生成树

王丽杰

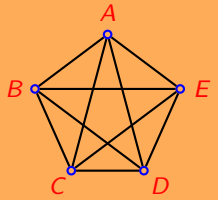
Email: ljwang@uestc.edu.cn

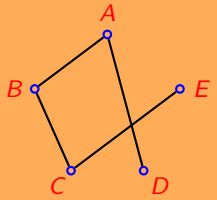
电子科技大学 计算机学院

2016-

生成树

考虑构建一个包含 5 个信息中心 A,B,C,D,E 的通信系统，可能的光纤连接如下左图所示。由于费用限制，要求铺设尽可能少的光纤线路，但又必须保持网络畅通。这实际上就是要求出一个边最少的连通图，这恰好符合树的特点。因而问题转化成在一个连通图中找到一棵树，一种可能的方式如右图所示。





生成树

Definition

给定图 G = <V, E> ,

生成树

Definition

给定图 G = <V, E>，

- 若 G 的某个生成子图是树，则称之为 G 的生成树 (spanning tree)，记为 T_G。生成树 T_G 中的边称为树枝。

生成树

Definition

给定图 G = <V, E>，

- 若 G 的某个生成子图是树，则称之为 G 的生成树 (spanning tree)，记为 T_G。生成树 T_G 中的边称为树枝。
- G 中不在 T_G 中的边称为弦，T_G 的所有弦的集合称为生成树的补。

生成树

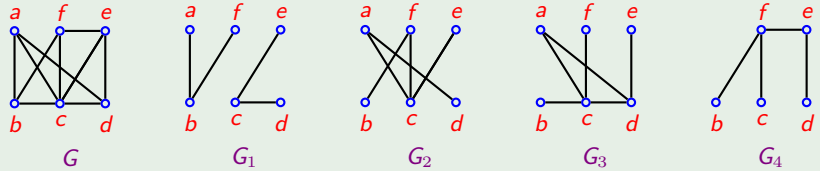
Definition

给定图 G = <V, E>，

- 若 G 的某个生成子图是树，则称之为 G 的生成树 (spanning tree)，记为 T_G。生成树 T_G 中的边称为树枝。
- G 中不在 T_G 中的边称为弦，T_G 的所有弦的集合称为生成树的补。

Example

对下图 G，则 G_1，G_2，G_3，G_4 四个图中，哪一个是它的生成树?



生成树

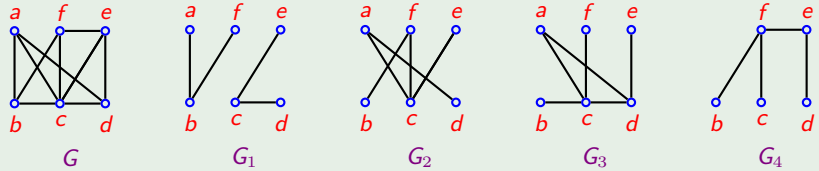
Definition

给定图 G = <V, E>，

- 若 G 的某个生成子图是树，则称之为 G 的生成树 (spanning tree)，记为 T_G。生成树 T_G 中的边称为树枝。
- G 中不在 T_G 中的边称为弦，T_G 的所有弦的集合称为生成树的补。

Example

对下图 G，则 G_1，G_2，G_3，G_4 四个图中，哪一个是它的生成树？(G_2)



生成树存在的条件

Theorem

一个图 $G =< V, E >$ 存在生成树 $T_G =< V_T, E_T >$ 的充分必要条件是 G 是连通的。

生成树存在的条件

Theorem

一个图 $G =< V, E >$ 存在生成树 $T_G =< V_T, E_T >$ 的充分必要条件是 G 是连通的。

Proof.

□

生成树存在的条件

Theorem

一个图 $G =< V, E >$ 存在生成树 $T_G =< V_T, E_T >$ 的充分必要条件是 G 是连通的。

Proof.

- 必要性：假设 $T_G =< V_T, E_T >$ 是 $G =< V, E >$ 的生成树，由树的定义知，则 T_G 连通的，于是 G 也是连通的。

□

生成树存在的条件

Theorem

一个图 $G =< V, E >$ 存在生成树 $T_G =< V_T, E_T >$ 的充分必要条件是 G 是连通的。

Proof.

- 必要性：假设 $T_G =< V_T, E_T >$ 是 $G =< V, E >$ 的生成树，由树的定义知，则 T_G 连通的，于是 G 也是连通的。
- 充分性：假设 $G =< V, E >$ 是连通的。如果 G 中无回路，G 本身就是生成树。如果 G 中存在回路 C_1，可删除 C_1 中一条边得到图 G_1，它仍连通且与 G 有相同的结点集。如果 G_1 中无回路，G_1 就是生成树。如果 G_1 仍存在回路 C_2，可删除 C_2 中一条边，如此继续，直到得到一个无回路的连通图 H 为止。可见，H 是 G 的生成树。

□

生成树算法

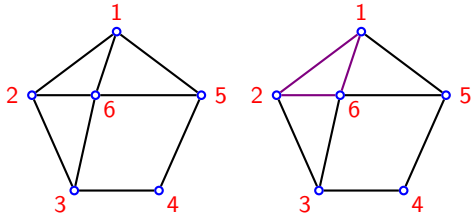
求连通图 $G = (n, m)$ 的生成树的算法：

- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。

生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

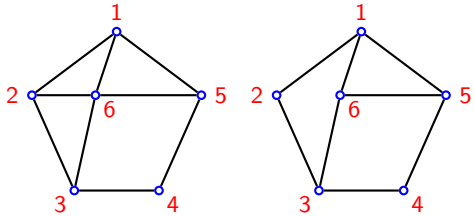
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

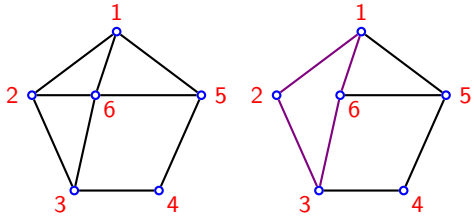
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

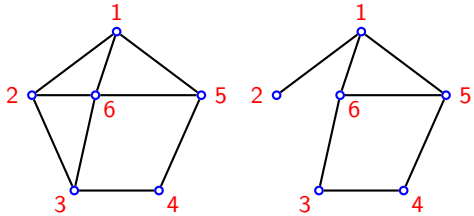
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

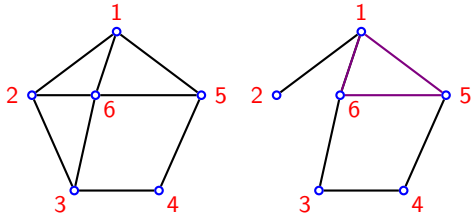
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

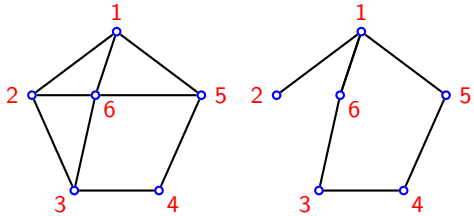
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

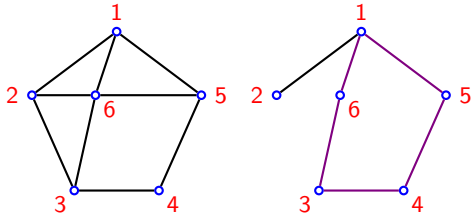
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

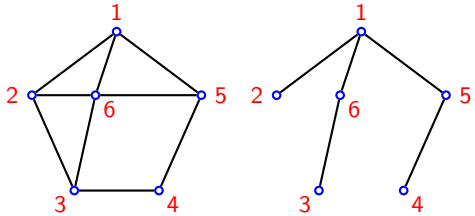
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

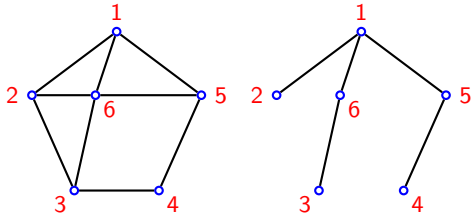
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

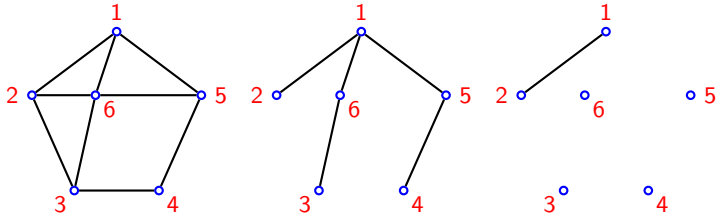
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

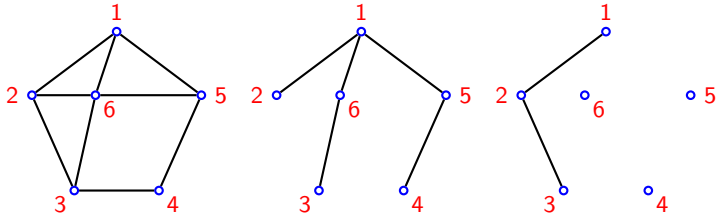
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

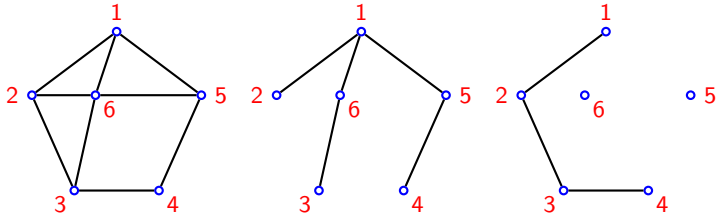
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

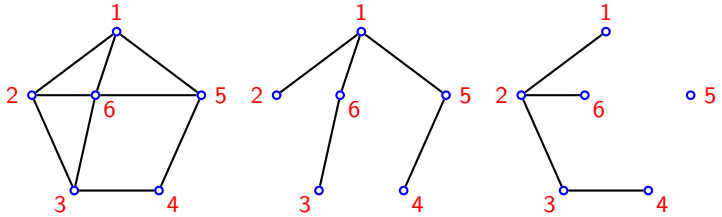
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

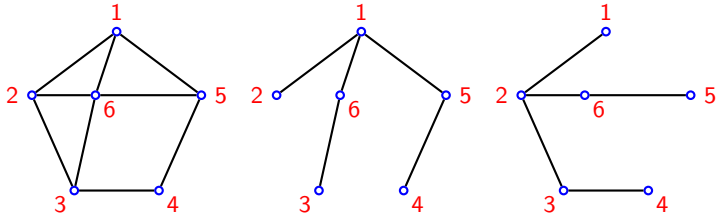
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

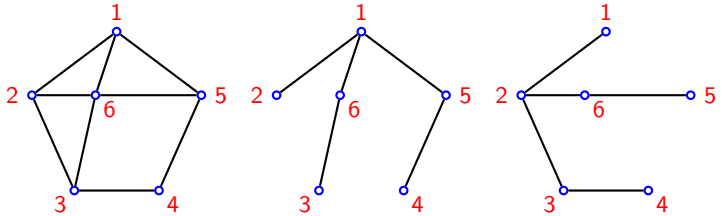
- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



生成树算法

求连通图 $G = (n, m)$ 的生成树的算法：

- 破圈法: 循环找到图中的回路并删除回路中的一条边，直到删除的边的总数为 $m - n + 1$。
- 避圈法: 循环选取 G 中一条与已选取的边不构成回路的边，直到选取的边的总数为 $n - 1$。



可见，生成树不唯一。(为什么 ?)

生成树算法

破圈法和避圈法很多时候不实用，这是由于他们都需要找出回路或验证不存在回路。因而较常用的方法是深度优先搜索算法和广度优先搜索算法来求出生成树。深度优先算法涉及回溯，这里以广度优先搜索算法为例来说明。

☞ 生成树的广度优先搜索算法

连通图 $G=<V,E>$,

1. 任选 $s \in V$，将 s 标记为 0，令 $L=\{s\}$，$V=V-\{s\}$，$k=0$，$E_G=\varnothing$；

生成树算法

破圈法和避圈法很多时候不实用，这是由于他们都需要找出回路或验证不存在回路。因而较常用的方法是深度优先搜索算法和广度优先搜索算法来求出生成树。深度优先算法涉及回溯，这里以广度优先搜索算法为例来说明。

☞ 生成树的广度优先搜索算法

连通图 $G=<V,E>$,

1. 任选 $s \in V$，将 s 标记为 0，令 $L=\{s\}$，$V=V-\{s\}$，$k=0$，$E_G=\varnothing$；
2. 如果 $V=\varnothing$，则结束，E_G 为所求的生成树中包含的所有边。否则令 $k=k+1$；

生成树算法

破圈法和避圈法很多时候不实用，这是由于他们都需要找出回路或验证不存在回路。因而较常用的方法是深度优先搜索算法和广度优先搜索算法来求出生成树。深度优先算法涉及回溯，这里以广度优先搜索算法为例来说明。

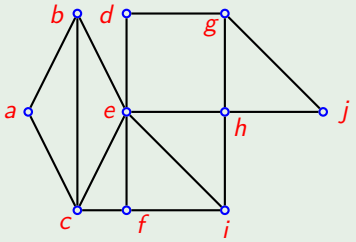
☞ 生成树的广度优先搜索算法

连通图 $G=<V,E>$,

1. 任选 $s \in V$，将 s 标记为 0，令 $L=\{s\}$，$V=V-\{s\}$，$k=0$，$E_G=\varnothing$；
2. 如果 $V=\varnothing$，则结束，E_G 为所求的生成树中包含的所有边。否则令 $k=k+1$；
3. 依次对 L 中所有标记为 $k-1$ 的结点 v，如果它与 V 中的结点 w 相邻接，则将 w 标记为 k，指定 v 为 w 的前驱，令 $L=L\cup\{w\}$，$V=V-\{w\}$，$E_G=E_G\cup\{(v,w)\}$，转 (2)；

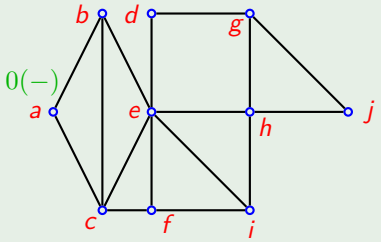
广度优先搜索求生成树

Example



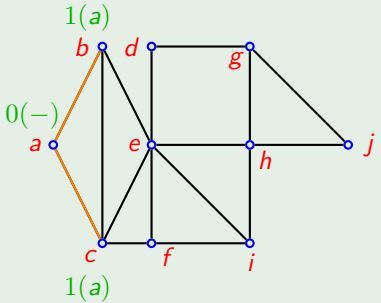
广度优先搜索求生成树

Example



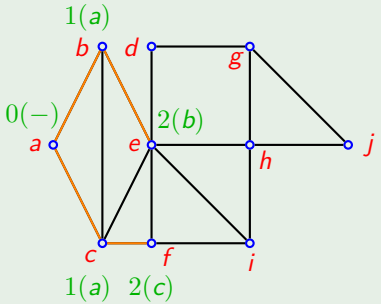
广度优先搜索求生成树

Example



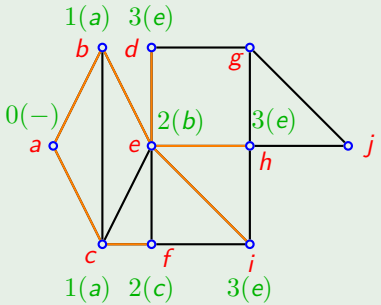
广度优先搜索求生成树

Example



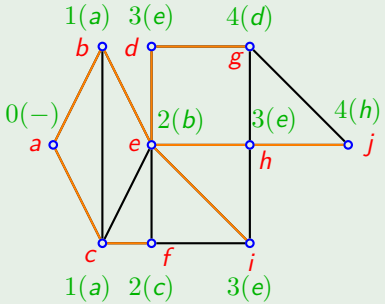
广度优先搜索求生成树

Example



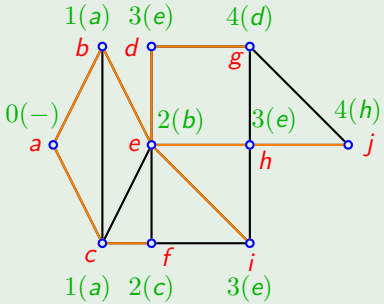
广度优先搜索求生成树

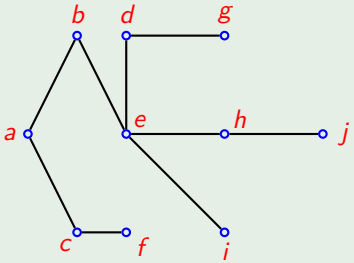
Example



广度优先搜索求生成树

Example

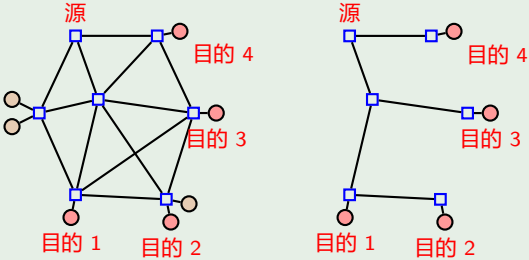




生成树在网络中的应用-IP 组播

Example

为了让数据能够穿过路由器组成的网络通路尽快到达目的计算机，并且要尽量的节省网络资源，则这些数据走过的通路就不应该存在回路，这也恰好符合树的特点，因而只要找出源计算机，中间通信网络和目的计算机所构成的连通图的生成树即可。下面右图给出了左图所对应的一棵生成树。



IP 网络示意图　　组播生成树

THE END, THANKS!