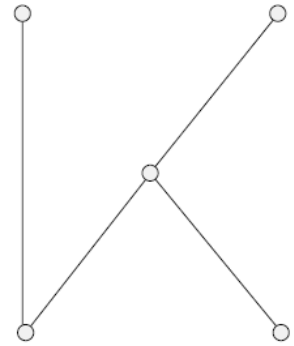
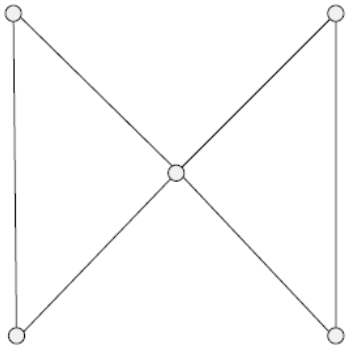
第三讲 图的连通性

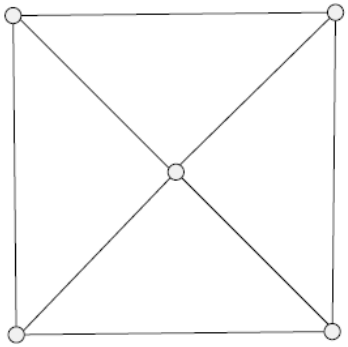
如何刻画一个图的连通程度？



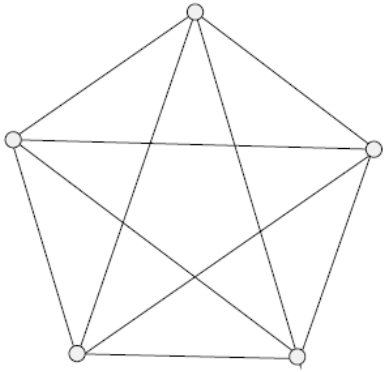
G_1



G_2



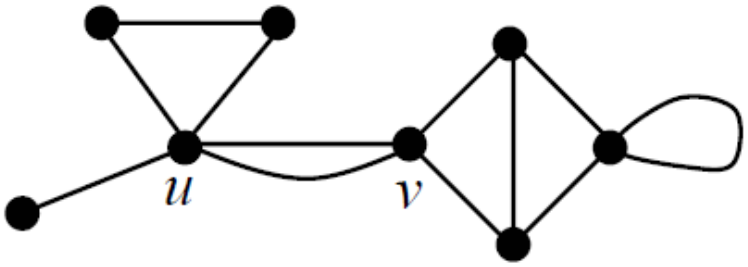
G_3



G_4

§2.1 割点和割边

定义 2.1.1 设 $v \in V(G)$，如果 $w(G-v) > w(G)$，则称 v 为 G 的一个**割点**。



定理 2.1.1 如果点 v 是简单图G 的一个割点，则边集 E(G)可划分为两个非空子集 E_1 和E_2 ，使得$G[E_1]$ 和 $G[E_2]$ 恰好有一个公共顶点v。

推论 2.1.1 对连通图 G，顶点 v 是 G 的割点当且仅当 $G-v$ 不连通。

定理 2.1.2 设 v 是树 T 的顶点，则 v 是 T 的割点当且仅当$d(v) > 1$。

推论2.1.2 每个非平凡无环连通图至少有两个顶点不是割点。

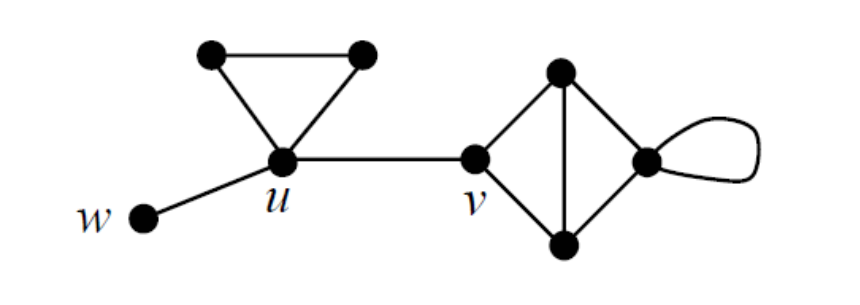
定理 2.1.3 设 v 是连通图G 的一个顶点，则下列命题等价：

（1） v 是G 的割点；

（2） 存在$u,w \in V(G)$，使得$u,w \neq v$且v 在每条(u,w)路上；

（3） 存在$V(G) \setminus \{v\}$的一个划分：$V(G) \setminus \{v\} = U \cup W$，$U \cap W = \varnothing$，使得对$\forall u \in U$和$\forall w \in W$，$v$ 在每条(u,w)路上。

定义 2.1.2 设$e \in E(G)$，如果$w(G-e) > w(G)$，则称e为G的一条**割边**。



定理 2.1.4 边 e 是 G 的割边当且仅当 e 不在 G 的任何圈中。

证明：证其逆否命题：e 不是割边当且仅当 e 含在 G 的某个圈中。

必要性：设 $e = xy$ 不是割边。假定 e 位于 G 的某个连通分支G_1中，则 $G_1 - e$ 仍连通。故在 $G_1 - e$ 中有 (x,y) 路 P，$P + e$ 便构成 G_1 中一个含有 e 的圈。

充分性：设 e 含在 G 的某个圈 C 中，而 C 含于某连通分支 G_1 中，则 $G_1 - e$ 仍连通。故 $w(G - e) = w(G)$，这说明 e 不是割边。证毕。

定理 2.1.5 一个连通图是树当且仅当它的每条边都是割边。

定理 2.1.6 设 e 是连通图 G 的一条边，则下列命题等价：

（1） e 是 G 的割边；

（2） e 不在 G 的任何圈上；

（3） 存在$u,v \in V(G)$，使得 e 在每条 (u,v)路上；

（4） 存在 $V(G)$的一个划分： $V(G) = U \cup W$ ， $U \cap W = \phi$ ，使得对 $\forall u \in U$ 和 $\forall w \in W$ ， e 在每条 (u , w)路上。

§2.2 连通度和边连通度

定义 2.2.1 对图 G，若V(G)的子集V'使得$w(G-V')>w(G)$，则称V'为图G的一个**顶点割集**。含有 k 个顶点的顶点割集称为**k**—顶点割集。

注：（1）割点是1—顶点割集。
（2）完全图没有顶点割集。

定义 2.2.2 图G 的**连通度**定义为

$$\kappa(G) = \min\{|V'| \mid V' \text{是连通图G 的顶点割集}\}。$$

特别地，完全图的连通度定义为$\kappa(Kn)=n-1$；

不连通图的连通度定义为0。

注： (1) 若G是平凡图，则$\kappa(G)=0$。

(2) 使得$|V'|=\kappa(G)$的顶点割集V'称为G的最小顶点割集。

(3) 若$\kappa(G)\geq k$，则称G为 ***k*** 连通的。

(4) 按上述定义，图 G 是 k 连通的，**当且仅当**G 的最小点割集至少含k个顶点，**当且仅当**G 中没有$k-1$ 点割集，**当且仅当**从G 中任意去掉$k-1$ 个顶点后，所剩图仍连通。

(5) 按照k-连通的定义，若图G是 k 连通的，则它也是$k-1$ 连通、$k-2$ 连通、…、1 连通的。因此，所有非平凡连通图都是1 连通的。

定义 2.2.3 对图G，若E(G)的子集E'使得$w(G-E')>w(G)$，则称E'为图G 的一个**边割集**。
含有k 条边的边割集称为**k－边割集**。

注：(1) 对非平凡图G，若E'是一个边割集，则$G\setminus E'$不连通。

(2) 一条割边构成一个1－边割集。

(3) 设$S\subset V(G)$，$S'\subset V(G)$，$S,S'\neq\emptyset$，记号$[S,S']$表示一端在S中另一端在S'中的所有边的集合。对图G的每个边割集E'，必存在非空的$S\subset V(G)$，使得$[S,\overline{S}]$是G的一个边割集，其中$\overline{S}=V\setminus S$。

定义 2.2.4 图G的边连通度定义为

$\kappa'(G) = \min\{|E'| \mid E'$是连通图G的边割集$\}$。

特别地，完全图的边连通度定义为$\kappa'(Kn)=n-1$；

不连通图的边连通度定义为0。

注：(1) 对平凡图G，$\kappa'(G) = 0$。

(2) G是含有割边的连通图，则$\kappa'(G) = 1$。

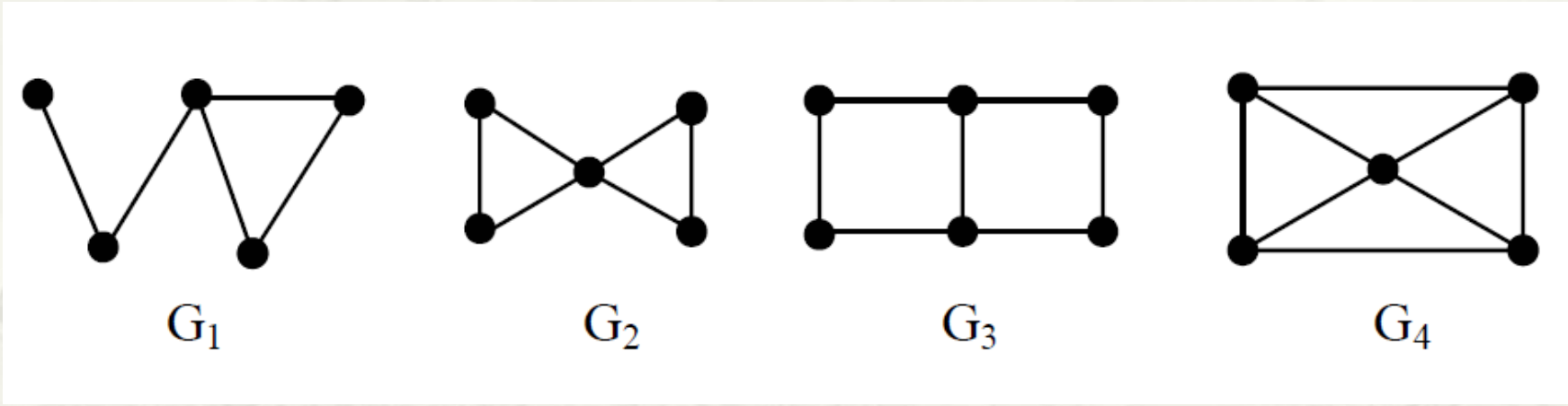
(3) 使得$|E'|=\kappa'(G)$的边割集E'称为G的最小边割集。

(4) 若$\kappa'(G) \geq k$，则称G为***k*** 边连通的。

(5) 按上述定义，图G 是k 边连通的，**当且仅当**G 的最小边割集至少含k 条边，**当且仅当**G 中没有$k-1$ 边割集，**当且仅当**从G 中任意去掉$k-1$ 条边后，所剩图仍连通。

(6) 按照k 边连通的定义，若图G 是k 边连通的，则它也是$k-1$ 边连通、$k-2$ 边连通、…、1 边连通的。因此，所有非平凡连通图都是1 边连通的。

下列图的最小点割集和最小边割集分别是什么？是几连通的？



定理 2.2.1 $\kappa(G) \le \kappa'(G) \le \delta(G)$ 。

证明：先证 $\kappa(G) \le \kappa'(G)$ 。

对图的边连通度 $\kappa'(G)$ 作数学归纳法。

对 $\kappa'(G) = 1$ 的图 G，若 $G = K_2$，则显然 $\kappa'(G) = v - 1 = 1$；若 $G \ne K_2$，则 G 至少 含三个点。设 $e = uv$ 是 G 的一条割边，则 u 或 v 必是割点，故 $\kappa(G) = 1$。

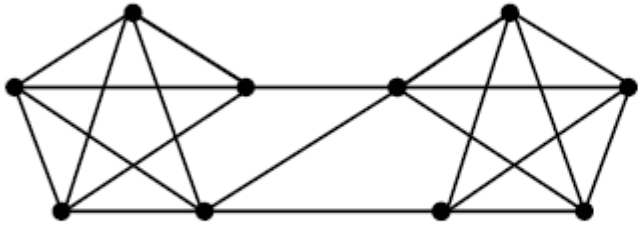
总之，此时 $\kappa(G) = \kappa'(G) = 1$ 。

假设对所有 $\kappa' = k$的图，都有 $\kappa' \le \kappa$，则对 $\kappa'(G) = k + 1$ 的图G，设E 是它的一个 $k + 1$ 边割集。任取边 $e = uv \in E(G)$，则 $E - e$ 是$G - e$ 的最小边割集，故 $\kappa'(G - e) = k$ 。由归 纳假设， $\kappa(G - e) \le \kappa'(G - e)$ 。取$G - e$ 的最小点割集 T，则

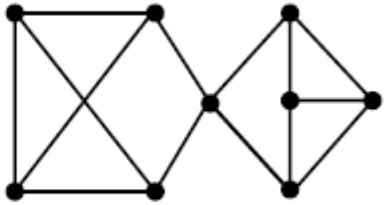
$$|T| = \kappa(G - e) \le \kappa'(G - e) = k ,$$

且 $T \cup \{u\}$ 构成 G 的最小点割集。故 $\kappa(G) = |T \cup \{u\}| \le |T| + 1 \le k + 1 = \kappa'(G)$ 。归纳完成。

下列图的连通度、边连通度和最小度分别是多少？



G_1



G_2

定理 2.2.2 对具有 v 个顶点 ε 条边的连通图 G，有$\kappa(G) \leq \left\lfloor \frac{2\varepsilon}{v} \right\rfloor$。

证明：因$2\varepsilon = \sum_{v \in V(G)} d(v) \geq \delta v$，故$\delta \leq \frac{2\varepsilon}{v}$，由定理 2.2.1，$\kappa \leq \delta \leq \frac{2\varepsilon}{v}$。

由于κ 是整数，因此$\kappa \leq \left\lfloor \frac{2\varepsilon}{v} \right\rfloor$。证毕。

定理 2.2.3 设 G 是一个简单图，k 是一个自然数，若 $\delta(G) \geq \frac{v+k-2}{2}$，则 G 是 k 连通的。

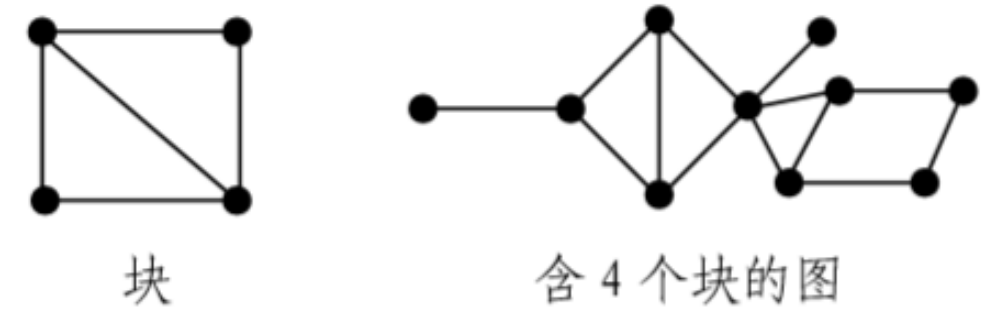
证明：用反证法。假如 G 不是 k 连通的，则 G 的连通度 $\kappa < k$，即存在 G 的点割集 S，使得 $|S| < k$，且 G−S 不连通。因 G −S 有 $v - |S|$ 个顶点，且至少有两个连通分支，故必有 G−S 的某个连通分支 G′ 含有不超过 $\frac{v-|S|}{2}$ 个顶点。注意到 G′ 中任一个顶点只可能与 G′ 内的点及 S 中的点相邻，因而其在 G 中的顶点度 $\leq \frac{v-|S|}{2} - 1 + |S| = \frac{v+|S|-2}{2}$。结合 $|S| < k$，这意味着 $\delta(G) \leq \frac{v+|S|-2}{2} < \frac{v+k-2}{2}$，与定理条件矛盾。证毕。

推论 2.2.1 设 G 是一个简单图，若 $\delta(G) \geq \frac{v-1}{2}$，则 G 是连通的。

§2.3 2-连通图的性质

定义2.3.1 无割点的连通图称为一个块(block)。设G是一个图，H是G的一个子图，若H本身是一个块且它是G中具有此性质的极大子图，则称H是图G的一个块。

下面是块及图的块的例子。



注：至少有三个顶点的图是块当且仅当它是2—连通图。（若只有两个顶点，则有反例，例如K_2是个块，但不是2连通的。）

关于块的部分等价命题总结在下一个定理中。

定理 2.3.2 设 G 是 $v \geq 3$ 的连通图，则下列命题等价：

（1） G 是 2 连通的（块）；

（2） G 的任二顶点共圈；

（3） G 的任一顶点与任一边共圈；

（4） G 的任二边共圈；

（5） 对 $\forall u, v \in V(G)$ 及 $\forall e \in E(G)$，存在 (u, v) 路含有边 e；

（6） 对 $\forall u, v, w \in V(G)$，存在 (u, v) 路含有顶点 w；

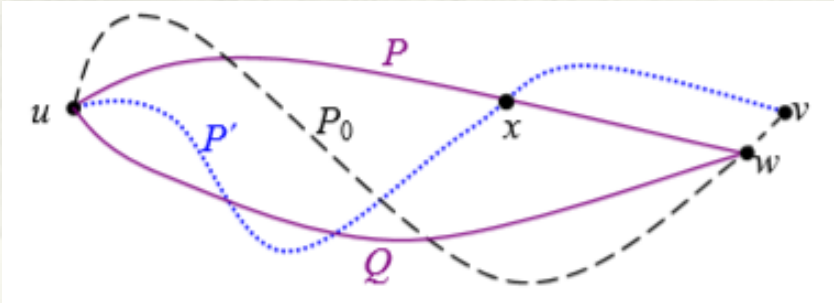
（7） 对 $\forall u, v, w \in V(G)$，存在 (u, v) 路不含有顶点 w。

对距离$d(u,v)$作归纳法。

$d(u,v)=1$时，因$\kappa' \geq \kappa \geq 2$，故$G$中无割边，$G-uv$仍连通。因此$G-uv$中存在一条$(u,v)$路$P_1$。这表明在$G$中$u,v$都在圈$P_1+uv$上。

假定$d(u,v)<k$时，结论成立。下证$d(u,v)=k$时结论也成立。

当$d(u,v)=k$时，设$P_0=u\cdots wv$是长为k的一条(u,v)路，则$d(u,w)=k-1$。由归纳法假设，u,w在同一圈上，故在u,w间有两条无公共内部顶点的路P和Q。因G是2连通图，故$G-w$仍连通。在$G-w$中存在(u,v)路P'。令x是P'上最后一个与$P \cup Q$的公共顶点（因$u \in P \cup Q$，这样的x存在）。不妨设$x \in P$，则P上(u,x)段$+P'$上(x,v)段与$Q+wv$是两条内部无公共点的(u,v)路。故u,v在同一圈上。归纳法完成。证毕。



§2.4 割点的求解算法

1. 问题的提出

对一个由无向图表示的网络，将顶点看做通讯站，无向边看做两个通讯站之间可以相互通讯。问:

(1) 若其中一个站点出现故障，是否会影响系统工作;

(2) 该通讯网可以分成哪几个含站点尽可能多的子网，这些子网内的任一站点出现故障，都不会影响子网工作。

2. 割点的蛮力搜索方法

蛮力法的原理就是通过定义求解割点。在图中去掉某个顶点，然后进行DFS遍历，如果连通分支数增加，那么该顶点就是割点。对每个顶点进行一次上述操作，就可以求出这个图的所有割点，我们称之为这个图的割点集。

在具体的代码实现中，并不需要真正删除该顶点和删除依附于该顶点所有边。需要在DFS前，将该顶点对应是否已访问的标记置为true，然后从其它顶点为根进行DFS即可。

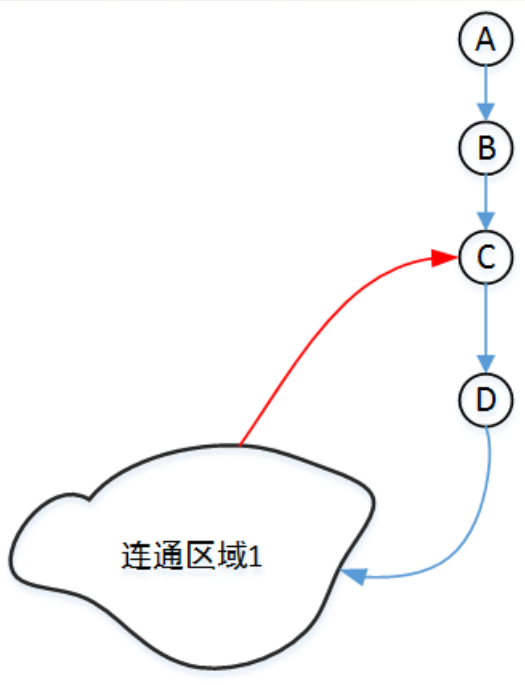
3. Tarjan算法

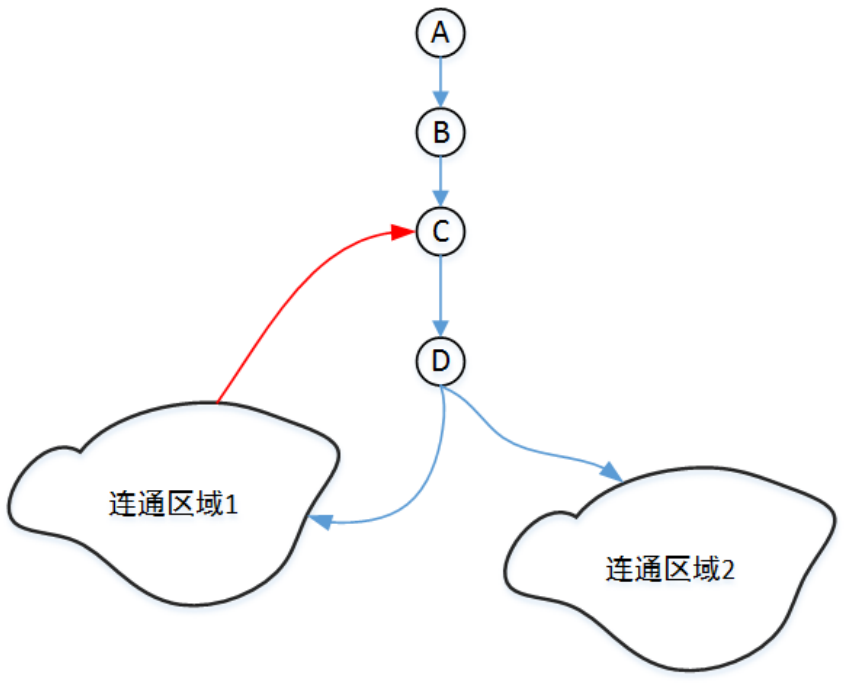
3.1 Tarjan算法原理

概念：父节点、孩子节点、祖先节点

在DFS中从顶点U访问到了顶点V（此时顶点V还未被访问过），则称顶点U为顶点V的**父顶点**，V为U的**孩子顶点**。在顶点U之前被访问过的顶点，称为U的**祖先顶点**。

Robert Tarjan
1986年获得图灵奖





如果顶点U的所有孩子顶点可以不通过父顶点U而访问到U的祖先顶点，那么说明此时去掉顶点U不影响图的连通性，U就不是割点。相反，如果顶点U至少存在一个孩子顶点，必须通过父顶点U才能访问到U的祖先顶点，那么去掉顶点U后，顶点U的祖先顶点和孩子顶点就不连通了，说明U是一个割点。

3.2 Tarjan算法实现

(1) dfn[] 数组

dfn数组的下标表示顶点的编号，数组中的值表示该顶点在DFS中的遍历顺序，每访问到一个未访问过的顶点，访问顺序的值就增加1。**子顶点的dfn值一定比父顶点的dfn值大**（但不一定恰好大1）。在访问一个顶点后，它的dfn的值就确定下来了，不会再改变。

(2) low[] 数组

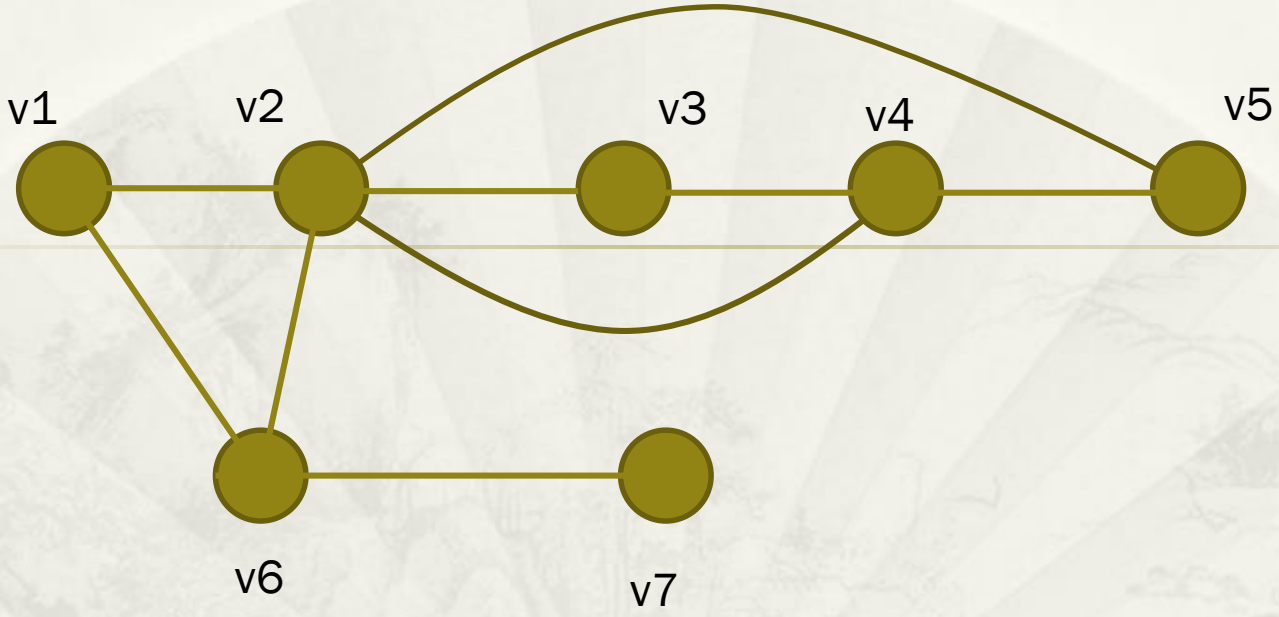
low数组的下标表示顶点的编号，数组中的值表示DFS中该顶点**不通过**父顶点能访问到的祖先顶点中最小的顺序值。每个顶点初始的low值和dfn值应该一样，在DFS中，根据情况不断更新low的值。

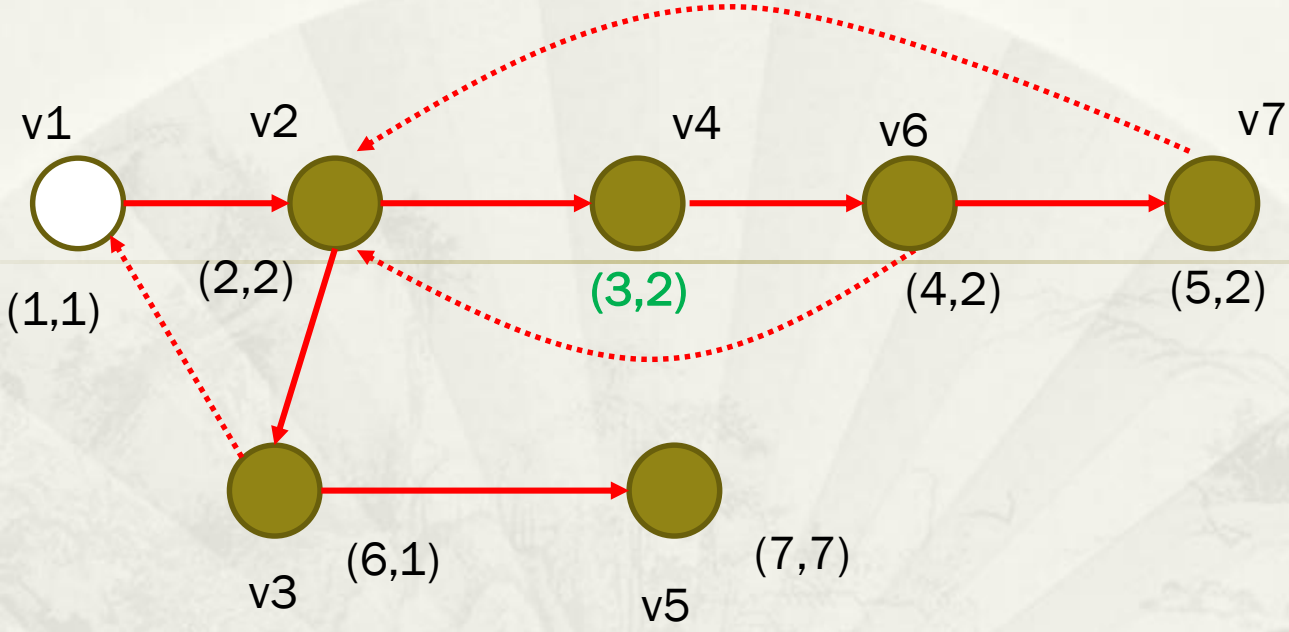
假设由顶点u访问到顶点v。当从顶点v回溯到顶点u时，

如果 dfn[v] < low[u], 那么令 low[u] = dfn[v]

如果顶点U还有它分支，每个分支回溯时都进行上述操作，那么顶点low[u]就表示了不通过顶点U的父节点所能访问到的最早祖先节点。

3.3 例子




v1
v2
v4
v6
v7
(1,1)
(2,2)
(3,2)
(4,2)
(5,2)
v3
(6,1)
v5
(7,7)

3.4 割点的判断方法

（1）若u不是根顶点，用u顶点的dnf值和它的所有的孩子顶点的low值进行比较，如果存在至少一个孩子顶点v满足low[v]>= dnf[u]，就说明顶点v访问顶点u的祖先顶点，必须通过顶点u，而不存在顶点v到顶点u祖先顶点的其它路径，所以顶点u就是一个割点。对于没有孩子顶点的顶点，显然不会是割点。

（2）若u是根顶点，则u是割点当且仅当u至少含有两个孩子顶点。

§2.5 连通度的求解算法

5.1 分离集

设 G 是一个简单图，x, y 是G 中任二不同顶点。如果从G 中删去一组顶点后不再有(x, y)路，则称这组顶点**分离x和y**，且称这组顶点为一个**(x, y)分离集**。G 中含点数最少的(x, y)分离集称为**最小(x, y)分离集**，其顶点数称为G 的**(x, y)分离数**，记为$s(x, y)$。此外，将G 中两两内部点不交的(x, y)路的最大条数记为$r(x, y)$。

5.2 Menger定理

定理 2.4.1（Menger, 1927）对任意图 G，设 x, y 是 G 中两个不相邻顶点，则 G 中分离 x, y 所需的最少顶点数等于 G 中两两内部点不交的 (x, y)路的最大条数，即

$$s(x, y) = r(x, y)。$$

推论 2.4.2 无向图 G 的连通度与顶点间内部不交路的最大条数之间存在如下关系：

$$\kappa(G)=\begin{cases}|V(G)|-1 & \text{若}G\text{是完全图}\\ \min\limits_{xy\notin E(G)}\{r(x,y)\} & \text{若}G\text{不是完全图}\end{cases}$$

5.3 求解连通度算法步骤

输入：无向图G；输出： G的连通度 $\kappa(G)$

步骤1：设 A= {(x,y)| x,y是G的不相邻顶点} ；

步骤2：置$\kappa(G) \leftarrow \infty$；

步骤3：任取(x,y)∈A，用最大流的方法求出 r(x,y) 和对应的割点集；

步骤4：若r(x,y) $<\kappa(G)$，则$\kappa(G) \leftarrow$r(x,y) ，并保存其割点集H；

步骤5：令A←A - {(x,y)} ；

步骤6：若A=∅，则算法停止，输出$\kappa(G)$ ，H即为最小割点集；否则，转步骤3。

最大流算法

Method	Complexity	Description
Linear programming		Constraints given by the definition of a legal flow. See the linear program here.
Ford–Fulkerson algorithm	$O(E \max\|f\|)$	As long as there is an open path through the residual graph, send the minimum of the residu The algorithm is only guaranteed to terminate if all weights are rational. Otherwise it is possil algorithm terminates, it is guaranteed to find the maximum value.
Edmonds–Karp algorithm	$O(VE^2)$	A specialization of Ford–Fulkerson, finding augmenting paths with breadth-first search.
Dinic's blocking flow algorithm	$O(V^2E)$	In each phase the algorithms builds a layered graph with breadth-first search on the residual and the maximum number of the phases is n-1. In networks with unit capacities, Dinic's algor
MPM (Malhotra, Pramodh-Kumar and Maheshwari) algorithm[9]	$O(V^3)$	Only works on acyclic networks. Refer to the Original Paper.
Dinic's algorithm	$O(VE\log(V))$	The dynamic trees data structure speeds up the maximum flow computation in the layered g
General push-relabel maximum flow algorithm	$O(V^2E)$	The push relabel algorithm maintains a preflow, i.e. a flow function with the possibility of exc excess, i.e. an active vertex in the graph. The push operation increases the flow on a residual be pushed. The height function is changed with a relabel operation. The proper definitions of
Push-relabel algorithm with *FIFO* vertex selection rule	$O(V^3)$	Push-relabel algorithm variant which always selects the most recently active vertex, and perfc edges from this vertex.
Push-relabel algorithm with dynamic trees	$O\left(VE\log\frac{V^2}{E}\right)$	The algorithm builds limited size trees on the residual graph regarding to height function. Th
KRT (King, Rao, Tarjan)'s algorithm[10]	$O(EV\log_{\frac{E}{V\log V}} V)$	
Binary blocking flow algorithm[11]	$O\left(E\cdot\min(V^{\frac{2}{3}},\sqrt{E})\cdot\log\frac{V^2}{E}\log U\right)$	The value U corresponds to the maximum capacity of the network.

Algorithm Ford－Fulkerson

Inputs Given a Network $G = (V, E)$ with flow capacity c, a source node s, and a sink node t

Output Compute a flow f from s to t of maximum value

1. $f(u, v) \leftarrow 0$ for all edges (u, v)
2. While there is a path p from s to t in G_f, such that $c_f(u, v) > 0$ for all edges $(u, v) \in p$:
    1. Find $c_f(p) = \min\{c_f(u, v) : (u, v) \in p\}$
    2. For each edge $(u, v) \in p$
        1. $f(u, v) \leftarrow f(u, v) + c_f(p)$ (*Send flow along the path*)
        2. $f(v, u) \leftarrow f(v, u) - c_f(p)$ (*The flow might be "returned" later*)

Algorithm Max-Flow Min-Cut (MFMC)

INPUT: a network $N := N(x, y)$ and a feasible flow f in N
OUTPUT: a maximum flow f and a minimum cut $\partial^+(X)$ in N

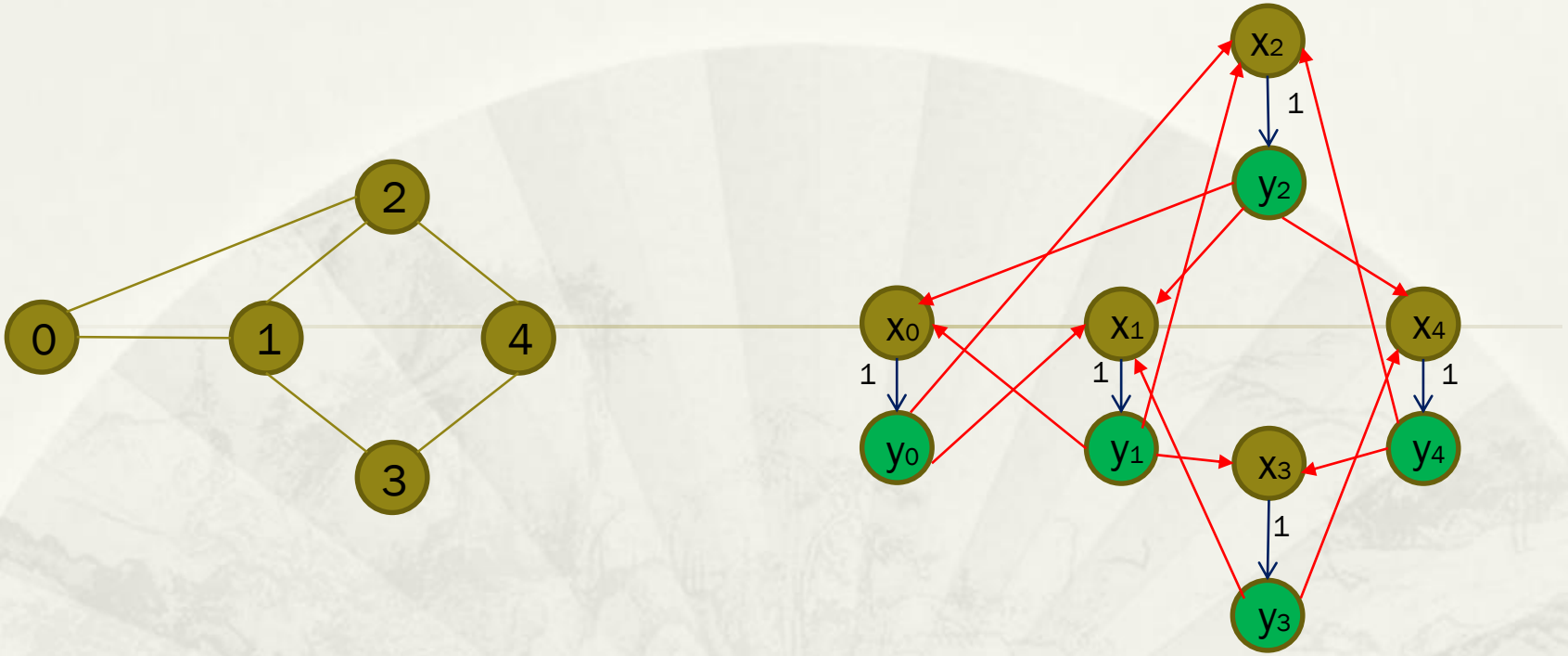
1: *set* $X := \{x\}$, $p(v) := \emptyset$, $v \in V$
2: ***while*** *there is either an* f*-unsaturated arc* $a := (u, v)$ *or an* f*-positive arc* $a := (v, u)$ *with* $u \in X$ *and* $v \in V \setminus X$ ***do***
3: *replace* X *by* $X \cup \{v\}$
4: *replace* $p(v)$ *by* u
5: ***end while***
6: ***if*** $y \in X$ ***then***
7: *compute* $\epsilon(P) := \min\{\epsilon(a) : a \in A(P)\}$, *where* P *is the* xy*-path in the tree whose predecessor function is* p
8: *for each forward arc* a *of* P, *replace* $f(a)$ *by* $f(a) + \epsilon(P)$
9: *for each reverse arc* a *of* P, *replace* $f(a)$ *by* $f(a) - \epsilon(P)$
10: *return to 1*
11: ***end if***
12: *return* $(f, \partial^+(X))$

5.4 容量网络 N 的构造方法

步骤1. 原图 G 的每个顶点 v 变成网络N的两个顶点 x_v 和 y_v，顶点 x_v 到 y_v 有一条弧连接，即$< x_v,y_v >$,其容量为1；

步骤2. 原图 G 中的每条边 e=(u,v) 在网络N中有两条弧 $< y_u,x_v >$和$< y_v,x_u >$，其容量均为∞;

转化：求G中两个顶点 a,b 之间的内部不交路的最大条数r(a,b)的问题就转化为求网络 N 中从 y_a 到 x_b 的最大流问题。


0
1
2
3
4
x0
x1
x2
x3
x4
y0
y1
y2
y3
y4
1
1
1
1
1