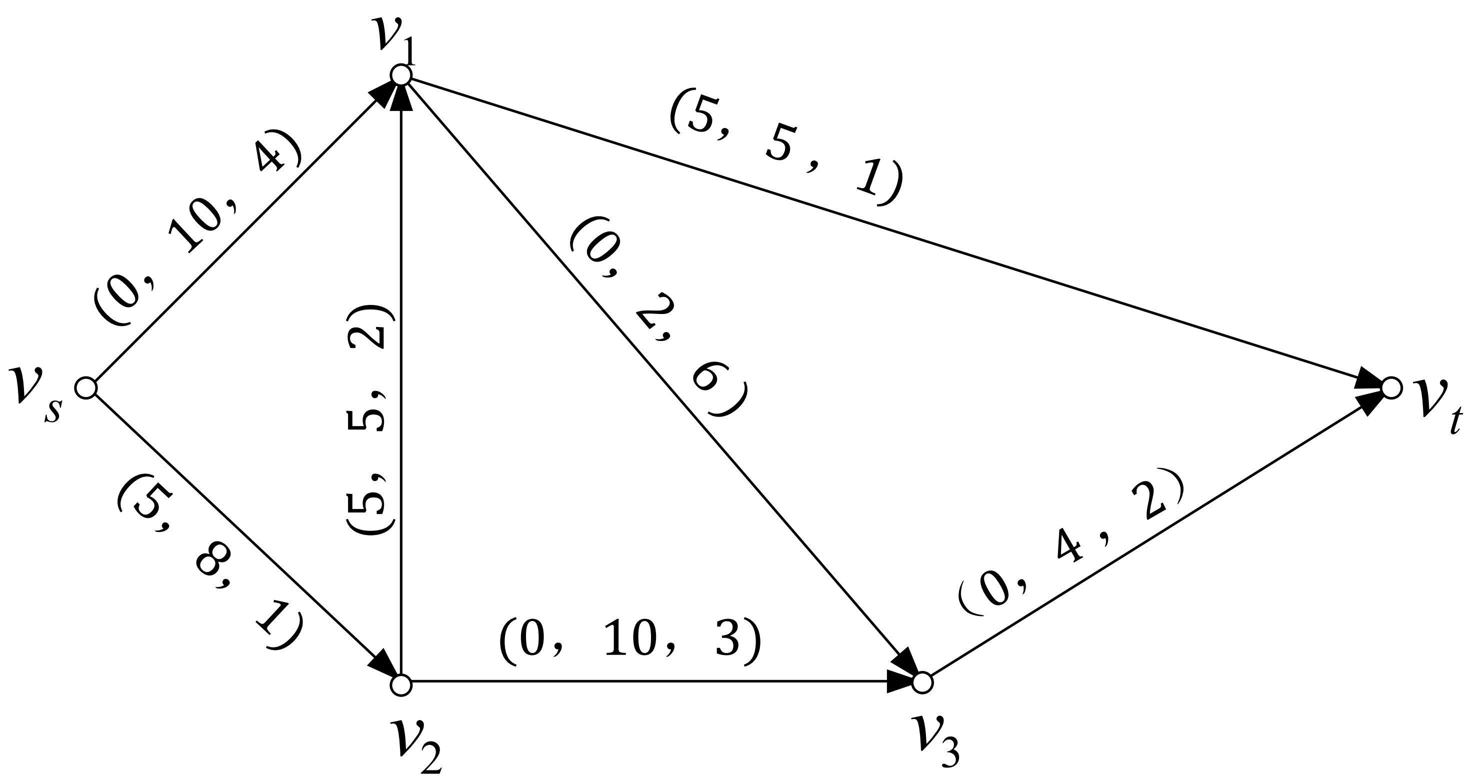
5.3 最小费用流—例题


$v_i \xrightarrow{(f_{ij}, c_{ij}, d_{ij})} v_j$
v_1
v_s
v_2
v_3
v_t
(0, 10, 4)
(0, 5, 1)
(0, 2, 6)
(0, 5, 2)
(0, 8, 1)
(0, 10, 3)
(0, 4, 2)

第1次剩余网络最短路

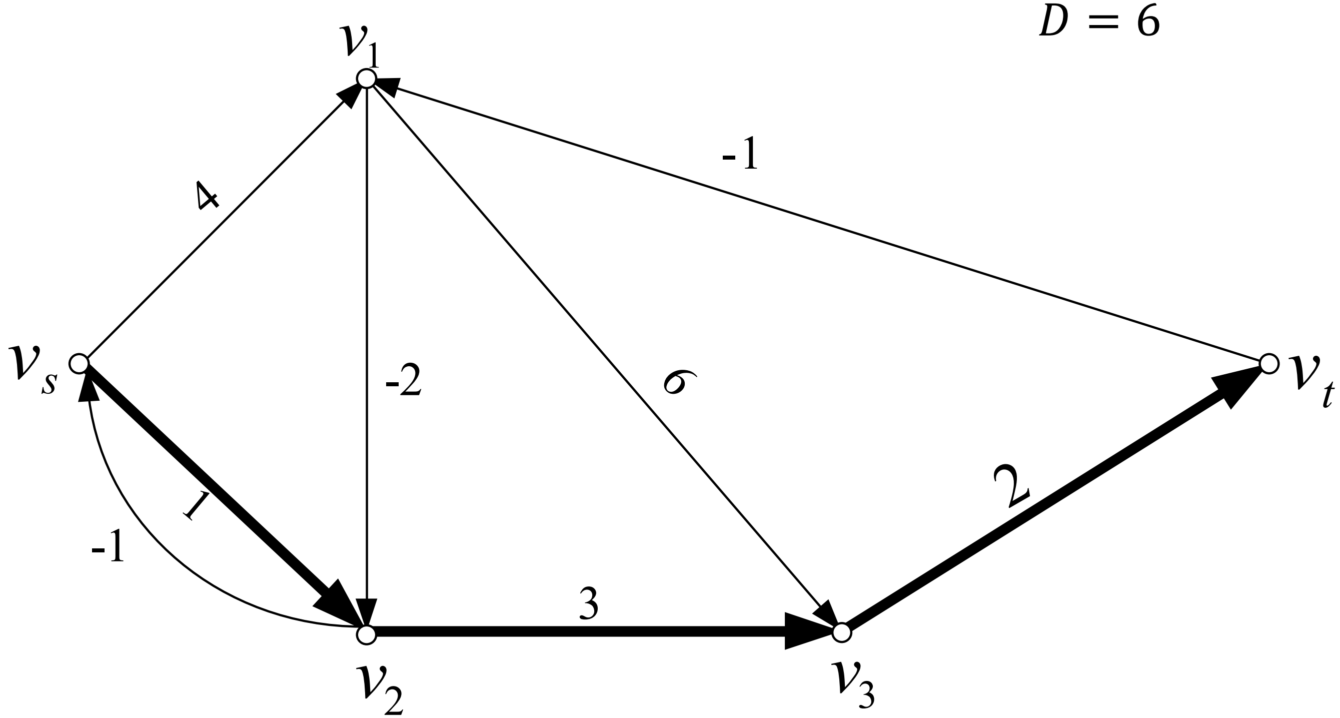


第1次调整网络流



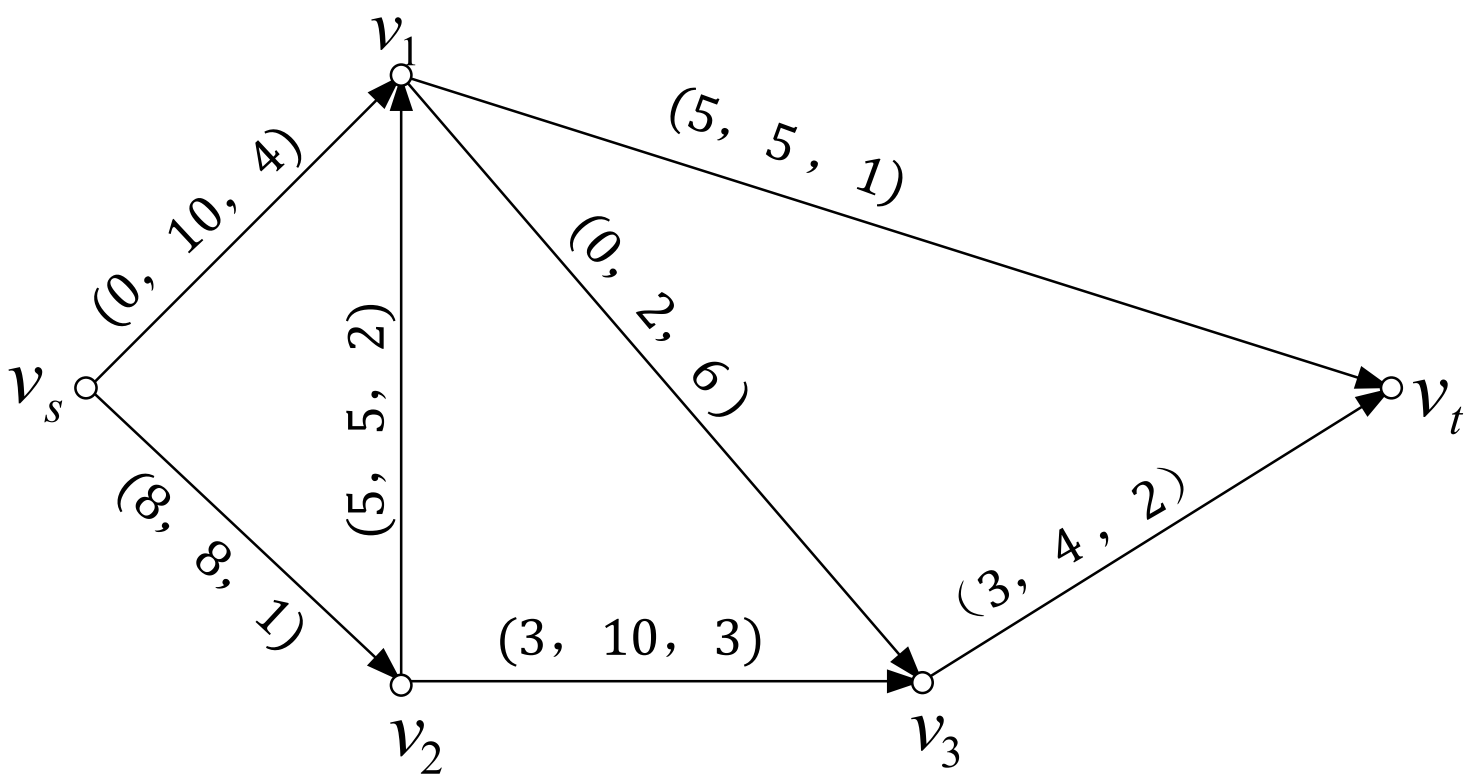
第2次剩余网络最短路

$D = 6$



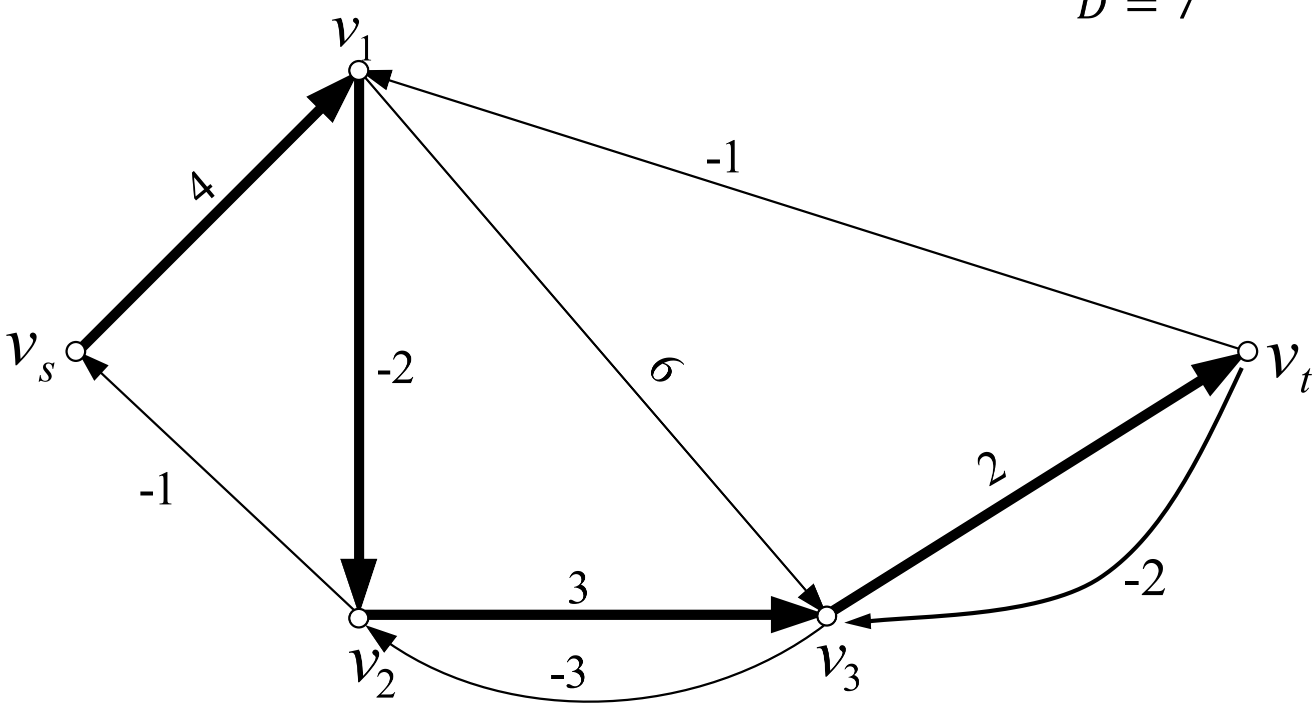
5.3 最小费用流—例题

第2次调整网络流

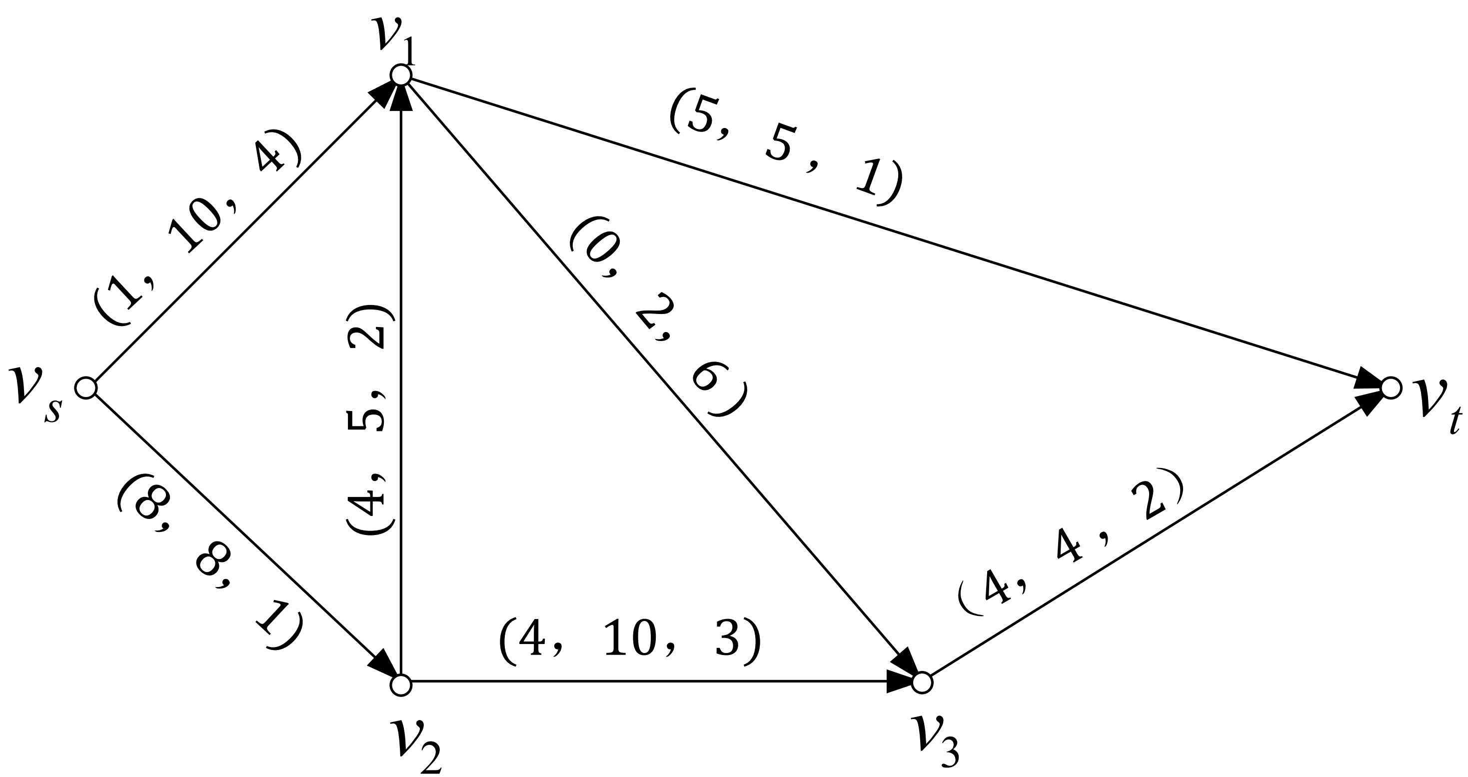


第3次剩余网络最短路

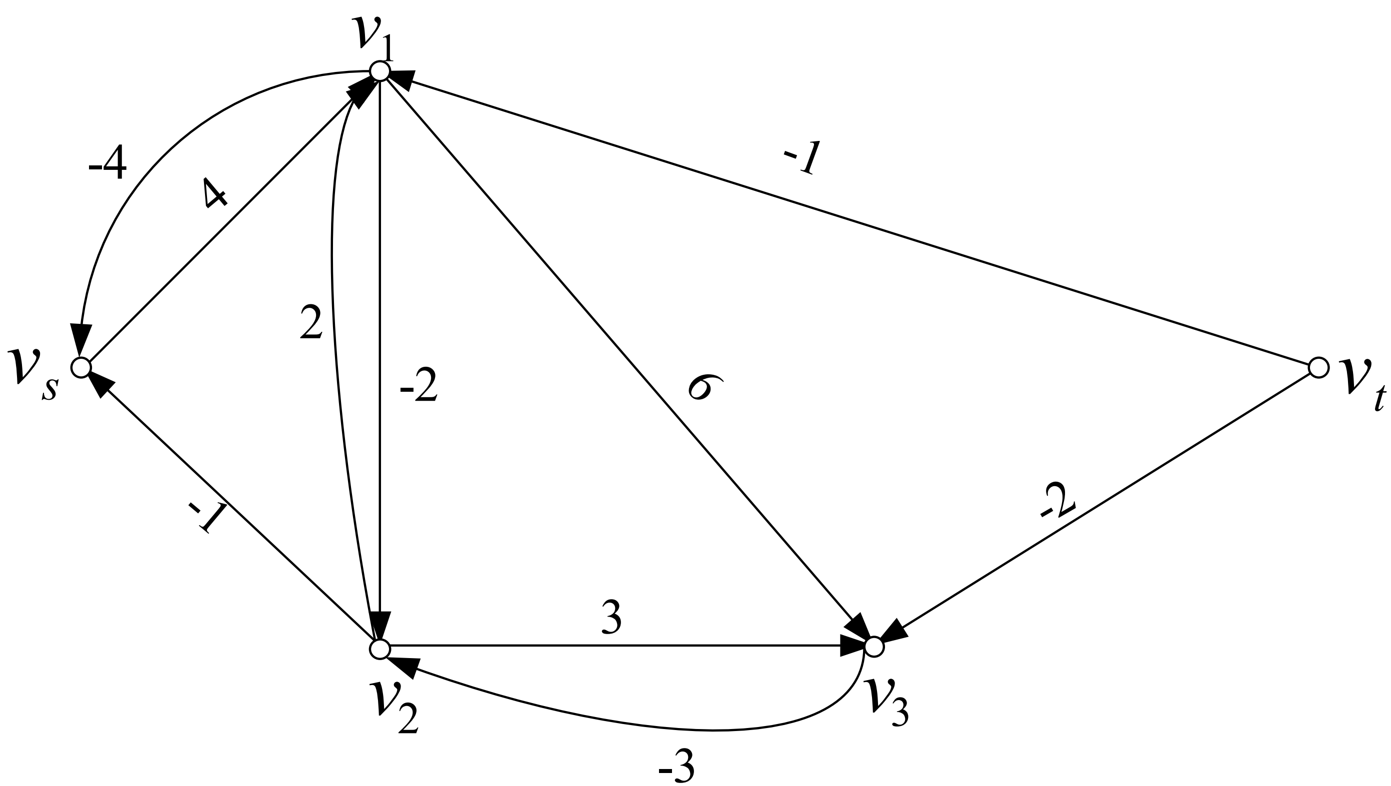
$D = 7$



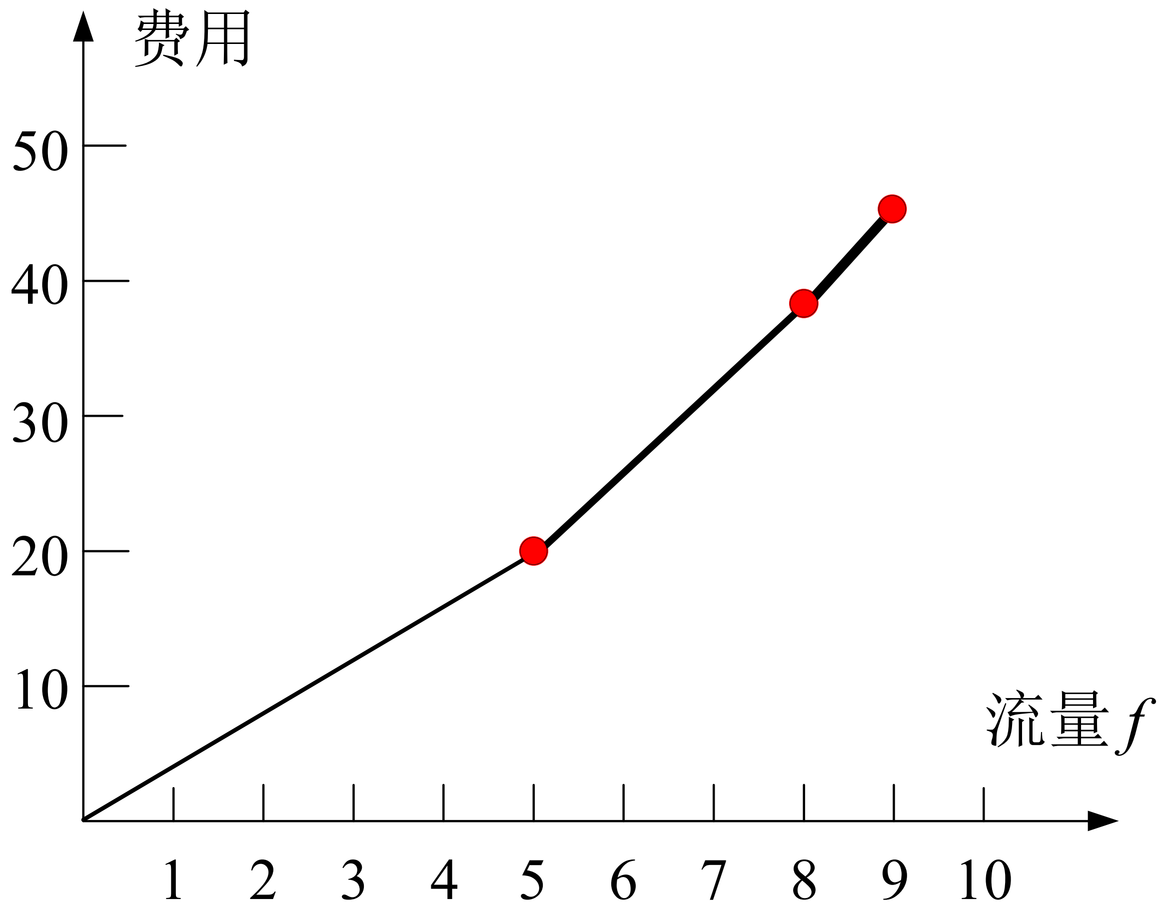
第3次调整网络流



剩余网络已不存在最短路



5.3 最小费用流—例题



一般形式的最小费用流问题

$$\min \quad c(\mathbf{x}) = \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = d_i, \forall i \in V \quad \begin{cases} d_i > 0, \text{ supply node/source} \\ d_i < 0, \text{ demand node/sink} \\ d_i = 0, \text{ transshipment node} \end{cases}$$

$$l_{ij} \le x_{ij} \le u_{ij} \longrightarrow 0 \le x_{ij} \le u_{ij}$$

— 容量受限的转运问题

— 对于单源单汇的情形，寻找从 s 流到 t 的给定流量的最小费用流，是经典的最小费用流问题

令 $d_s = v,\ d_t = -v,\ d_i = 0$(当$i \neq s, t$) 即可。

最小费用流问题的特例

- **最短路问题**

令所有弧的容量下界为 0，容量上界为 1。图中某个节点 s 的供需量为 1，某个节点 t 的供需量为 -1。再令所有弧的费用为"弧长"，则此时的最小费用流问题就是最短路问题。

- **运输问题**

运输问题是研究比较早的最小费用流问题之一，早在1941年Hitchcock就进行了研究，因此运输问题又称为Hitchcock问题。

- **最大流问题**

设s为起点，t为终点，增加弧(t, s)，令$c_{ts} = -1$, $u_{ts} = +\infty$，而令所有其他弧上的费用为 0，所有节点上的供需量全为 0。