

运筹学中把一些研究对象用节点表示，对象之间的关系用连线边表示。用点、边的集合构成图。图论是研究有节点和边所组成图形的数学理论和方法。图是网络分析的基础，根据具体研究的网络对象（如：铁路网、电力网、通信网等），赋予图中各边某个具体的参数，如时间、流量、费用、距离等，规定图中各节点代表具体网络中任何一种流动的起点，中转点或终点，然后利用图论方法来研究各类网络结构和流量的优化分析。

经典的图与网络分析往往可以转化为经典的线性规划问题，也可以反过来。

如下例

产地 \ 销地	B_1	B_2	B_3	B_4	产量
A_1	x_{11} 8	x_{12} 6	x_{13} 10	x_{14} 9	35
A_2	x_{21} 9	x_{22} 12	x_{23} 13	x_{24} 7	50
A_3	x_{31} 14	x_{32} 9	x_{33} 16	x_{34} 5	40
销量	45	20	30	30	

数学规划模型为

$$\begin{aligned} \min \sum_{i=1}^3 \sum_{j=1}^4 c_{ij} x_{ij} &= 8x_{11} + 6x_{12} + 10x_{13} + 9x_{14} + 9x_{21} + 12x_{22} + 13x_{23} \\ &\quad + 7x_{24} + 14x_{31} + 9x_{32} + 16x_{33} + 5x_{34} \\ \text{s.t. } x_{11} + x_{12} + x_{13} + x_{14} &= 35 \\ x_{21} + x_{22} + x_{23} + x_{24} &= 50 \\ x_{31} + x_{32} + x_{33} + x_{34} &= 40 \\ x_{11} + x_{21} + x_{31} &= 45 \\ x_{12} + x_{22} + x_{32} &= 20 \\ x_{13} + x_{23} + x_{33} &= 30 \\ x_{14} + x_{24} + x_{34} &= 30 \quad x_{ij} \geq 0, \forall 1 \leq i \leq 3, 1 \leq j \leq 4 \end{aligned}$$

求解如下线性规划问题：

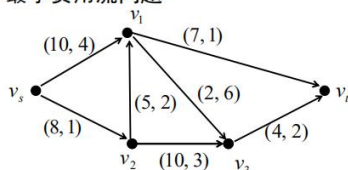
$$\begin{aligned} \min C^T X \\ \text{s.t. } AX = B, 0 \leq X \leq D \end{aligned}$$

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ -10 \end{bmatrix}$$

$$C^T = [4 \ 1 \ 2 \ 6 \ 3 \ 1 \ 2]$$

$$D^T = [10 \ 8 \ 5 \ 2 \ 10 \ 7 \ 4]$$

等价于如下最小费用流问题



括号内第一个数字是容量，第二个是单位流量费用

目标：从发点到收点的总的流量费用最小

约束：1) 容量约束，各边流量不大于容量

2) 流量平衡约束，各点进出流量总和相等

3) 从发点到收点的总流量为10

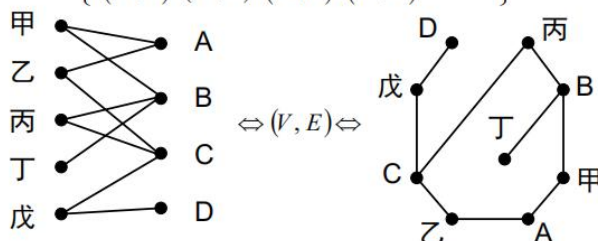
下面引入图论中的若干概念：

图的定义

点集 $V \rightarrow$ 边集 $E \rightarrow$ 图 $G=(V, E)$

例 $V = \{ \text{甲, 乙, 丙, 丁, 戊, A, B, C, D} \}$

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

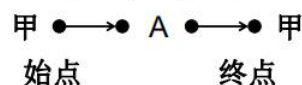


无向图

$$(\text{甲}, A) = (A, \text{甲})$$

有向图

$$(\text{甲}, A) \neq (A, \text{甲})$$



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

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

自回路 两端点相同的边，或称为环

多重边 两点之间多于一条不同的边

简单图与多重图

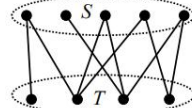
简单图 不含自回路和多重边的图

多重图 含有多重边的图



完全图 任意两个顶点间都有边相连的无向简单图称为完全图，有 n 个顶点的无向完全图记为 K_n ，任意两个顶点间都有且仅有一条边相连的有向简单图称为有向完全图

二分图 如果 V 可以划分为两个不相交的子集 S, T ，使得 E 中每条边的两个端点必有一个属于 S ，一个属于 T ，称 $G=(V, E)$ 为二分图记为 $G=(S, T, E)$



端点的次 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

出次与入次

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

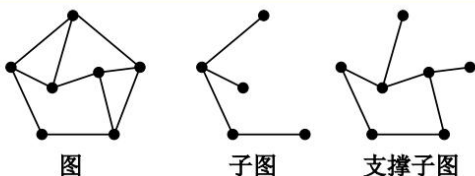
定理1 任何图中，顶点次数总和等于边数的 2 倍

定理2 任何图中，奇点的个数为偶数个

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

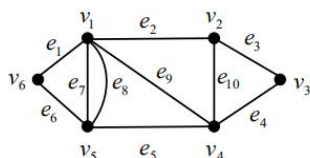
联，则称 $G'=(V',E')$ 是 G 的子图，特别是，若 $V'=V$ ，则称 G' 为 G 的支撑子图



链（路） 点、边交替（可重复）的序列，如

$$S = \{v_6, e_6, v_5, e_7, v_1, e_8, v_5, e_7, v_1, e_9, v_4, e_4, v_3\}$$

初等链 无重复点边的链 $S_1 = \{v_6, e_6, v_5, e_5, v_4, e_4, v_3\}$



圈 始点和终点为同一点的链（初等链）

道路（回路） 有向图中各边方向相同的链（圈）

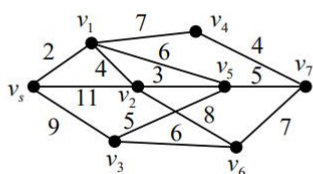
连通图 任何两点间有链相连的图

当我们往有向图或者无向图中的点或边添加权重时，图也就变成了网络。

网络（赋权图）

点或边带有数值（权）的图称为网络

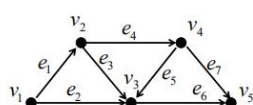
例 某物资供应站 v_s 与用户之间的公路网络



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**

为了方便实际计算和编程处理，一般的图可以用关联矩阵或者邻接矩阵来表示。

有向图的关联矩阵



将有向图关联矩阵的-1换成1就得到无向图关联矩阵

图的邻接矩阵

两点间有边为1，否则为0

$$A = \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ v_1 & 1 & 1 & & & & \\ v_2 & -1 & & 1 & 1 & & \\ v_3 & & -1 & -1 & & -1 & 1 \\ v_4 & & & -1 & 1 & & 1 \\ v_5 & & & & & -1 & -1 \end{pmatrix}$$

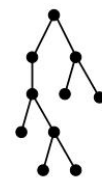
$$B = \begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 \\ v_1 & 0 & 1 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 0 \\ v_3 & 1 & 1 & 0 & 1 & 1 \\ v_4 & 0 & 1 & 1 & 0 & 1 \\ v_5 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

图的最小支撑树在电信网络的设计、低负荷运输网络的设计（如铁路、公路等）、总成本最小高压输电线路网络的设计、管道网络设计中有很多应用。

所谓的树，是指**连通且不含圈的无向图**（森林：无圈的图）

定理 1

设 $T=(V,E)$ ， $n=|V| \geq 3$ ， $m=|E|$ ，则以下等价：



- 1) T 是一个树
- 2) T 无圈，且 $m=n-1$
- 3) T 连通，且 $m=n-1$
- 4) T 无圈，但每加一新边（不加点）即得唯一一个圈
- 5) T 连通，但舍去任一边就不连通
- 6) T 中任意两点，有唯一链相连

定理 2 每个树至少有两个次为 1 的点。

若 $T=(V,E)$ 恰好有两个次为 1 的点，则其它点的次必为 2，因此 $T=(V,E)$ 是一条链（路）

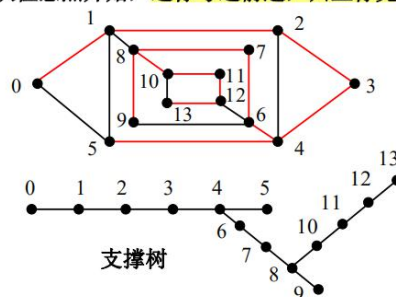
而则称其为 G 的支撑树，如果 $G=(V,E)$ 的支撑子图 $G'=(V',E')$ 是树， G 中属于支撑树的边称为**树枝**，不属于支撑树的边称为**弦**。支撑树的存在性有如下定理：

定理 3 图 $G=(V,E)$ 有支撑树的充要条件是 G 是**连通图**

确定支撑树的方法主要有深探法和广探法两种：

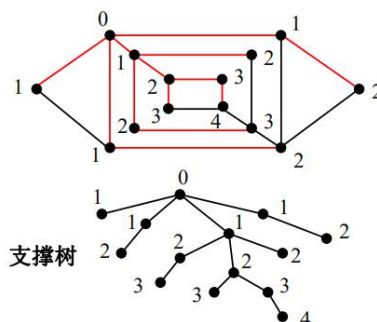
1) 深探法

从任意点开始，边标号边前进，只至标完所有点



2) 广探法

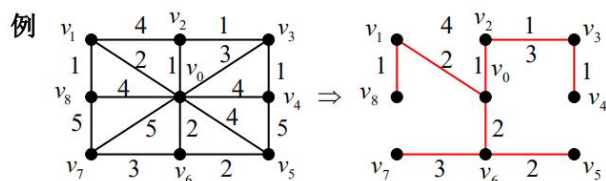
从任意点开始，把当前标号点附近标号完再前进



支撑树所有树枝上的权的总和称为这个支撑树的权（长）具有最小权的支撑树称为最小支撑树（最小树）。

求解最小支撑树主要有 Kruskal 算法（避圈法）、Dijkstra 算法、Prim 算法等。

Kruskal 算法是一种贪心算法，它将所有边按权值从小到大排序，从权值最小的边开始选树枝，如果可能形成圈则跳过，直到选够顶点数减 1 的树枝。



所有边从小到大排列

$$\begin{aligned} (v_0, v_2) &= 1, (v_2, v_3) = 1, (v_3, v_4) = 1, (v_1, v_8) = 1, (v_0, v_1) = 2 \\ (v_0, v_6) &= 2, (v_5, v_6) = 2, (v_0, v_3) = 3, (v_6, v_7) = 3, (v_0, v_4) = 4 \\ (v_0, v_5) &= 4, (v_0, v_8) = 4, (v_1, v_2) = 4 \end{aligned}$$

从小到大顺序选择不构成圈的边形成右上支撑树

性质：加入任何弦形成的圈中，弦的权值最大

Kruskal 算法的有效性由下面的定理来保证：

定理 T 是最小支撑树的充要条件是：加入任何弦形成的圈中，弦的权值最大

\Rightarrow Kruskal 算法产生的是最小支撑树

证明必要性：

如果加入某个弦形成的圈中有比该弦的权值更大的树枝，则用该弦代替最大数值形成的支撑树的总权值会变小，和最小支撑树定义矛盾

证明充分性：

设 T_1 是满足条件的支撑树， T_2 是所有最小支撑树中和 T_1 不同的树枝数最少的树（一定存在），记

$$T_1 = \{e_1, e_2, \dots, e_m, \hat{T}\}_2, \quad T_2 = \{\bar{e}_1, \bar{e}_2, \dots, \bar{e}_m, \hat{T}\}$$

其中 $w(\bar{e}_i) \leq w(e_i), \forall 2 \leq i \leq m$

将 \bar{e}_1 加入 T_1 会形成回路，一定有 $e_k \in T_1 \setminus \hat{T} \Rightarrow w(e_k) \leq w(\bar{e}_1)$

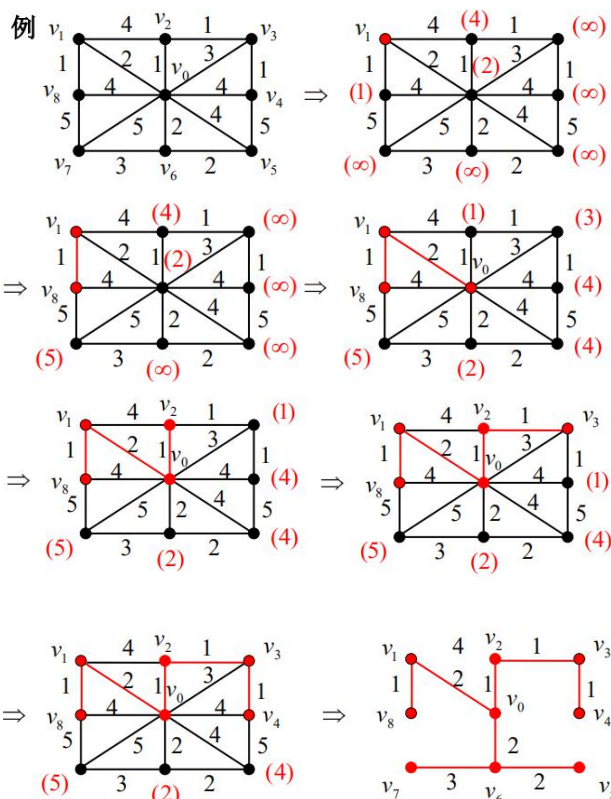
将 e_k 加入 T_2 会形成回路，一定有 $\bar{e}_j \in T_2 \setminus \hat{T}$

$w(e_k) \leq w(\bar{e}_1) \leq w(\bar{e}_j) \Rightarrow T_2(e_k \setminus \bar{e}_j)$ 仍是最小支撑树

$T_2(e_k \setminus \bar{e}_j)$ 和 T_1 的不同树枝数为 $m-1$ ，矛盾！ $\Rightarrow m=0$

Dijkstra 算法从任意点开始逐渐增加某个点集，记为 S ，每次从不在 S 的点集里选择距 S 一步距离最小的点加入 S ，将相应边取为树枝，直至 S 包含所有的点。

求最小支撑树的 Dijkstra 算法（1959年提出， $O(n^2)$ ）



在判断一个数是否已经是最小支撑树的充要条件是：任何树枝都是所在唯一的割集中权值最小的边。

必要性：如果不是，用权值最小的边代替相应树枝可得总权值更小的支撑树

充分性：加入任何弦形成的回路中，弦和回路上任何树枝都在某个唯一的割集上，所以弦的权值最大

\Rightarrow Dijkstra 算法产生的是最小支撑树

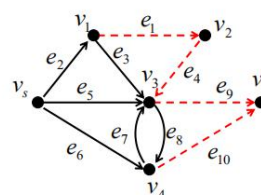
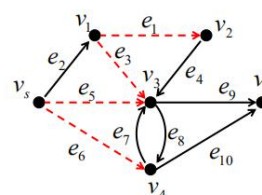
算法复杂性：Kruskal $n^2 \log_2 n > n^2$ Dijkstra

其中割集的概念：

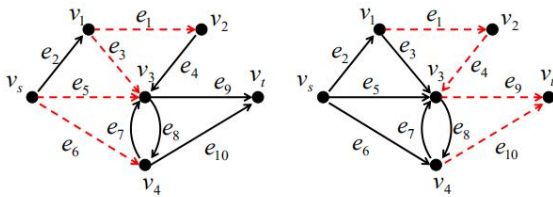
边割 对于图 $G=(V, E)$ ，任取 $S \subset V$ ，其补集为 \bar{S} ，若 S 和 \bar{S} 都不是空集，称两个端点分属 S 和 \bar{S} 的边的集合为 G 的一个边割，记为 $\{S, \bar{S}\}$

$$\{e_1, e_3, e_5, e_6\}$$

$$\{e_1, e_4, e_9, e_{10}\}$$



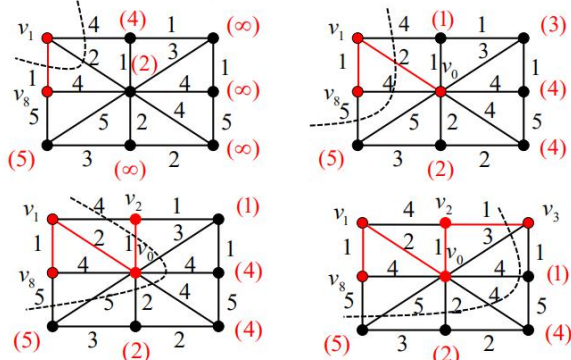
割集 当除去边割后，连通图变为不连通，而除去边割的真子集后，连通图仍然连通



$\{e_1, e_3, e_5, e_6\}$ 是割集, $\{e_1, e_4, e_9, e_{10}\}$ 是边割, 不是割集

当我们用边割的角度去分析 Dijkstra 算法的每一步时可以看到每一步的新增边是虚线所指出的割集中的最短的边:

求最小支撑树的 Dijkstra 算法的性质



新增边是虚线所指出的割集中的最短的边

求最短路问题的 Dijkstra 算法的一般流程如下
一般性问题:

连通图 $G=(V, E)$ 各边 (v_i, v_j) 有权 l_{ij} ($l_{ij} = \infty$ 表示两点间无边), 任意给定两点 v_s, v_t , 求一条道路 μ , 使它是从 v_s 到 v_t 的所有道路中总权最小的道路, 即

$$L(\mu_s) = \min_{\mu \in \Omega_{st}} L(\mu) = \sum_{(v_i, v_j) \in \mu} l_{ij}$$

其中 Ω_{st} 表示从 v_s 到 v_t 的所有道路的集合

适用 Dijkstra 算法的问题: 所有权值非负

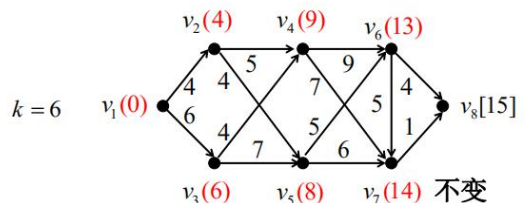
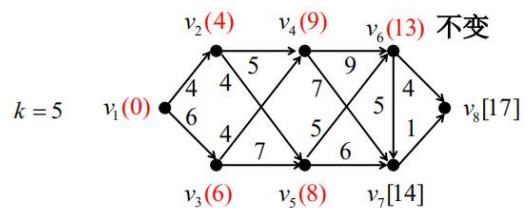
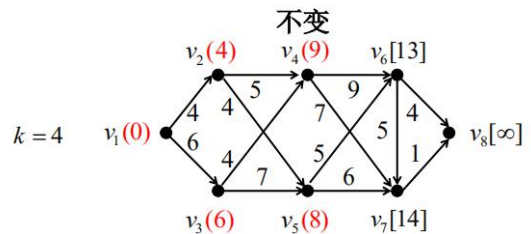
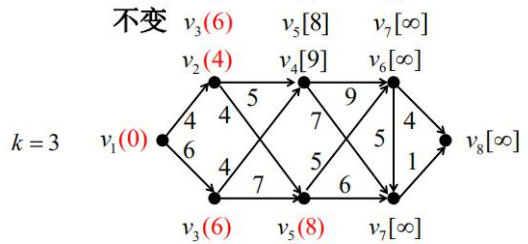
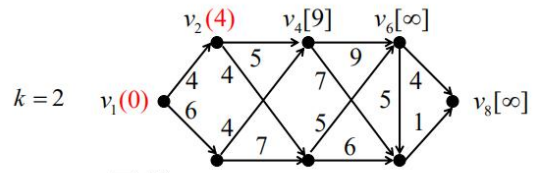
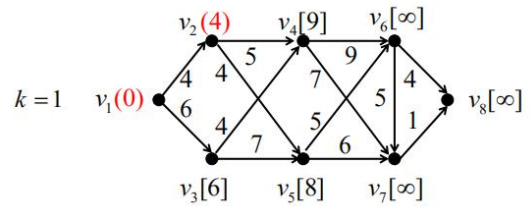
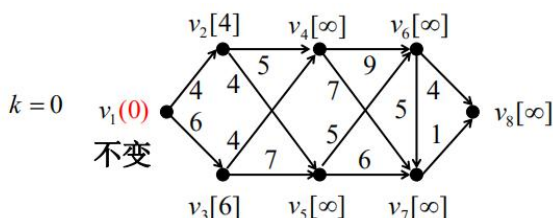
值迭代公式 (顺推)

$$f_1(v_j) = l_{1j}, \forall j$$

$$f_{k+1}(v_j) = \min_i \{f_k(v_i) + l_{ij}\}, \forall j$$

其中 $f_k(v_j)$ 表示经过 $k-1$ 个中间点到达 v_j 的最短路

用 Dijkstra 算法求下面的最短路问题:



最大流问题 (maximum flow problem), 一种组合最优化问题, 就是要讨论如何充分利用装置的能力, 使得运输的流量最大, 以取得最好的效果。

容量网络

有向连通图 $G=(V, E)$ 各边 (v_i, v_j) 有非负容量 c_{ij} , 仅有一个入次为 0 的点 v_s , 称为发 (源) 点, 一个出次为 0 的点 v_t , 称为收 (汇) 点, 常记为 $G=(V, E, C)$, 其中 $C=\{c_{ij}\}$

可行流 满足流量平衡约束和容量约束的 $X=\{x_{ij}\}$

$$\sum_{(v_i, v_j) \in E} x_{ij} = \sum_{(v_k, v_i) \in E} x_{ki}, \forall v_i \in V, i \neq s, t \quad (\text{流量平衡约束})$$

$$0 \leq x_{ij} \leq c_{ij}, \forall (v_i, v_j) \in E \quad (\text{容量约束})$$

$$\text{可行流的网络总流量 } W = \sum_{(v_i, v_j) \in E} x_{ij} = \sum_{(v_k, v_t) \in E} x_{kt}$$

用矩阵表示图：

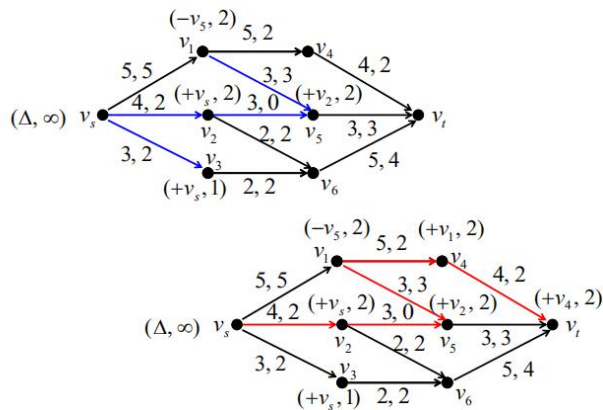
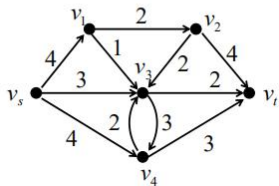
$$\max W$$

$$\text{s.t. } \sum_{(v_i, v_j) \in E} x_{ij} - \sum_{(v_j, v_i) \in E} x_{ji} = \begin{cases} W & \text{if } i = s \\ 0 & \text{if } i \neq s, t \\ -W & \text{if } i = t \end{cases}$$

$$0 \leq x_{ij} \leq c_{ij}, \forall (v_i, v_j) \in E$$

$$\text{s.t. } AX = B$$

$$0 \leq X \leq C$$



解决最大流问题的有效方法需要运用所谓的“可增广链”：

可增广链

设 μ 是从 v_s 到 v_t 的一条链，定义 μ 的方向为从 v_s 到 v_t 的方向，对于 μ 上的任意边，如果其方向和 μ 相同则称其为**前向边**，否则为**后向边**，用 μ^+ 和 μ^- 分别表示前向边和后向边的集合，如果 $X = \{x_{ij}\}$ 是一个可行流，且满足

$$\begin{aligned} 0 \leq x_{ij} < c_{ij} & \quad \forall (v_i, v_j) \in \mu^+ \\ 0 < x_{ij} \leq c_{ij} & \quad \forall (v_i, v_j) \in \mu^- \end{aligned}$$

意思是后向边中存在正的流量

则称 μ 是从 v_s 到 v_t (关于 X) 的**可增广链**

对于当前网络权重是否达到最大流的判断则有如下定理：

定理 (增广链定理) 一个可行流是最大流当且仅当不存在关于它的可增广链

必要性显然成立，下面证明充分性

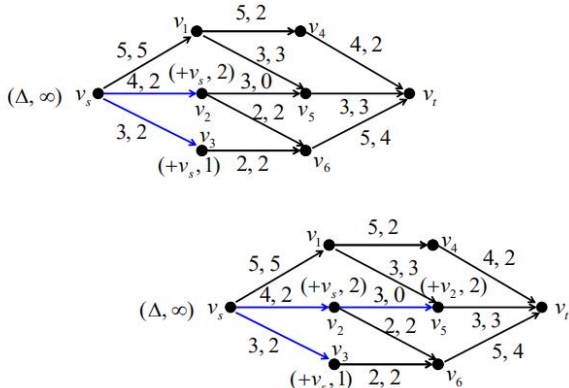
定义 S 为下述顶点的集合：对任意的 $v_k \in S$ 存在从 v_s 到 v_k 的链，满足可增广链的两个条件，即

如果 (v_i, v_j) 与链的方向相同，则 $0 \leq x_{ij} < c_{ij}$

如果 (v_i, v_j) 与链的方向相反，则 $0 < x_{ij} \leq c_{ij}$

用 \bar{S} 表示 S 的补集，不存在可增广链 $\Rightarrow v_t \in \bar{S}$

由上的分析，求解最大流的问题可等价转化为一个不断消除可增广链的过程。



我们前面论述的割集在这里依然可以作为判断依据，具体来说，在最大流问题中，**等于割集容量的可行流一定是局部网络的最大流**。而自然地，全局网络的最大流便是所有局部网络最大流的最小值。

定理 (最大流—最小割定理)

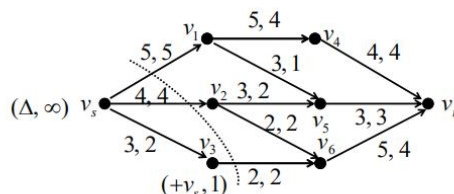
对于任何容量网络 $G = (V, E, C)$ ，从 v_s 到 v_t 的最大流的流量等于分割 v_s 和 v_t 的最小割集的容量

理由

$$\hat{W} \leq C(S, \bar{S}) = W \leq C(S', \bar{S}')$$

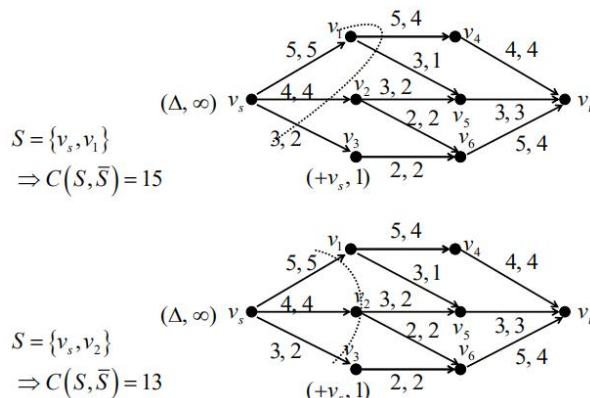
其中 \hat{W} 是任意可行流的流量， (S', \bar{S}') 是任意分割 v_s 和 v_t 的割集

在上面问题的最后一步我们可以观察得到：



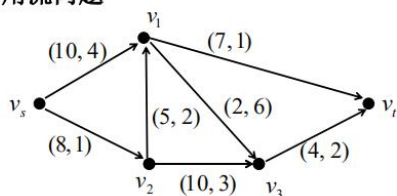
无法标号，停止，此时虚线所示割集容量已经等于可行流的流量，所以已经得到最大流

一些求割集容量的其它例子：



最小费用流问题是最大流问题的对偶问题，二者十分相似，但是这两者的处理方法截然不同。一般地，它有如下的标准形式：

例 最小费用流问题



括号内第一个数字是容量，第二个是单位流量费用

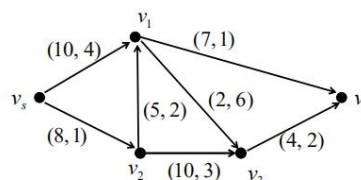
目标：从发点到收点的总的流量费用最小

约束：1) 容量约束，各边流量不大于容量

2) 流量平衡约束，各点进出流量总和相等

3) 从发点到收点的总流量为 w

显然最小费用流问题是一种特殊的线性规划问题：



$$\begin{aligned} & \min D^T X \\ & \text{s.t. } AX = B, 0 \leq X \leq C \end{aligned}$$

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ -10 \end{bmatrix}$$

$$D^T = [4 \ 1 \ 2 \ 6 \ 3 \ 1 \ 2] \quad C^T = [10 \ 8 \ 5 \ 2 \ 10 \ 7 \ 4]$$