第四章 图的一些计算机算法

4.1 最佳路径搜索算法

4.1.1 最短路径搜索算法

4.1.2 最大流和最小割

4.2 最优树搜索算法

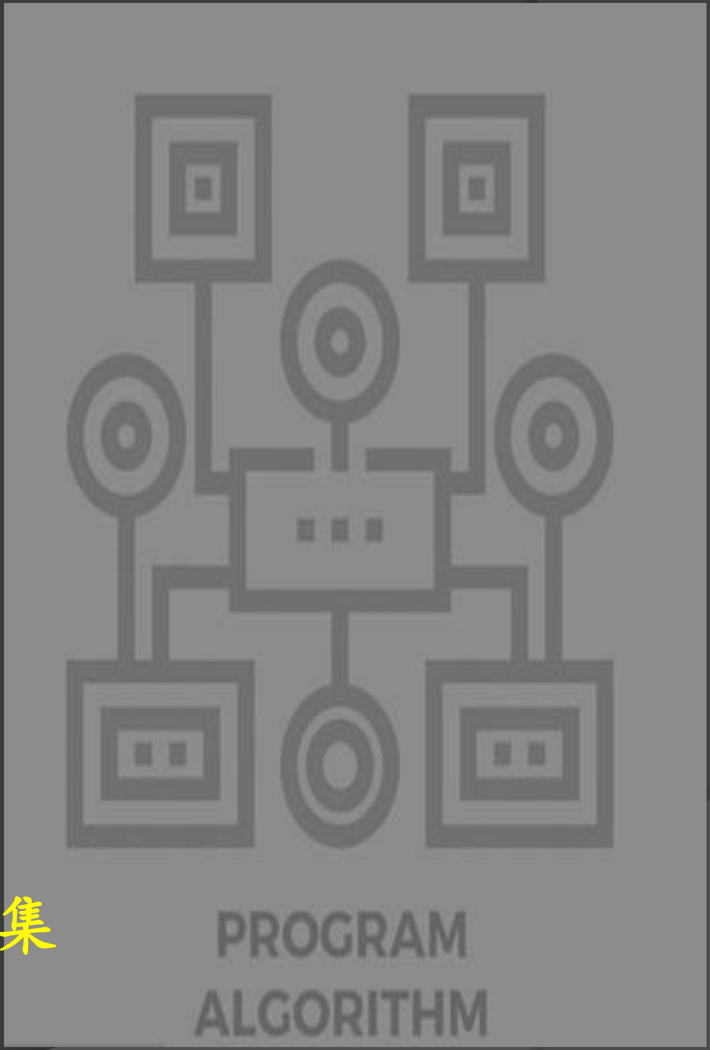
4.2.1 最小生成树搜索

4.2.2 Steiner树搜索

4.3 特殊点子集搜索算法

4.3.1 支配集、独立集、覆盖集

4.3.2 网络划分、社团发现



图的一些计算机算法

算法复杂度：是指在编写成可执行程序后，算法运行时所需要的资源开销，包括运算的**时间复杂度**和存储的**空间复杂度**。

设时间频度$T(n)$为一个算法中基本语句重复执行次数；设辅助函数$\boldsymbol{f(n)}$，如果有$\lim_{n\to\infty} {}^{T(n)}/_{f(n)} = c$（常数且$\neq \mathbf{0}$），则令$\boldsymbol{T(n) = O(f(n))}$，且称$\boldsymbol{O(f(n))}$为算法的时间复杂度。

非多项式
时间复杂度

常规等级：$O(1) < O(\log_2 n) < O(n) < O(n^k) < \boxed{O(c^n) < O(n!)}$

例如，$\mathrm{T}(n) = \mathrm{c}$，$O(1)$；$T(n) = n^2 + 2n$、$T(n) = 3n^2$，$O(n^2)$。

求解步骤：

- 找出算法中的基本语句；
- 计算基本语句执行次数的量级；
- 用$\boldsymbol{O}$表示算法的时间性能。

$\boldsymbol{for}$（$\boldsymbol{i = 1; i \le n; i{+}{+}}$）

$\boldsymbol{x{+}{+}}$；

$\boldsymbol{for}$（$\boldsymbol{i = 1; i \le n; i{+}{+}}$）

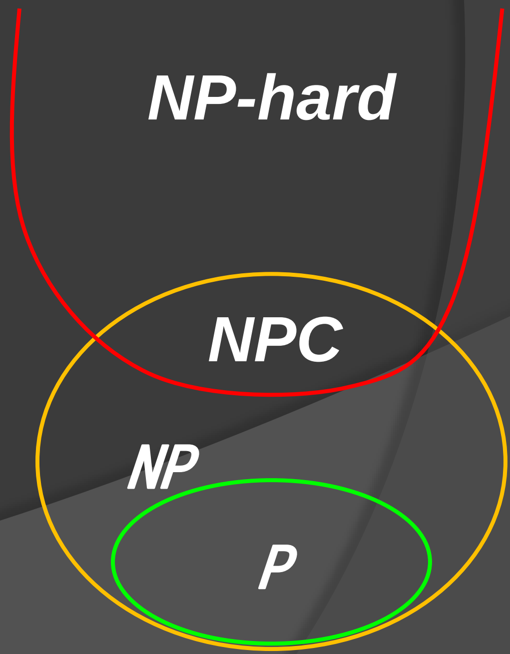
$\boldsymbol{for}$（$\boldsymbol{j = 1; j \le n; j{+}{+}}$）

$\boldsymbol{x{+}{+}}$；

$\boldsymbol{O(n + n^2)}$

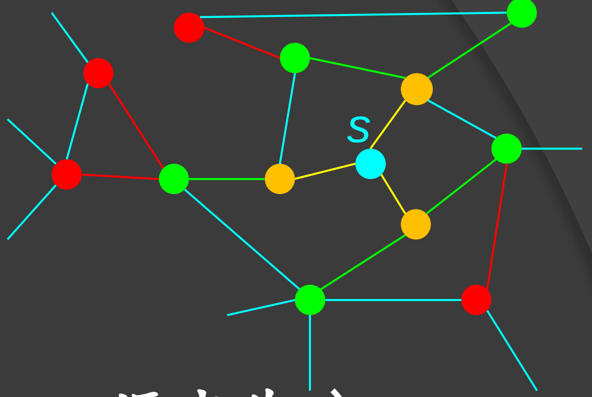
$\boldsymbol{= O(n^2)}$

- 确定性算法：算法执行中，每一步都有一个确定的结果。
- 非确定性算法：首先，猜测一个结果；然后，用一个确定性算法验证猜测结果是否是问题的一个合适解。
- 多项式(Polynomial）问题：在多项式时间内求解的问题。
- 多项式非确定性(Non-determinism Polynomial）问题：在多项式时间内验证一个“猜测”结果是否是问题的合适解。
- 完全多项式非确定性(NP-Complete)问题：NP问题中最复杂的问题，1）属于NP问题；2）所有NP问题皆可规约于此。
- 多项式非确定性难度(NP-hard)问题：NPC问题(最复杂的NP问题)，以及多项式时间内无法验证某个解正确与否的各类问题。



4.1 最佳路径搜索算法

1、最短路径搜索算法



图的一些计算机算法

广度优先搜索

算法过程(设网络节点数为n，边数为m，源点为s)

初始化：距离数组$D[n]$中$D[s]=0, D[n\neq s]=-1$；$d=0$。$O(n)$

迭代(循环)：

- 遍历$D[n]$，查找到$\boldsymbol{s}$距离为$\boldsymbol{d}$的所有节点$\boldsymbol{v_i}$； $O(rn)$
- 查找这些节点$\boldsymbol{v_i}$的所有邻居$\boldsymbol{v_j}$； $O(m)$
- 如果所有$\boldsymbol{D[v_j]\neq -1}$，算法结束；
- 如果$D[v_j]=-1$，$D[v_j]=d+1$；
- $d=d+1$。 $O(1\times r)$

$\boldsymbol{O(n+rn+m+r)}$

↓

$O(n^2+m)$

or

$O(n\log n+m)$

改进的广度优先算法

输入：非加权(有向)图$G=(V,E)$和初始点s。

初始化：$S=\{s\}$；$T=\emptyset$；$d(v,s)=0, v\in V$。 $O(n)$

迭代：

$\boldsymbol{O(m+n)}$

- 如果$S=\emptyset$，算法结束；
- 否则， $v_i=S[1]$； $O(1\times r)$
- 所有v_i的邻居且$\notin S\cup T$的v_j加入S队尾； $O(m)$
- $d(v_j,s)=d(v_i,s)+1$； $O(1\times r)$
- 从S中删除v_i，将其加入T的队尾。

稀疏网络

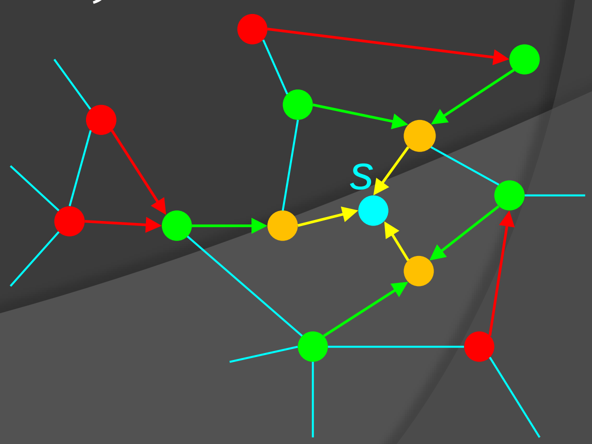
$m\propto n$

$\downarrow$

$\boldsymbol{O(n)}$

添加一条由v_j指向v_i的有向边。

算法结束时，生成的有向图即为以s为根的一棵最短路径树。



Dijkstra算法

输入：非负权值图$G=(V,E,w)$和初始点s。

初始化：$S=\{s\}$；$D(s)=0$；对所有点$v_k \neq s$，

如果$a_{sk}=1$，$D(k)=w_{sk}$；否则，$D(k)=\infty$。

创建s为根的树，连v_k到根的边。

$O(n)$

迭代：

- $D(i)=\min_{k\notin S} D(k)$； $O(n)$
- 将v_i加入S，即$v_i \to S$；
- 遍历v_i的所有邻居v_j， $O(m/n)$

  $D(j)=min\{D(j), D(i)+w_{ij}\}$；
- 如果$S=V(G)$，结束算法。

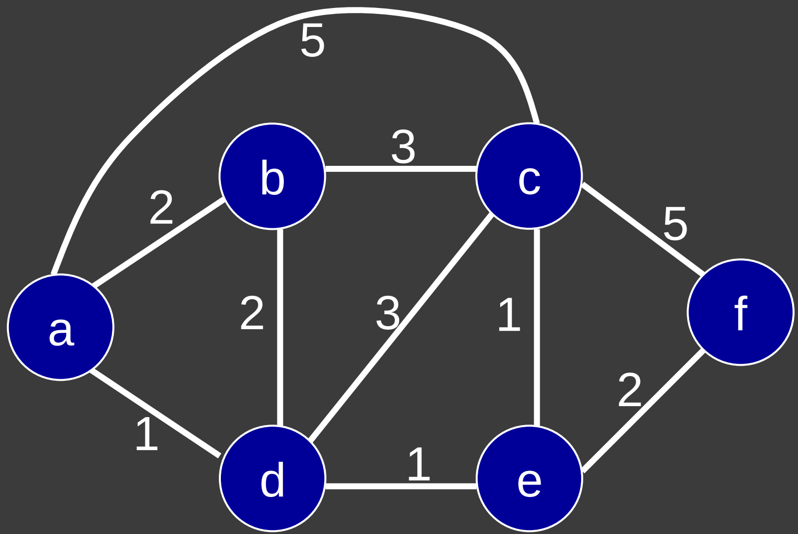
如果$D(i)+w_{ij}<D(j)$，删除v_j原来去往s的边，增加v_i到v_j的边。

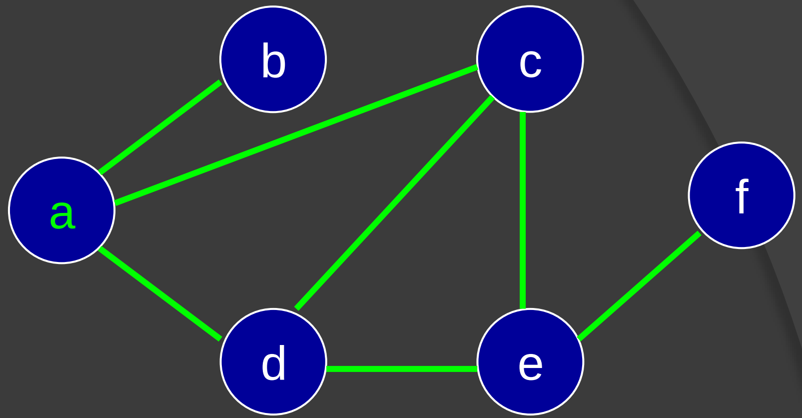
$$\boldsymbol{O\left(n\left(n+\frac{m}{n}\right)\right)=O(n^2+m)}$$

$m \propto n$，$\boldsymbol{O(n^2)}$

如果采用二叉堆技术存储数组，整个算法的复杂度为$\boldsymbol{O(n\log n)}$。

图的一些计算机算法





迭代	S	$T(b)$	$T(c)$	$T(d)$	$T(e)$	$T(f)$
0	{a}	2	5	1	∞	∞
1	{a,d}	2	4	1	2	∞
2	{a,d,b}	2	4	1	2	∞
3	{a,d,b,e}	2	3	1	2	4
4	{a,d,b,e,c}	2	3	1	2	4
5	{a,d,b,e,c,f}	2	3	1	2	4

a节点路由表：

目的节点	后继结点
a	-
b	b
c	d
d	d
e	d
f	d

Bellman-Ford算法

输入：没有负权值环路的图和初始点s。

初始化：距离数组$D(k \neq s) = \infty$，$D(s) = 0$。 $O(n)$

迭代：

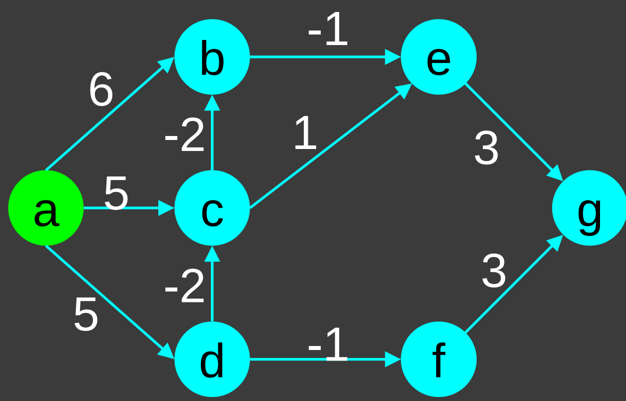
- 遍历所有节点$v_i (i \neq s)$；
  - 遍历v_i的所有邻居v_j，计算$D(j) + w_{ij}$，
    $D(i) = min\{D(j) + w_{ij}\}$； $O(n \times \frac{m}{n})$
- 如果$D(k)$没有更新，结束迭代； $\boldsymbol{O(mn)}$

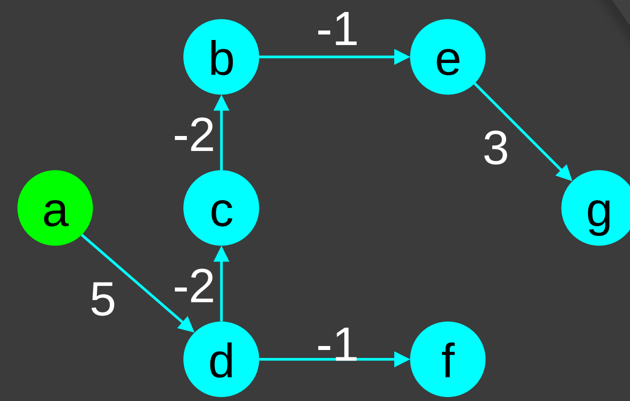
迭代：

- 遍历检查每一条边e_{ij}， $O(m)$

  如果$D(i) + w_{ij} \geq D(j)$，结束算法；

  否则，存在负权值环路，无法求解最短路径。 $\boldsymbol{O(n^2)}$





迭代	$D(a)$	$D(b)$	$D(c)$	$D(d)$	$D(e)$	$D(f)$	$D(g)$
0	0	∞	∞	∞	∞	∞	∞
1	0	6	5	5	5	4	7
2	0	3	3	5	2	4	5
3	0	1	3	5	0	4	3
4	0	1	3	5	0	4	3

边检查示例：

边(c,e)：$D(c)=3$、$D(e)=\mathbf{0}$、$w_{ce}=1$，

$D(c)+w_{ce}=3+1>D(e)=0$。

b的邻居a、c。

$D(a)+w_{ab}=0+6$

$D(c)+w_{cb}=\infty-2$

故，$D(b)=\mathbf{6}$。

b的邻居a、c。

$D(a)+w_{ab}=0+6$

$D(c)+w_{cb}=5-2$

故，$D(b)=\mathbf{3}$。

2、最大流/最小割

点连通度：图中指定的两点之间的点独立路径个数。

边连通度：图中指定的两点之间的边独立路径个数。

最大流/最小割定理：一个流量网络中，从源点到达目标点的最大流量等于这两点之间最小边割集中每条边上的流量之和。

设图中指定两点之间的边独立路径个数为n，每条路径的流量为r,利用Menger定理可以证明指定两点之间流量的上限为nr，流量的下限为nr，即最大流等于边独立路径数量乘以边的流量。

在无向加权网络中，可将每条边上分配的流量作为权值，两点间的最大流就等于其加权最小边割集中所有边的权重之和。

✓ 无向图中，任意两点之间有三个参数是相等的：两点间的边连通度；两点间最小边割集的规模；两点间的最大流量。

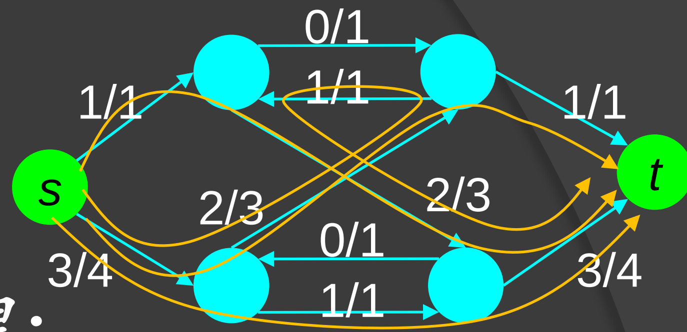
最大流问题

设C_{ij}为点v_i和v_j间边的最大流量，

f_{ij}为点v_i和v_j间边的当前流量，有定理：

- $e_{ij} \in E$，$f_{ij} \le C_{ij}$。
- $\sum_{e_{ki}\in E} f_{ki} = \sum_{e_{ij}\in E} f_{ij}$，$\boldsymbol{s}$和$\boldsymbol{t}$除外。

${}^{f_{ij}}/_{C_{ij}}$表示边e_{ij}的权值/最大容量。



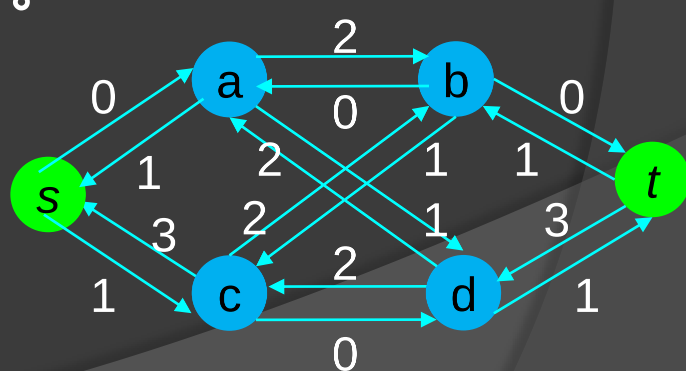
$$\boldsymbol{max \sum\nolimits_{s,t\in V} f_{s.t}}$$

$$\boldsymbol{max\, f_{s.t} = 4}$$

剩余流量：$r_{ij} = C_{ij} - f_{ij}$；且令$r_{ji} = -f_{ij}$。

剩余网络：流量r_{ij}和r_{ji}组成的网络。

增广路径：已知一个流网络$\boldsymbol{G}$和流$\boldsymbol{f}$，其剩余网络中从$\boldsymbol{s}$到$\boldsymbol{t}$的流量大于零的简单路径即为增广路径。



已无增广路径

Ford-Fulkerson方法

初始化：给定图$\boldsymbol{G}=(V,E,w)$，

对$e_{ij}\in E$，令$f_{ij}=0$。

迭代：

- 构建剩余网络； $O(m)$
- 利用广度优先最短路径算法，搜索一条增广路径； $O(n)$
- 搜索成功，将路径上所有边的$f_{ij}+1$，且$f_{s.t}+1$；否则，算法结束。 $O(1)$

$w_{k_s}=\sum_{v_i\in V}a_{si}w_{si}$，

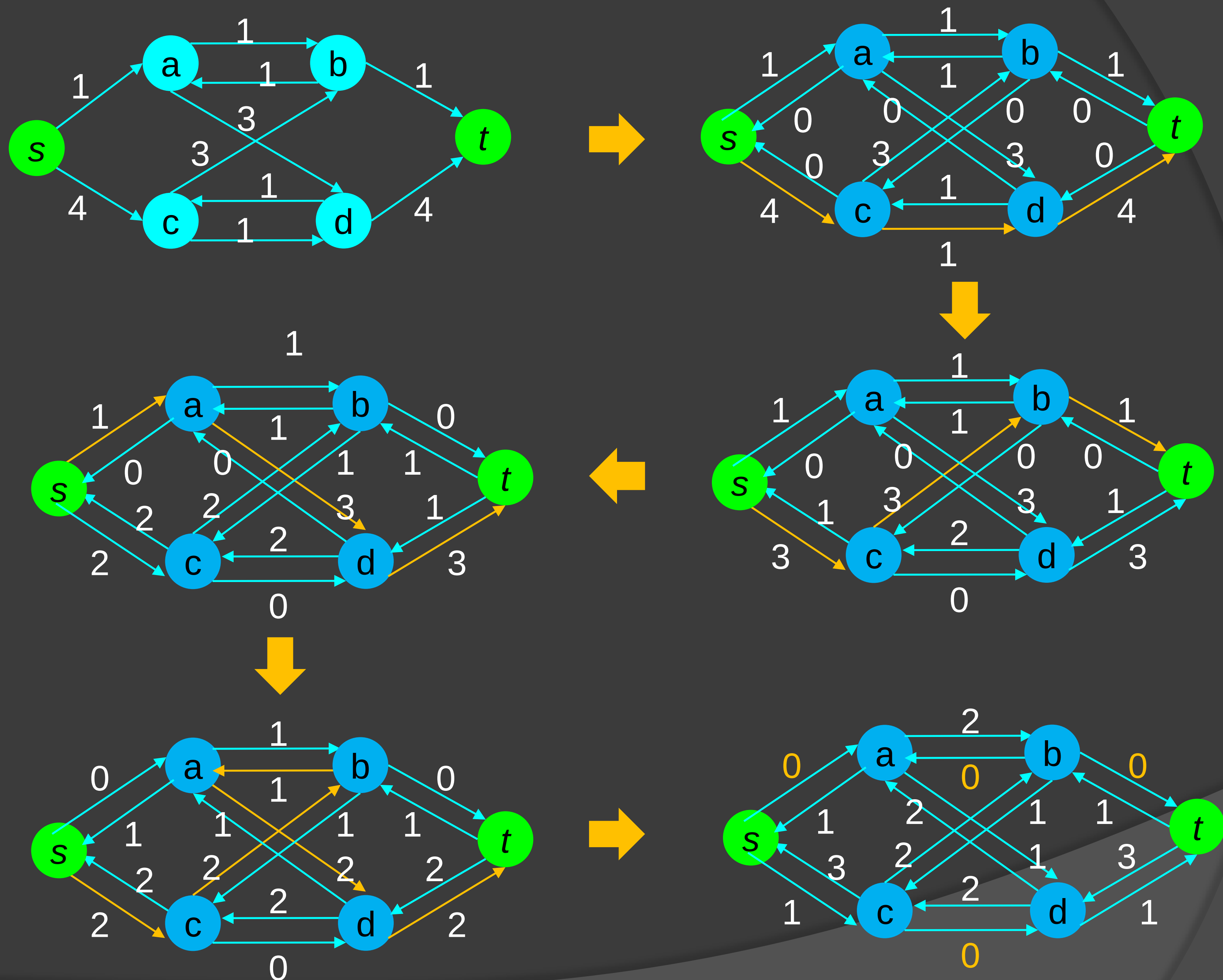
$w_{k_t}=\sum_{v_j\in V}a_{jt}w_{jt}$，

$\boldsymbol{O[min(w_{k_s},w_{k_t})(m+n)]}$

迭代次数 $O[min(w_{k_s},w_{k_t})]$，$\boldsymbol{s}$和$\boldsymbol{t}$之间可能的最多路径数量，每条路径仅承担一个单位的流量。

最大流量/最小割定理：在有向加权网络中，给定两个点之间的最大流量等于这两点之间权重之和最小的边割集的边数。

图的一些计算机算法

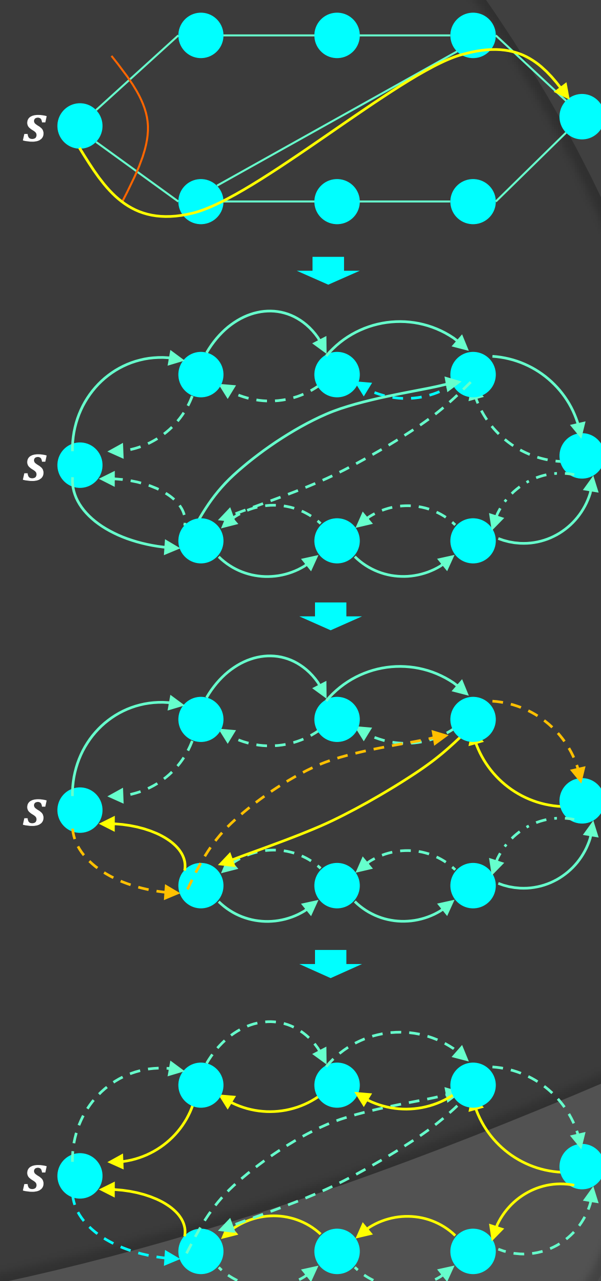


3、独立路径搜索

边独立路径搜索方法

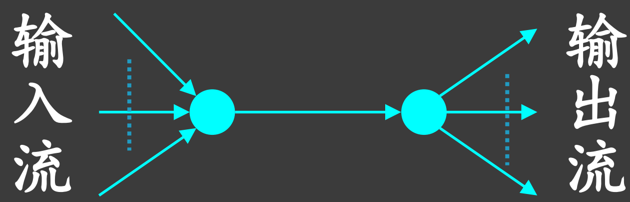
- 将图中边的权值都设为1；
- 利用Ford-Fulkerson方法查询所有的增广路径；
- Ford-Fulkerson算法结束后，删除所有没有承载网络流量的边，沿剩余边重构独立路径。

令V_s为从s沿剩余图可到达的点的子集，V_t为不在V_s中的点的子集，原图中连接V_s和V_t的边的集合即为一个最小边割集。



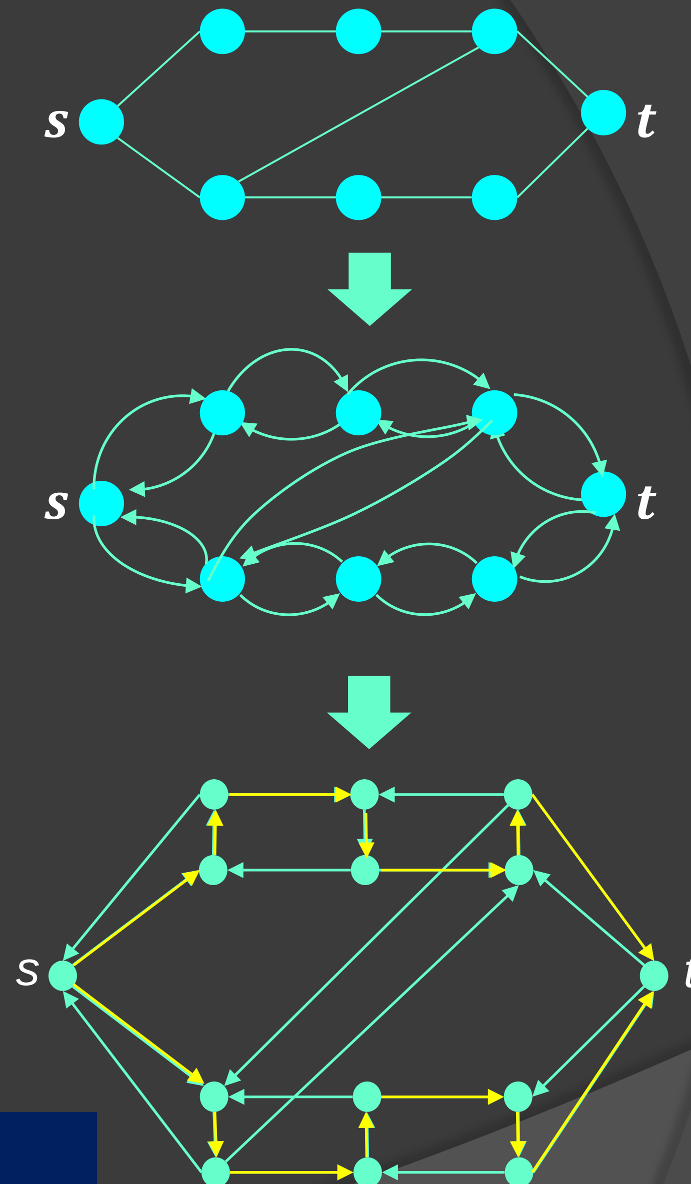
点独立路径搜索方法

- 将图中边的权值都设为1；
- 构建剩余网络；
- 修改每个节点为两节点有向图；



- 利用标准增广路径搜索算法求出边独立路径；
- 输出的边独立路径即为原始网络中的点独立路径。

因为每个节点只能经过一次，所以，将节点内改为单向有向图结构。



4.2 最优树搜索算法

1、最小生成树算法

Kruskal算法

输入：无向加权连通图$G=(V,E,w)$。

初始化：构造n个$v\in V$点的无边图，

$e_{ij}\in E$按照w_{ij}升序排序，$H=\emptyset$。

$O(m\log m)$

迭代：

- 取出w_{ij}最小的边e_{ij}；
- 若v_i、v_j独立，合并为一棵树，$e_{ij}\to H$；
- 否则，丢弃e_{ij}；
- 若H中边个数为$n-1$，结束算法。

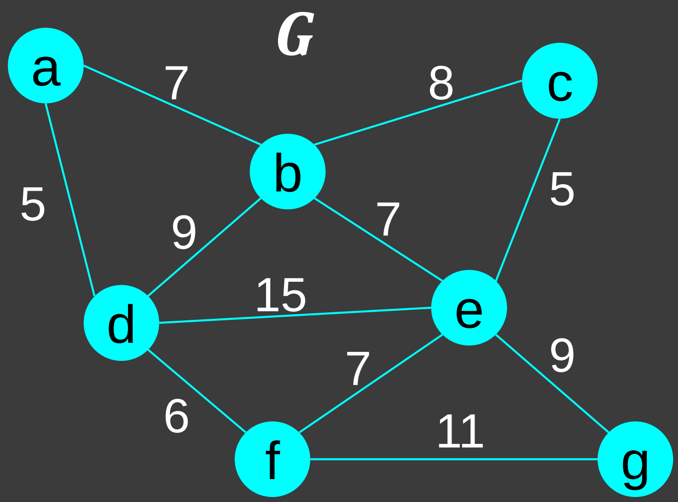
$O(m\log n)$

并查集运算：(Union Find)

- 查询a和b是否为同一组；
- 将a和b合并为同一组。

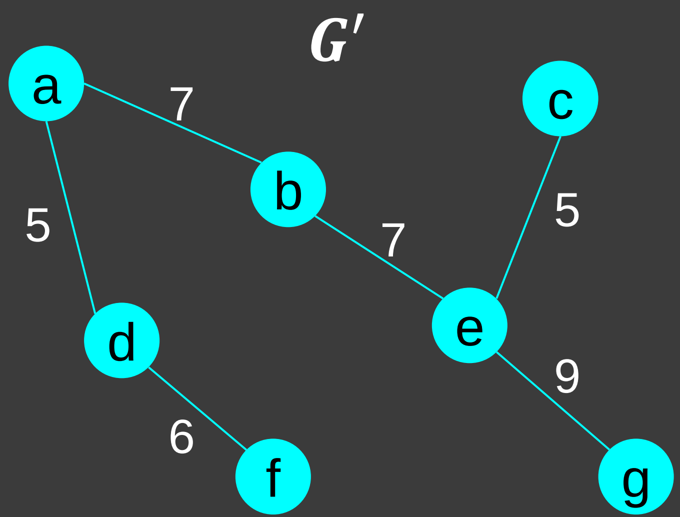
a和b所属树的树根相同则同属一组；将两根连接则两组合并为一组。

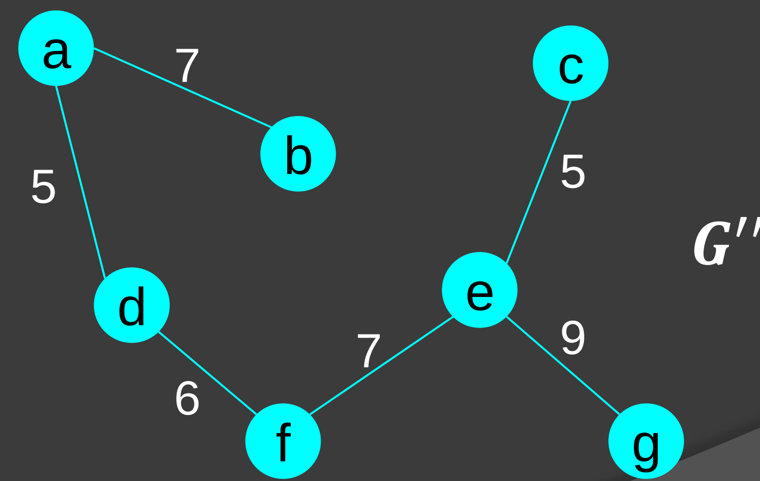
$O(\log n)$



e_{ad}	$w_{ad}=5$	e_{ef}	$w_{ef}=7$
e_{ce}	$w_{ce}=5$	e_{bc}	$w_{bc}=8$
e_{df}	$w_{df}=6$	e_{eg}	$w_{eg}=9$
e_{ab}	$w_{ab}=7$	e_{fg}	$w_{fg}=11$
e_{be}	$w_{be}=7$	e_{de}	$w_{de}=15$

$n=7$，$m=10$。





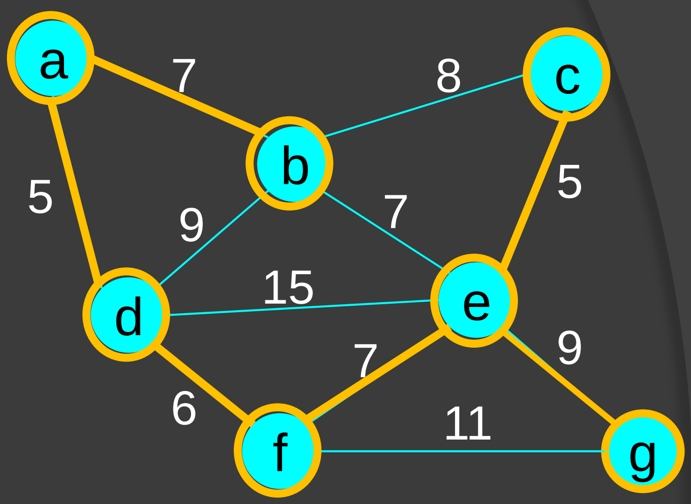
$$\sum_{e_{ij}\in G'} w_{ij}=29$$

$$\sum_{e_{ij}\in G''} w_{ij}=29$$

Prim算法

输入：无向加权连通图$G=(V,E,w)$和起始点s。

初始化：$V_{new}=\{s\}$，$E_{new}=\emptyset$。



迭代：

- 如果$V_{new}=V$，算法结束；
- 遍历E，搜索e_{ij}，

  其中，$v_i\in V_{new}$，

  $v_j\notin V_{new} and \in V$；　　$O(m)$

  $\boldsymbol{O(nm)}$

- 搜索e_{kl}，满足$w_{kl}=\min_{k\in i,l\in j} w_{ij}$；　　$O(m)$
- $v_l\rightarrow V_{new}$，$e_{kl}\rightarrow E_{new}$。

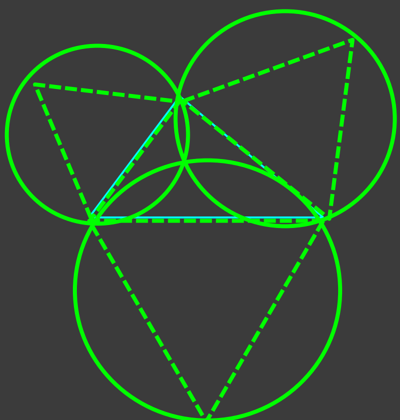
使用二叉堆和邻接表，复杂度为 $\boldsymbol{O(n\log m)}$。

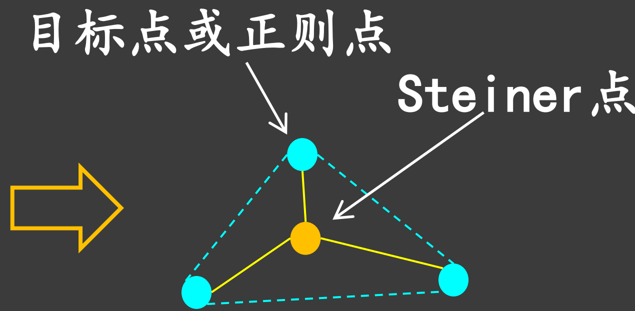
输出：使用V_{new}和E_{new}构造最小生成树。

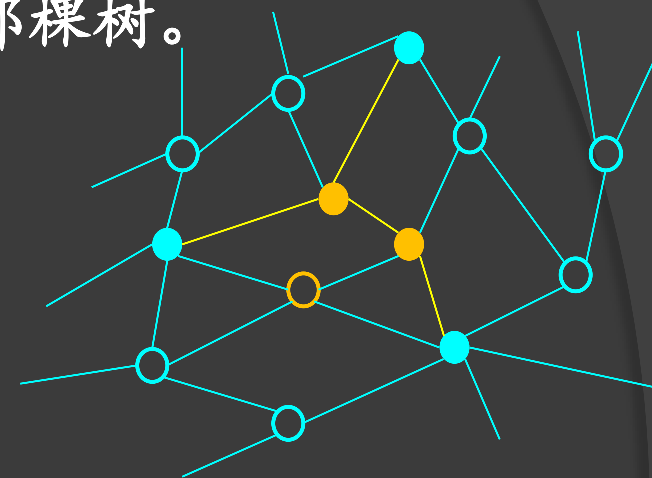
2、Steiner树问题

Steiner树是总代价最小的分布树，也就是连接一个给定图中k个特定点的分支所需代价最少的那棵树。

最短网络问题







应用：通信网络规划与设计；确定一组成员间通信传输的最佳路由。

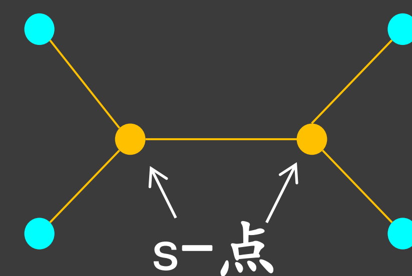
- 平面Steiner树问题：在二维平面给定点集P，添加新点集S，使$P\cup S$的最小生成树总长度小于P的最小生成树总长度。
- 图的Steiner树问题：寻找图$\boldsymbol{G}(V,E,w)$中的一棵树$\boldsymbol{T}(P,U,w^T)$，P为V的子集，U为E的子集，使得$w^T=\sum_{u\in U} w_u$最小。

平面上的Steiner最小树

设平面上点$p_1(x_1,y_1),p_2(x_2,y_2)$，以两点间欧几里得距离$d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$为度量的称**欧氏Steiner最小树**。

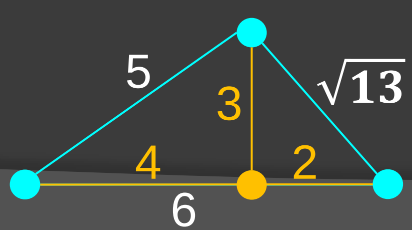
- 设平面上点数为$\boldsymbol{n}$，其欧氏Steiner最小树满足s-点数$\le n-2$。
- 欧氏Steiner最小树上与s-点关联的边有3条，其中任意2条边之间的夹角为120°。
- **满Steiner树**中的s-点数等于$n-2$。

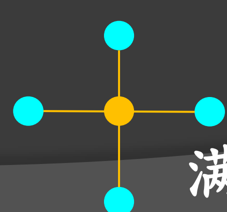
欧氏Steiner最小树

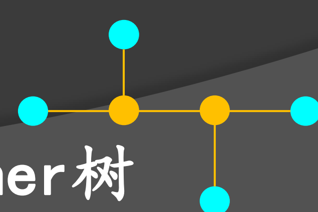


满Steiner树

以p_1,p_2之间的曼哈顿距离$d=|x_1-x_2|+|y_1-y_2|$为度量的称为**绝对值Steiner最小树**，其连边由水平或垂直线段组成。



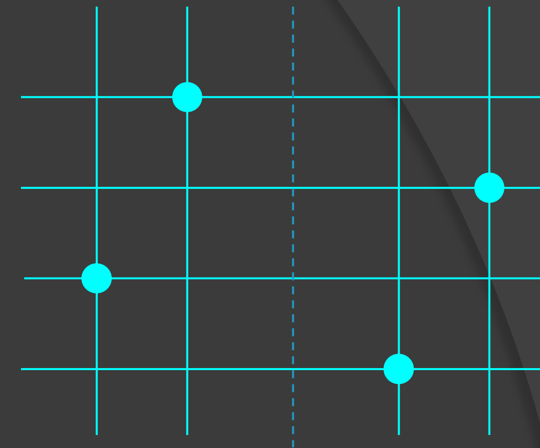


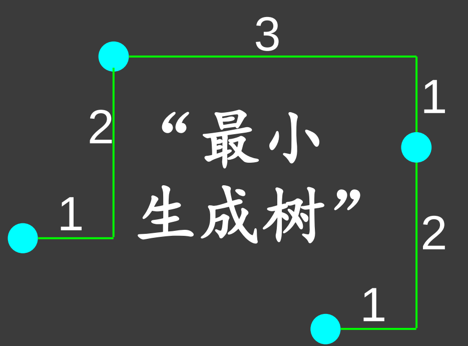


满Steiner树

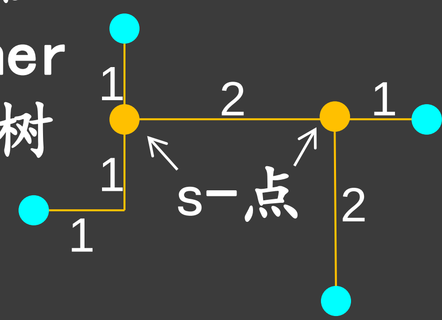
s-点数$\le n-2$

- **Hanan定理**：任意目标点集P的绝对值Steiner最小树的s-点均在P的Hanan网格格点上(P中所有点的水平线与垂直线的交叉点）。





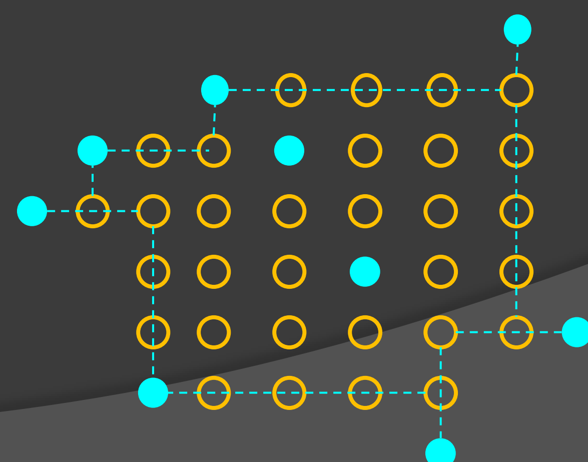
绝对值Steiner最小树



$L_{最小生成树} = 10$

$L_{Steiner最小树} = 8$

- 设$|P| = n$，可选的s-点位置最多有$n^2 - n$个。
- $3 \leq$任意s-点的度数≤ 4。
- 设n个目标点所围成的区域为凸包形状，则有效的m个s-点都在凸包之内。



$n = 9$

$9 \times 8 = 72$

$m = 31$

穷举法：

- 给定P，令$C=\infty$，从m个有效位置中任选$s\leq n-2$个。
- 构造这$n+s$个点的完全(连接)图$\boldsymbol{G'}$，每条边的权值为$\boldsymbol{P}$的Hanan网格中两点之间最短的直角折线距离之和。
- 用Kruskal算法计算$\boldsymbol{G'}$的最小生成树T_s以及其边的总长C_s。
- 如果$C_s<C$，$C_s\rightarrow C$，$T_s\rightarrow T_p$；否则，丢弃T_s。
- 另外选取s个点，重复2、3、4步骤，直至穷尽m个位置点的所有可能的组合。
- 输出$\boldsymbol{T_p}$，即为基于曼哈顿距离的绝对值Steiner最小树。

迭代次数为$\left(\begin{bmatrix}m\\0\end{bmatrix}+\begin{bmatrix}m\\1\end{bmatrix}+\cdots+\begin{bmatrix}m\\s\end{bmatrix}\right)$。

前页示例中，9个站点的绝对值Steiner树计算共需3572224次迭代，若每次迭代耗时0.01秒，大约需要10个小时。

启发式算法思想：

- 计算目标点集P的最小直角折线生成树，记录其总长度c。
- 有效的位置点集M中选出一个s加入S，然后，计算$P \cup S$的最小生成树$T_{P\cup S}$。
- 如果计算的$T_{P\cup S}$总长度$< c$，更新c值为$T_{P\cup S}$总长度；否则，从S中删除第二步中加入的位置点s。
- 如果S中的点数达到$n-2$，或者$M=\emptyset$，结束算法；否则，返回第二步。
- 输出S为s-点，构造Steiner最小树。

最多迭代n^2-n次。复杂度计算还需乘以每次迭代计算最小生成树的开销。

其他改进算法：

- 贪婪算法
- 模拟退火算法
- 遗传算法
- 蚁群算法

…

图的Steiner最小树

求解图$\boldsymbol{G}=(V,E,w)$的Steiner最小树属于NPC问题，但有几种特殊情况可以在多项式时间内求得最优解。

- $|P|=2$，求两点之间的最短路径。 $O(|V|^2)$
- $P=V$，求图的最小生成树问题。 $O(|V|^2)/O(|E|\log|E|)$
- $|P|=3$，只有一个Steiner点且与目标点构成星形最短路径连接。 $O(|V|^2)/O(|V||E|\log|V|)$

图的Steiner最小树的性质：

- s-点的数目最多不超过正则点的个数减二，即$|P|-2$。
- 度数为1的非正则点一定不属于Steiner最小树。
- 度数为2的非正则点v_k的两个邻居v_i,v_j，若存在$e_{ij}\in E$，且$w_{ik}+w_{jk}>w_{ij}$，则v_k不属于Steiner最小树。

求解算法分类：

- 精确算法，枚举所有可能的Steiner点并计算相应的树的总长度。$O(2^{|V|-|P|}|V|^2+|V|^3)/O(3^{|P|}|V|+2^{|P|}|V|^2+|P|^2|V|)$
- 启发式算法，在多项式时间内寻找近似的最优解，研究得较多的是基于最小生成树的启发式方法，其实用性较好。

修剪最小生成树

- 构造图G的最小生成树；
- 删除图中所有度数为1的非正则节点及其连边。

ADH(Average Distance Heuristic)算法

- 初始化：森林$T=(T_1,\cdots,T_{|P|})$由$|P|$个孤立点组成； $\boldsymbol{O(|V|^3)}$
- 迭代：$v\in V$，如果$T_p\leftrightarrow v\leftrightarrow T_q=\min\limits_{i,j\in T}\{d_{iv}+d_{vj}\}$，用经过$v$的两条最短路连接$T_p$和$T_q$，直到$T$变为一棵树。

KMB算法

输入：无向加权图$G=(V,E,w)$以及目标点集$P\subseteq V$。

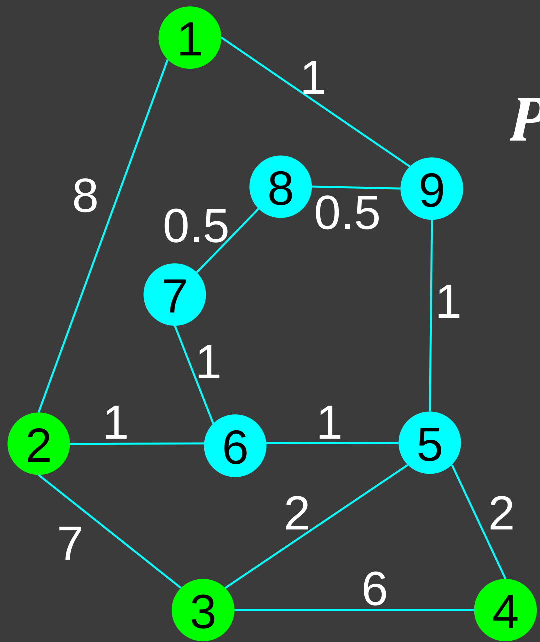
- 基于P构造完全图$G'=(V',E',w')$，其中$V'=P$，对所有$v_i,v_j\in V'$，连边E'_{ij}，$w'_{ij}=v_i,v_j$之间最短路径权值。
- 计算G'的最小生成树T'，如果有多棵树则随机选取一棵。
- 构造G的子图G_s，只保留T'中的边所对应的最短路径上的点和边，如果对应有多条最短路径则随机选一条。
- 计算G_s的最小生成树T_s，如果有多棵则随机选取一棵。
- 删除不属于P的叶子点和连边，生成G的Steiner树T_p。

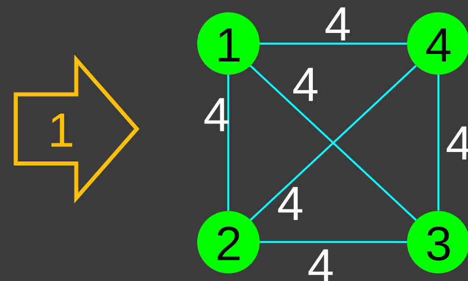
复杂度：步骤1=$O(|P||V|^2)$，步骤2=$O(|P|^2)$，步骤3=$O(|V|)$，步骤4=$O(|V|^2)$，步骤5= $O(|V|)$。

$O\left(|P||V|^2\right)$

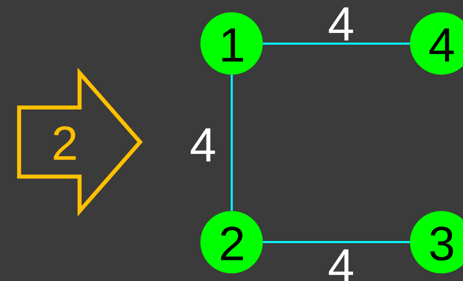
目标点

$$P = (1, 2, 3, 4)$$

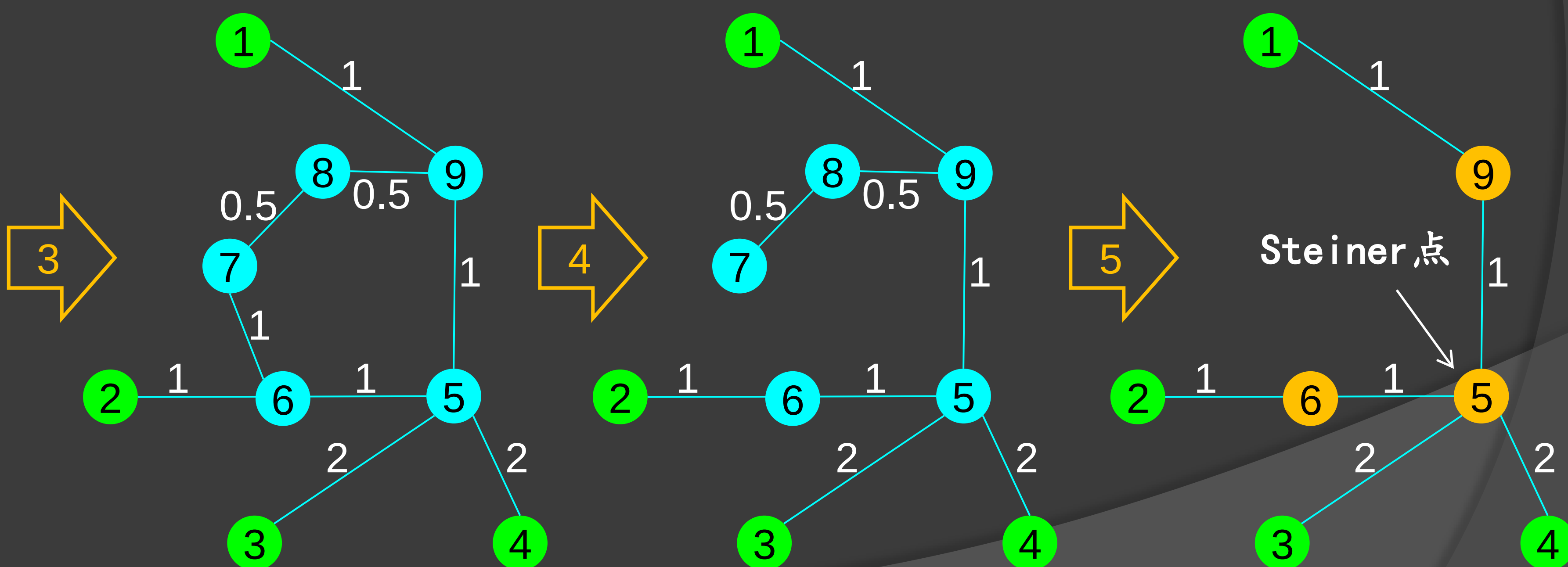




G'



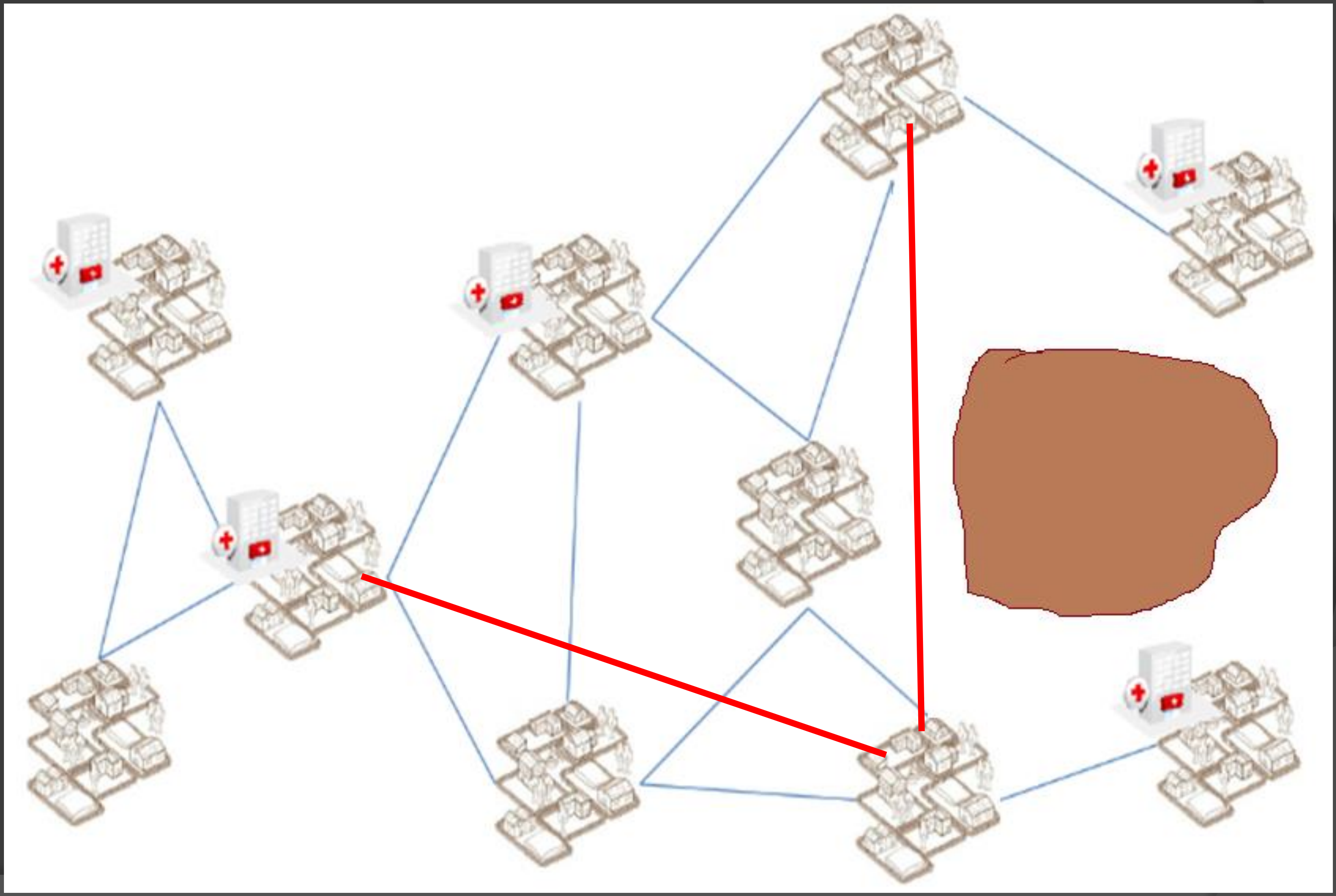
T'



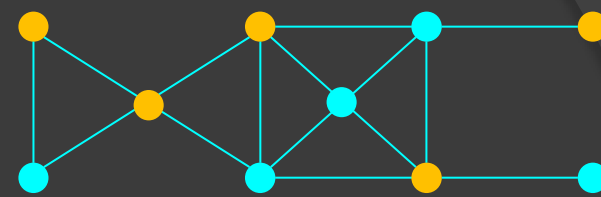
G_s T_s T_p

4.3 特殊点子集搜索算法

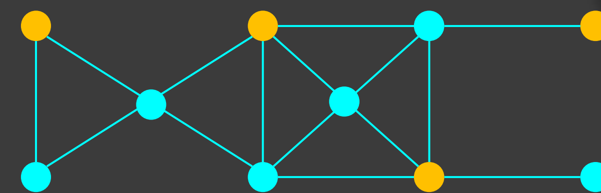
1、支配集、独立集、覆盖集



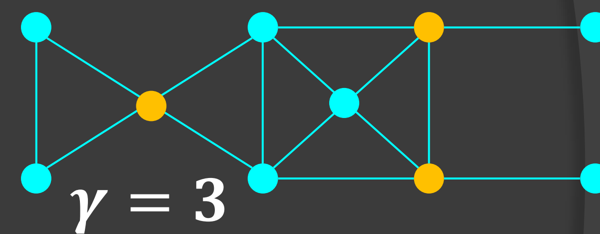
支配集： 图$G=(V,E)$中，设$D\subseteq V$，若每个点$v_j\notin D$都与某个$v_i\in D$点之间有一条边$e_{ij}\in E$，则D是图G的一个**点支配集**。



- 支配集中任何一个真子集都不再是支配集的称为**极小支配集**。



- 所有极小支配集中节点数最少的称为**最小支配集**，其节点个数称为**支配数**，记为$\gamma(G)$。



支配集的一些特性：

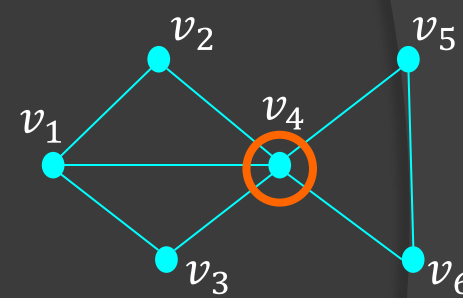
$$\gamma(G)\le\frac{1+\ln(k_{min}+1)}{k_{min}+1}|V|，\quad \frac{|V|}{1+k_{max}}\le\gamma(G)$$

- 无孤立点的图G必有一个最小支配集D，但不一定唯一。
- 最小支配集一定是极小支配集，反之不一定成立。
- 如果D中点的邻居与D的交集为空，则D是极小支配集。

最小支配集计算方法

计算方法：根据定义(每个点+其邻居点)与(其它点+其邻居点)的交集就是极小支配集中的元素，故将$\prod_{i=1}^{n}\left(v_i+\sum_{v_j\in e_{ij}} v_j\right)$展开成积之和的形式，每一项给出一个极小支配集。

$$\prod_{i=1}^{n}\left(v_i+\sum_{v_j\in e_{ij}} v_j\right)=(v_1+v_2+v_3+v_4)\cdot$$
$$(v_2+v_1+v_4)(v_3+v_1+v_4)(v_5+v_4+v_6)\cdot$$
$$(v_4+v_1+v_2+v_3+v_5+v_6)(v_6+v_4+v_6)$$
$$=v_1v_5+v_1v_6+v_4+v_2v_3v_5+v_2v_3v_6$$

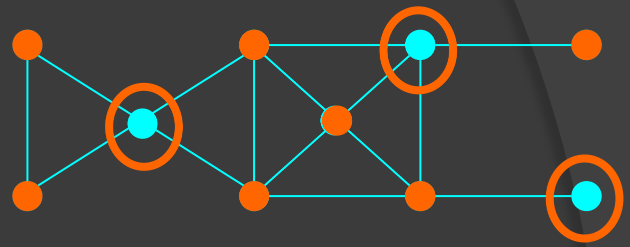


展开式中最多有$\sum_{i=1}^{|V|}\binom{|V|}{i}=2^{|V|}-1$个乘积项，令$|V|=n$，故计算复杂度为$O(2^n)$量级。一些精确算法基于图的搜索技术求解，利用支配集特性约减子问题个数，可达$O(1.5063^n)$。

一种贪心算法的思想：

设 $N(v_i) = \{v_j \in V\ and\ e_{ij} \in E\}$，$N[v_i] = \{v_i \cup (v_j \in \boldsymbol{e_{ij}})\}$。

初始化：输入连通图$\boldsymbol{G} = (V, E)$，$D = \emptyset$。



迭代

- 从$\boldsymbol{V}$中搜索度数最大的点$\boldsymbol{v_i}$；$O(\log|V|)$
- $D = D \cup v_i$，$G - N[v_i] = V - v_i - v_j\ and\ E - e_{ij}$；　$O(1)$
- 如果$V = \emptyset$，结束迭代。

$$\boldsymbol{O}(\frac{|\boldsymbol{V}|}{\boldsymbol{3}}\log|V|)$$

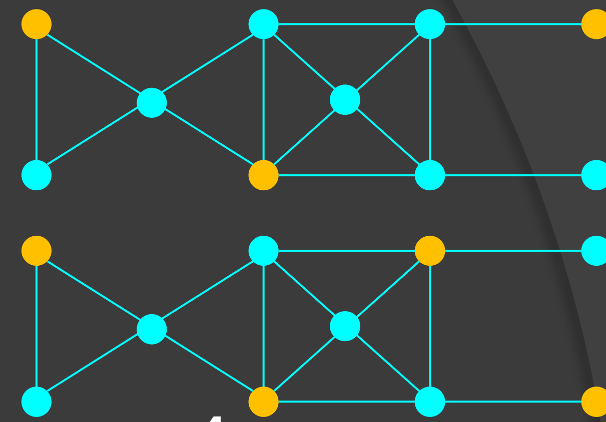
输出支配集$\boldsymbol{D}$。

最小连通支配集：等价于寻找图的一棵生成树的枝干节点，要求树的叶子节点数最大，类似于货郎担问题和Steiner树问题。

最小独立支配集：与最小连通支配集恰恰相反。

最小加权支配集：为每个节点赋予一个权值。

独立集：图$G=(V,E)$中，设 $I\subseteq V$，若$\forall v_i\in I$和$v_j\in I$，有 $e_{ij}\notin E$，则I是G的**点独立集**。



- 若I加入$(V-I)$中的任一节点后不再是独立集，则I是G的一个**极大独立集**。
- 所有 I 中点个数最多的为**最大独立集**，其点的个数记为$\alpha(G)$。

$\alpha=4$

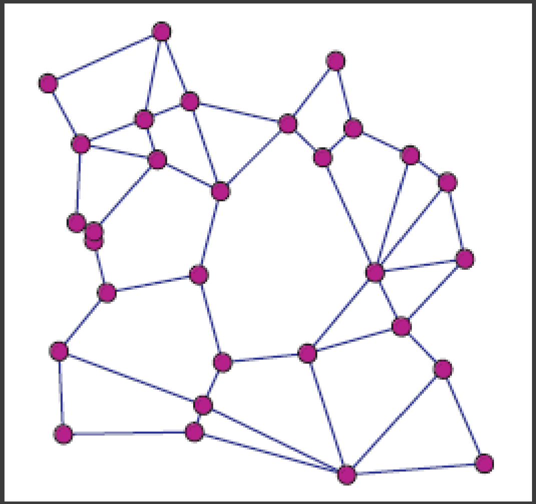
图G的极大独立集必是G的极小支配集，但反之不成立。

最大独立集的选择策略：

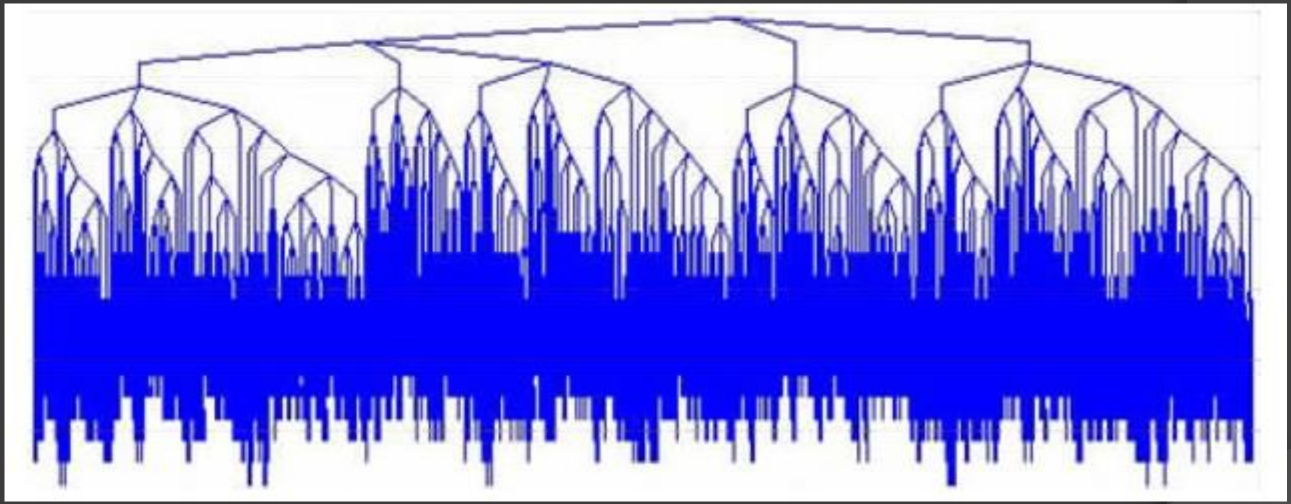
- G中度数$k(v)=0$的点v属于I_{max}。
- $k(v)=1$，若v属于I_{max}，v的邻居$N(v)$必不属于I_{max}。
- 所有$k(v)=2$(环)，任选一点加入I_{max}后按前2特性处理即可。
- 如果$e_{ij}\in E$，且v_j支配v_i，可令$v_j\notin I_{max}$。
- $v\notin I_{max}$且$N[v]$并非全相连， $N(v)$可有≥ 2的点属于I_{max}。

最大点独立集精确求解算法

分支化简降阶技术思想：分析问题性质进行降阶；对无法降阶的问题，将其分解为小规模的子问题（分支），递归地对子问题分解，直到分为基本问题；对分支后的基本问题求解。此类算法也被形象地称为搜索树。



搜索过程树



分支方法的复杂度公式：$T(n)=T(n-k_1)+\cdots+T(n-k_r)$。

令$T(n)=x^n$，求$x^n=x^{n-k_1}+\cdots+x^{n-k_r}$的唯一解或者最大解。

设解为$1\leq c\leq 2$，时间复杂度为$O(c^n)$。

最大(点)独立集MIS(G, I)

图的一些计算机算法

1. 初始化：输入图$G=(V,E)$，$I=\emptyset$。

2. 如果有$k(v_i)=0$，返回MIS$(G-v_i$，$I\cup v_i)$；

3. 如果有$k(v_i)=1$，返回MIS$(G-N[v_i]$，$I\cup v_i)$；

4. 如果有$k(v_i)=2$，继续判断：

$O(1^n)$

4.1 如果所有$k(v_i)=2$，任选一v_i，返回MIS$(G-N[v_i]$，$I\cup v_i)$；

4.2 如果v_i有邻居v_j, v_l，且$k_j\geq 3$或$k_l\geq 3$，继续判断：

4.2.1 $e_{jl}\in E$， 返回MIS$(G-N[v_i]$，$I\cup v_i)$；

4.2.2 否则，分成2个问题分别求解：

(1) MIS$(G-N[v_i]$，$I\cup v_i)$；

(2) MIS$(G-N[v_j]-N[v_l]$，$I\cup v_j\cup v_l)$；

$T(n)=T(n-3)+T(n-5)$

$c\approx 1.194$

$O(1.194^n)$

5. $k(v_i)=3$，令$N(v_i)=\{v_j, v_l, v_o\}$，继续判断：

5.1 若有3条互连的边，返回MIS$(G-N[v_i]$，$I\cup v_i)$；

5.2 若有2条互连的边$(e_{jl}, e_{jo}/e_{lo}, e_{lj}/e_{oj}, e_{ol})$，分成2个问题分别求解：

(1) MIS$(G-N[v_i]$，$I\cup v_i)$；(2) MIS$(G-N[v_o]-N[v_l]$，$I\cup v_o\cup v_l)$；

5.3 若有1条互连的边(如e_{jl})，继续判断：

5.3.1 如果v_l支配v_j，分成2个问题分别求解：

(1) MIS($G-N[v_i]$，$I\cup v_i$)；

(2) MIS($G-N[v_o]-N[v_j]$，$I\cup v_o\cup v_j$)；

5.3.2 否则，分成3个问题分别求解：

(1) MIS($G-N[v_i]$，$I\cup v_i$)；

(2) MIS($G-N[v_o]-N[v_j]$，$I\cup v_o\cup v_j$)；

(3) MIS($G-N[v_o]-N[v_l]$，$I\cup v_o\cup v_l$)；

5.4 若没有互连的边，分成4个问题分别求解：

(1) MIS($G-N[v_i]$，$I\cup v_i$)；

(2) MIS($G-N[v_j]-N[v_l]$，$I\cup v_j\cup v_l$)；

(3) MIS($G-N[v_j]-N[v_o]$，$I\cup v_j\cup v_o$)；

(4) MIS($G-N[v_l]-N[v_o]$，$I\cup v_l\cup v_o$)；

6. 如果有$k(v_i)\geq 4$，继续判断：

6.1 如果存在$k(v_j)\geq 5$，分成2个问题分别求解：

(1) MIS($G-N[v_j], I\cup v_j$)；

(2) MIS($G-v_i$，I)；

6.2 否则，分成7个问题求解：

(1) 删除v_i;

(2) 删除任意2个邻居构成的组合。

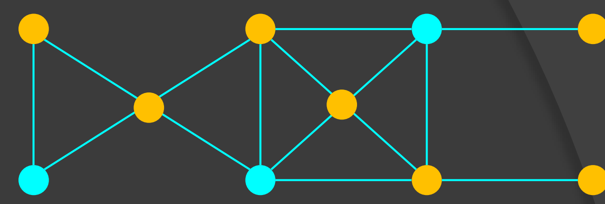
返回步骤4;

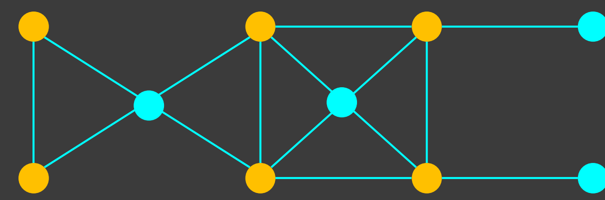
7. 比较所有极大独立集I，输出最大的I_{max}。

$O(1.285^n)$

覆盖集： 图$G=(V,E)$中，对所有$\forall e_{ij}\in E$，若$v_i\notin F$，必有$v_j\in F$，则称F是G的一个**点覆盖集**。

- 如果F的真子集不再是点覆盖集，称F是一个**极小点覆盖集**。
- F中点数最少的为**最小点覆盖集**，其点的个数记为$\beta(G)$。





$\beta=6$

点覆盖集的一些特性：

- 最小点覆盖集必为极小点覆盖集。
- 在连通图中，点覆盖集必是支配集，但反之不一定成立。
- 极小点覆盖集未必是极小点支配集。
- 如果$V-F$是$G=(V,E)$的(极大)点独立集，F必是G的(极小)点覆盖集，即$\alpha(G)+\beta(G)=|V|$。

2、网络划分、社团发现

- **网络划分（图划分）**：将网络节点划分为指定规模、数量的非重叠群组，并使得群组之间互连的边数最少。
- **社团发现**：找到网络内部不同群组之间的自然分割线。

图划分

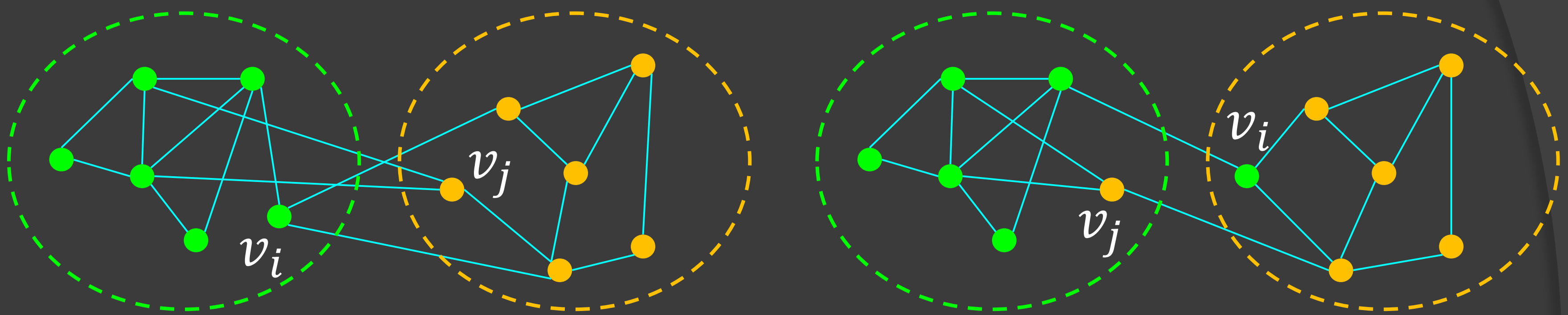
最简单的图划分就是将图对分为两部分，反复使用可将图划分为任意数量的部分。

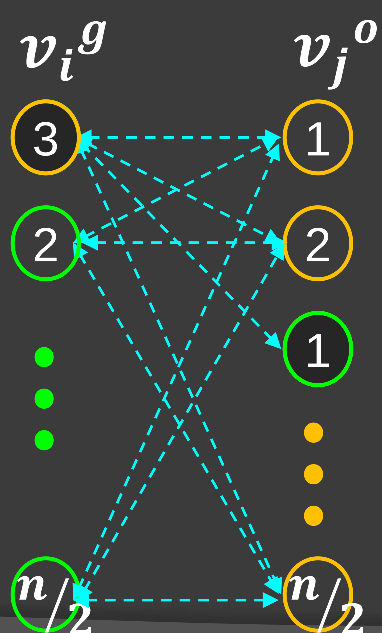
将$\boldsymbol{n}$个节点的网络划分为指定数量为n_1和n_2的两个群组，总共有$\frac{n!}{n_1!n_2!}$种组合。因$n! = \sqrt{2\pi}(^n/_e)^n$，$n = n_1 + n_2$，可得

$$\frac{n!}{n_1!n_2!} \approx \frac{\sqrt{2\pi}(^n/_e)^n}{\sqrt{2\pi}(^{n_1}/_e)^{n_1}\sqrt{2\pi}(^{n_2}/_e)^{n_2}} = \frac{n^{n+1/2}}{(n_1)^{n_1+1/2}(n_2)^{n_2+1/2}} \approx \frac{2^{n+1}}{\sqrt{n}}\text{。}$$

Kernighan-Lin算法

求解思想：将图的所有点随机分为两个组，然后，对调两组中的一对点，通过组间割集规模判断每个点更适合哪个组。





从绿色组中选v_i^g，i从1循环到$^n/_2$。

- 依次与橙色组中的$v_j^o(j=1,2,\cdots,{}^n/_2)$对调，并计算组间边减少的数量；$v_i^g$与使边减少得最多的$v_j^o$交换组；注意：交换过的点不再参与对调比较。

$O(mn^2)$

谱划分

定义图$G=(V,E)$的**拉普拉斯矩阵**：$L_{ij}=\begin{cases} k_i, & i=j \\ -1, & i\neq j \text{且} e_{ij}\in E。 \\ 0, & \text{其他} \end{cases}$

令$\delta_{ij}=\begin{cases} 1, & i=j \\ 0, & i\neq j \end{cases}$，$\boldsymbol{D}=\begin{bmatrix} k_1 & 0 & \cdots & \\ 0 & k_2 & 0 & \cdots \\ \vdots & 0 & \ddots & \\ & \vdots & & \end{bmatrix}$(度矩阵)，则有

$L_{ij}=\delta_{ij}k_i-a_{ij}$ ，矩阵形式为 $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$。

拉普拉斯矩阵的性质：

1）对$\boldsymbol{n}$维向量$\boldsymbol{x}\in R^n$，有$\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}=\lambda x x^T=\frac{1}{2}\sum_{ij}a_{ij}(x_i-x_j)^2$。

2）$\boldsymbol{L}$是半正定对称矩阵,特征值为非负实数$\lambda_1\leq\lambda_2\leq\cdots\leq\lambda_n$。

3）$\boldsymbol{L}$的最小特征值=0，对应特征向量 $\boldsymbol{1}=(1,\cdots,1)$，$\boldsymbol{L}\cdot\boldsymbol{1}=0$。

4）非负权值图的分支个数与该图的$\boldsymbol{L}$的0特征值个数相等。

5）连通图的$\boldsymbol{L}$的第二小(Fiedler)特征向量的特征值$\lambda_1\leq\lambda_2\leq\cdots\leq\lambda_i$。

图$G=(V,E)$的点分为绿色点V^g和橙色点V^o两组，割集的边数

$R=\frac{1}{2}\sum a_{ij}(v_i$与$v_j$不同组)。设$s_i=\begin{cases}+1, & v_i\in V^g\\ -1, & v_i\in V^o\end{cases}$，可推得

$\frac{1}{2}(1-s_is_j)=\begin{cases}1, & v_i\text{和}v_j\text{在不同组}\\ 0, & v_i\text{和}v_j\text{在同一组}\end{cases}$，$R=\frac{1}{4}\sum_{ij}a_{ij}(1-s_is_j)$；

$\sum_{ij}a_{ij}=\sum_i k_i {s_i}^2=\sum_{ij}k_i\delta_{ij}s_is_j$，$R=\frac{1}{4}\sum_{ij}(\delta_{ij}k_i-a_{ij})\,s_is_j$；

矩阵形式：$R=\frac{1}{4}\boldsymbol{s}^T\boldsymbol{L}\boldsymbol{s}$，问题转化为为求$\min_s\{R(s)\}$。

设$\vec{\mathbf{v}}$为$\boldsymbol{L}$的一个归一化特征向量，有$\vec{\mathbf{v}}^T\vec{\mathbf{v}}=1$；设$\boldsymbol{c}$为一常向量，

且满足$\boldsymbol{s}=\boldsymbol{c}\vec{\mathbf{v}}$，$\boldsymbol{s}^T=\boldsymbol{c}^T\vec{\mathbf{v}}^T$。因$\boldsymbol{c}^T\boldsymbol{c}=s^T\boldsymbol{s}=\sum_i {s_i}^2=n$，则有

$\boldsymbol{R}=\frac{\mathbf{1}}{\mathbf{4}}\boldsymbol{c}^T\vec{\mathbf{v}}^T\boldsymbol{L}\vec{\mathbf{v}}\boldsymbol{c}=\frac{\mathbf{1}}{\mathbf{4}}\lambda\boldsymbol{c}^T\boldsymbol{c}\vec{\mathbf{v}}^T\vec{\mathbf{v}}=\frac{n}{4}\lambda$，$\boldsymbol{R}$与$\boldsymbol{L}$的特征值成正比。

$\boldsymbol{R}$最小，应选择$\frac{n}{4}\lambda=\mathbf{0}$，但其对应的特征向量为$\mathbf{1}=(\mathbf{1},\cdots,\mathbf{1})$，

若令$\boldsymbol{s}=(\mathbf{1},\cdots,\mathbf{1})$则无法依据$\boldsymbol{s}$进行图划分，故$\lambda$为无效解。

选取L的Fiedler向量$\vec{v}^f$作为一个合适的优化解。因$s_i = \pm 1$，所以s与$\vec{v}^f$不平行。与$\vec{v}^f$最近似的s应尽量使两者的乘积最大，$(\vec{v}^f)^T s = \sum_i s_i \vec{v}_i^f$。另外，$\vec{v}^f$中正负值的个数与$s_i$中正负1的个数也会不相等，故可分别按照降序和升序的方式排列$\vec{v}^f$值，再将前面$|V^g|$个对应的点划分给V^g，剩余的划入V^o；以两种排序方式获得的割集规模较小者为最终结果。

基于普划分的图对分算法： $O(mn)$

- 根据输入的图的点对关系，构造邻接矩阵A和度矩阵D；
- 依据A和D计算图的拉普拉斯Laplacian矩阵$L = D - A$；
- 计算L的第二小特征值及其对应的Fiedler特征向量$\vec{v}^f$；
- 对$\vec{v}^f$值进行排序，根据尺寸要求进行点的分割；
- 比较两种分割结果，将较好的结果映射到原始图中。

社团发现

设图的边数$=m$，c为点的类型，$\delta_{ij}{}^c = \begin{cases} 1，点i和j同属c类 \\ 0，其他 \end{cases}$，则同类点之间的总边数$=\frac{1}{2}\sum_{ij} a_{ij}\,\delta_{ij}{}^c$。随机连接条件下同类点之间总边数的期望值$=\frac{1}{2}\sum_{ij}\frac{k_i k_j}{2m}\,\delta_{ij}{}^c$。同类点之间边连接程度的**模块度**$\boldsymbol{Q=\frac{1}{2m}\sum_{ij}\left(a_{ij}-\frac{k_i k_j}{2m}\right)\delta_{ij}{}^c}$即网络同配性的度量指标。

简单模块度最大化方法：

- 将网络随机地划分为两个群组，分别分配两个类型值；

迭代：

$\boldsymbol{O(n^2)}$

- 依次计算每个点移到另一群组后两组的模块度；
- 将使对方组模块度增加最多的或本组模块度减少最少的那个点移到另一组，并标记该节点（每个点仅移动一次）。

谱模块度最大化方法：

定义模块度矩阵$b_{ij}=a_{ij}-\frac{k_i k_j}{2m}$，则有$Q=\frac{1}{2m}\sum_{ij}b_{ij}{\delta_{ij}}^c$。

引入谱划分中的$\boldsymbol{s_i}$，有${\delta_{ij}}^c=\frac{1}{2}\left(1+s_i s_j\right)$，又因$\sum_j b_{ij}=0$，

有$Q=\frac{1}{4m}\sum_{ij}b_{ij}\left(s_i s_j+1\right)=\frac{1}{4m}\sum_{ij}b_{ij}s_i s_j$，即$\boldsymbol{Q=\frac{1}{4m}s^T B s}$。

社团发现问题为：寻找合适的向量$\boldsymbol{s}$，使得$\max_s\{\,Q(s)\,\}$。

类似谱划分中的推导，可知$\boldsymbol{s}$应选择最大特征值λ_n对应的$\vec{\mathbf{v}}^n$。

因$\boldsymbol{s_i=\pm 1}$，应使$(\vec{\mathbf{v}}^n)^T\boldsymbol{s}=\sum_i \boldsymbol{s_i}\overrightarrow{\mathbf{v}_i}^n$最大化，又因组无尺寸限制，

所以，$s_i=\begin{cases}+1, & \overrightarrow{\mathrm{v}_i}^n\geq 0\\ -1, & \overrightarrow{\mathrm{v}_i}^n\leq 0\end{cases}$。

$\boldsymbol{O(n^2)/O(n^3)}$

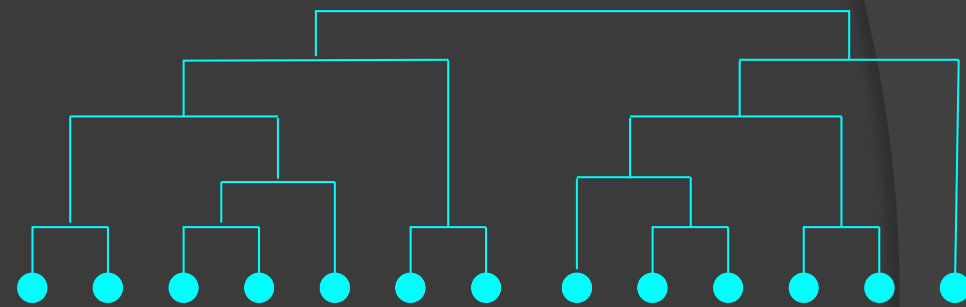
对多组中的组$\boldsymbol{c}$，有${b_{ij}}^{(c)}=b_{ij}-{\delta_{ij}}^c\sum_{l\in c}b_{il}$，$\boldsymbol{\Delta Q=\frac{1}{4m}s^T B^{(c)}s}$。

若无法找到使$\boldsymbol{\Delta Q>0}$的网络划分，则社团不再被划分。

基于介数的方法：

迭代　　$O(mn(m+n))$

- 计算网络中所有边的介数，寻找并删除最大介数的边；
- 重新计算边的介数，继续迭代，直至网络划分为单个节点。



层次聚类的方法：

层次聚类是一种合并技术，基于网络结构定义一种点之间的相似性（余弦相似性、相关系数、欧几里得距离等）或者点之间联通强度测度，然后将最接近或者最相似的点合并成群组。

问题：不同相似性测度会有不同的分组，且无法判断那个更好。

作业：

33、求背包问题的最优解属于NPC问题，其蛮力算法如下：

输入：背包最大承重量c，物品个数n，单个物品重量w[n]，价值v[n]。

输出：装在背包的物品以及产生的最大价值。

1. 初始化最大价值$V_{max}=0$，结果子集$s=\phi$；
2. 循环，对集合{1, 2, …, n}的每个排列T情况，执行：
   - 2.1 初始化背包的价值v=0，背包的重量w=0;
   - 2.2 循环，对T的每个元素j:
     - 2.2.1 如果$w+w_j<c$，则$w=w+w_j$、$v=v+v_j$;
     - 2.2.2 否则，跳出循环，转步骤2.3;
   - 2.3 如果$V_{max}<v$，则$V_{max}=v$、s=T，转步骤2;
3. 输出s中的元素。

写出该算法计算复杂度的数学分析表达式。

34、图4中，边的权值代表链路容量，以源到目的节点链路容量最大化为优化目标。基于Dijkstra算法计算以a为源到所有节点的的最短路径树，写出完整的计算过程。

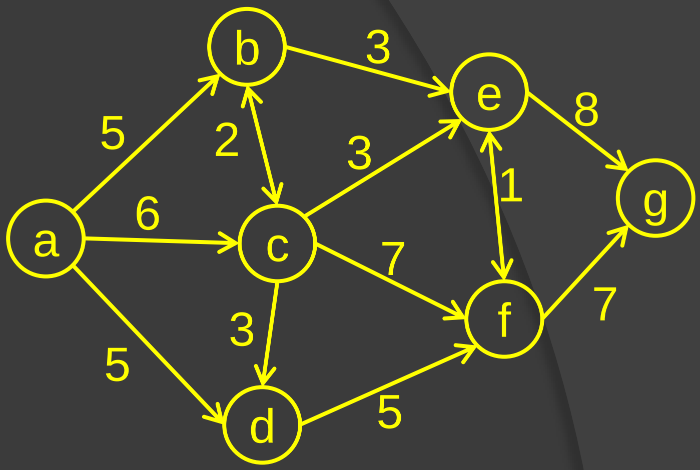


图4

35、上题图4中的箭头代表着链路上允许数据流动的方向，基于Ford-Fulkerson计算点a到点g的最大流。

36、基于Ford-Fulkerson方法（增广路径）来进行不相交路径的搜索的主要好处是什么？

37、在图4中选出2条点a到点g的并发路径，用以分流，达到负载均衡的目的。说说你的选择的理由。

38、Kruskal算法和Prim算法适用的条件有什么不同？

39、贪婪算法的特点是“以单步搜索的最优去逼近全局最优”。参照“贪婪”思路，PPT课件中关于平面Steiner最小树的启发式算法应做怎样的修改？修改后的算法复杂度是多少？怎候可以使选点的选取范围更小一些呢?

40、图5是一通信网络，其中橙色节点是一组播组。基于KMB算法计算该组的Steiner路由树。

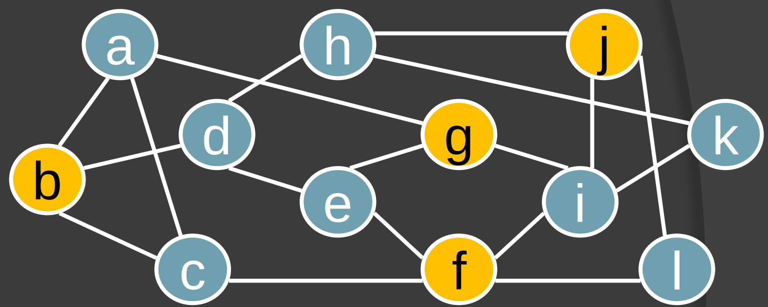


图5

41、无线传感网常常采用组簇方式的层次路由机制。普通传感节点将检测数据以单跳近距离通信方式传给簇头，簇头再将数据中继

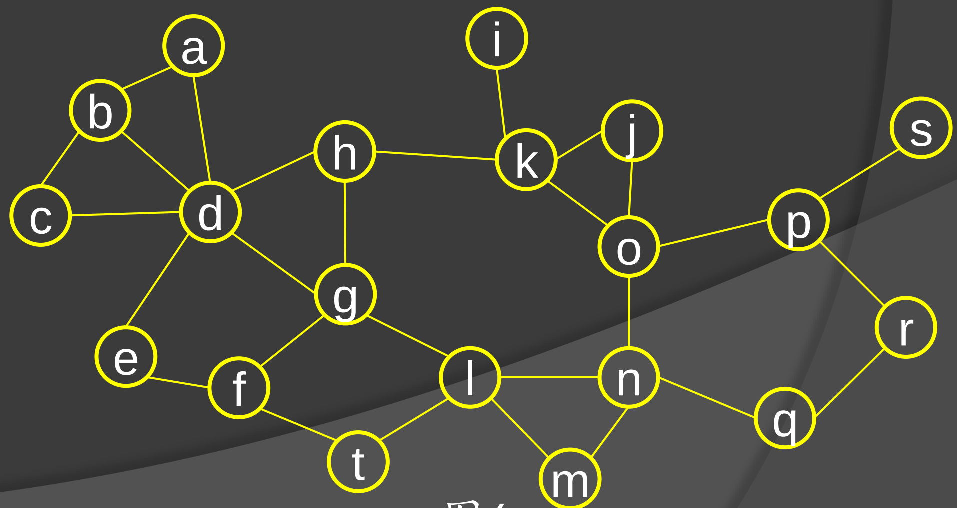


图6

给连接互联网的网关。所以，簇头既具备传感节点的功能，又具备远距离通信的路由功能，但其价格要比普通传感设备贵很多。图6是一传感网的一跳邻居互连拓扑图，怎样部署路由节点既满足应用需要，又使建设成本最低。

42、地图着色问题(相邻区域不同色)可以借助图论中的一些方法来解决。结合图7给出的地图，谈一谈你的解题思路。用了几种颜色？

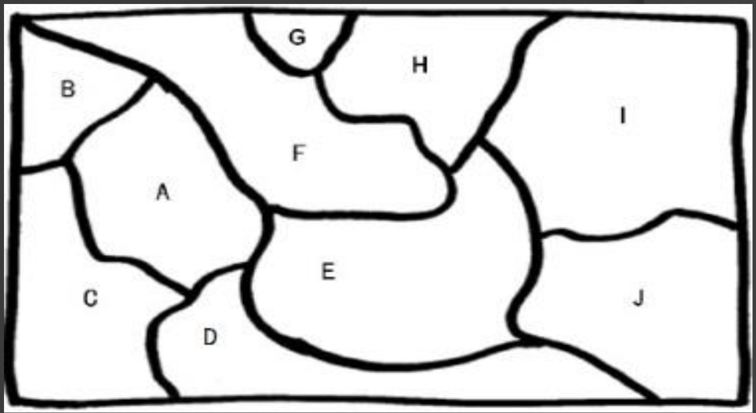


图7

43、“图划分”与“社团发现”有什么不同？Kernighan-Lin算法的思路是怎样的？与其相比，试论述“谱方法”中采用了怎样的方法来降低计算复杂度？

44、你在网上搜索一下，最新的关于社团发现的研究有哪些？