# 离散数学

## 第八章: 树

卢杨

厦门大学信息学院计算机科学与技术系

luyang@xmu.edu.cn

# 8.1 无向树

# 无向树

- 树是一种特殊图, 在计算机领域中具有非常重要的应用.
- 本章所讲回路均指初级回路或简单回路.

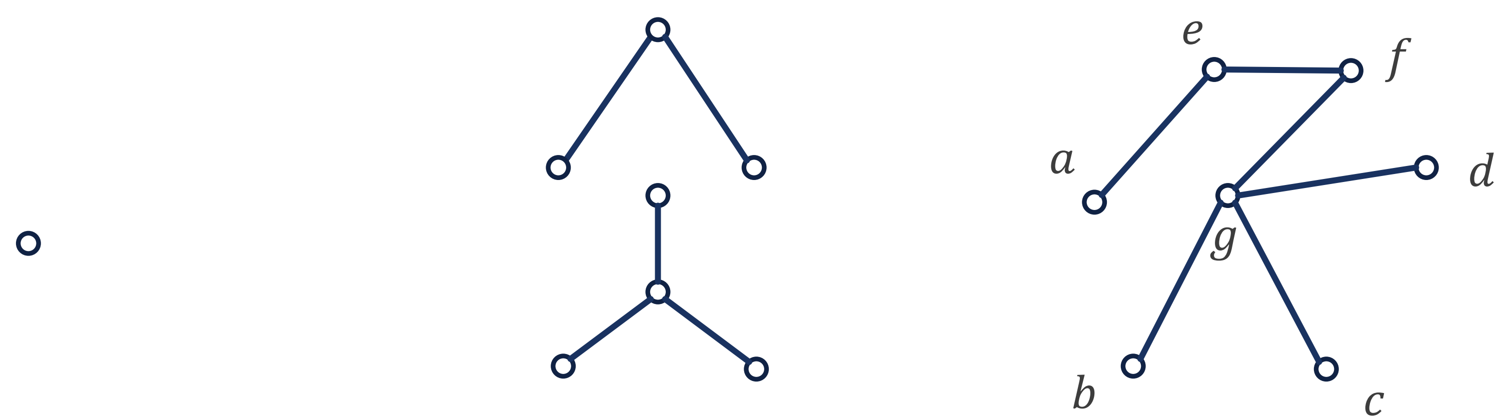
**定义 8.1**

连通不含回路的无向图称为无向树, 简称为树. 常用$T$表示一颗树. 每个连通分支都是树的非连通无向图称为森林. 平凡图称为平凡树.

- 设$T = \langle V, E \rangle$为一棵无向树, $v \in V$, 若$d(v) = 1$, 则称$v$为$T$的树叶; 若$d(v) \geq 2$, 则称$v$为$T$的分支点.

# 无向树

- (1)是平凡树; (2)是2棵树组成的森林; (3)是1棵无向树.
- (3)中, $a, b, c, d$均为树叶; $e, f, g$均为分支点.




(1) (2) (3)

# 无向树

**定理 8.1**

设$G = \langle V, E \rangle$, $|V| = n$, $|E| = m$下列各命题等价.

(1) $G$是树 ($G$连通且不含回路);

(2) $G$的每对顶点之间存在唯一的一条初级通路;

(3) $G$是连通的, 且$m = n - 1$;

(4) $G$中无回路, 且$m = n - 1$;

(5) $G$中无回路，但在$G$中任何两个不相邻的顶点之间增加一条新边，就得到唯一的一条初级回路;

(6) $G$是连通的, 且$G$中任何边均是桥. (树$T$中分支点必为割点. 树是最弱的连通图).

- 我们使用循环证明法来证明该定理.

# 无向树

(1) $G$是树 ($G$连通且不含回路);

(2) $G$的每对顶点之间存在唯一的一条初级通路;

证明 (1)⇒(2)

- 首先证明$G$的每对顶点之间存在通路. 设$u$, $v$为$G$中任意两个顶点. 由于$G$是连通的, 所以$u, v$之间有通路.
- 再通过反证法证明该通路是初级的. 若该通路不是初级的, 则必然存在某个$v'$在该通路中出现2次, 则$G$中必有回路, 与$G$不含回路矛盾, 因此该通路是初级通路.
- 再证明该初级通路是唯一的. 若$u$, $v$之间的初级通路多于一条, 必形成回路, 这与$G$中不含回路矛盾, 所以$G$的每对顶点之间存在唯一的一条初级通路.

# 无向树

(2) $G$的每对顶点之间存在唯一的一条初级通路;

(3) $G$是连通的, 且$m = n - 1$;

证明 (2)⇒(3)

- 首先证明$G$是连通的. 由于$G$中任意两个顶点之间均有初级通路, 所以任意两个顶点均是连通的, 所以$G$是连通的.
- 再通过归纳法证明$m = n - 1$.
  - 当$n = 1$时, $G$为平凡树, $m = 0$, 结论显然成立.
  - 假设当$n \leq k$时结论成立, 证明$n = k + 1$时结论也成立.
    - 设$e = (u, v)$为$G$中的一条边，由(2)可知$uv$是唯一的初级通路，因此$G - e$有两个连通分支$G_1$与$G_2$.
    - 设它们的顶点数为$n_1, n_2$, 边数为$m_1, m_2$, 显然$n_1 \leq k, n_2 \leq k$, 且$G_1$和$G_2$都有性质(2).
    - 由归纳假设得$m_1 = n_1 - 1, m_2 = n_2 - 1$. 从而$m = m_1 + m_2 + 1 = n_1 - 1 + n_2 - 1 + 1 = n - 1$.

# 无向树

(3) $G$是连通的, 且$m = n - 1$;

(4) $G$中无回路, 且$m = n - 1$;

证明 (3)⇒(4)

只需要证明$G$中无回路. 通过反证法.

- 若$G$中有回路，从回路中删去任意一条边后，所得图仍然连通，若所得图中再有回路, 再从回路中删去一条边, 直到所得图中无回路为止.
- 设共删去$r(r \geq 1)$条边所得图为$G'$. $G'$无回路, 但仍是连通的, 即$G'$是树.
- 由(1)⇒(2)⇒(3)，可知$G'$中$m' = n' - 1$. 而$n' = n$，$m' = m - r$. 于是得到$m - r = n - 1$, 即$m = n - 1 + r\ (r \geq 1)$, 这与已知条件矛盾.

# 无向树

(4) $G$中无回路, 且$m = n - 1$;

(5) $G$中无回路, 但在$G$中任何两个不相邻的顶点之间增加一条新边, 就得到唯一的一条初级回路;

证明 (4)⇒(5)

- 首先证明$G$是连通的. 通过反证法.
  - 假设$G$是非连通的, 则$G$有$k(k \geq 2)$个连通分支$G_1, G_2, \ldots, G_k$. 设$G_i$有$n_i$个顶点, $m_i$条边.
  - 由于$G$中无回路, 故连通分支$G_i$中也无回路, 因此每个连通分支$G_i$都是树.
  - 由(1)⇒(2)⇒(3), 可得$m_i = n_i - 1$, 于是$n = n_1 + n_2 + \cdots + n_k = m_1 + 1 + m_2 + 1 + \cdots + m_k + 1 = m + k$ $(k \geq 2)$. 这与已知$m = n - 1$矛盾, 因此$G$是连通的.
- 因而$G$是连通的, 又是无回路的, 即$G$是树. 由(1)⇒(2), $G$中任意两个不相邻的顶点$u, v$之间存在唯一的初级通路, 在$u$与$v$之间加上边就得到唯一的一条初级回路.

# 无向树

(5) $G$中无回路，但在$G$中任何两个不相邻的顶点之间增加一条新边，就得到唯一的一条初级回路;

(6) $G$是连通的, 且$G$中任何边均是桥. (树$T$中分支点必为割点. 树是最弱的连通图).

证明 (5)⇒(6)

- 首先证明$G$是连通的. 通过反证法.
  - 假设$G$是非连通的，设$G_1$，$G_2$是其中的两个连通分支. $v_1$为$G_1$中的一个顶点，$v_2$为$G_2$中的一个顶点.
  - $v_1$与$v_2$不相邻, 但是在$G$中加边$(v_1, v_2)$不形成回路, 这与已知条件矛盾, 所以$G$是连通的.
- 然后证明$G$中任何边均是桥. 通过反证法.
  - 若$G$中存在一条边$e = (u, v)$不是桥，即$G - e$仍连通，说明在$G - e$中存在$u$到$v$的通路.
  - 此通路与$e$构成$G$中的回路, 这与$G$中无回路矛盾.

# 无向树

(6) $G$是连通的，且$G$中任何边均是桥. (树$T$中分支点必为割点. 树是最弱的连通图).

(1) $G$是树 ($G$连通且不含回路);

证明 (6)⇒(1)

只需证明$G$中无回路. 通过反证法.

- 若$G$中含回路$C$，删除$C$上任何一条边后，所得的图仍连通，这与$G$中任何边均是桥矛盾，因此$G$中不含回路，即$G$是树.

# 无向树

**定理8.2**

设$T$是$n$阶非平凡的无向树, 则$T$至少有两片树叶.

**证明** 在非平凡树中, 任何顶点的度数均大于等于1.

设$G$中有$k$个1度顶点, 即$k$片树叶, 则其余$n-k$个分支点的度数均大于等于2.

由握手定理可知

$$2m=\sum d(v_i)\geq k+2(n-k),$$

由定理8.1知$m=n-1$, 代入上式可得

$$2n-2\geq k+2n-2k,$$

解得$k\geq 2$. 说明$T$至少有两片树叶.

# 无向树

**推论** 阶大于2的树$T$必有割点.

**证明**

由$m = n - 1$可知，$n$永远比$m$多1，根据握手定理$\sum_{i=1}^{n} d(v_i) = 2m = 2(n - 1)$, 在$n > 2$时, 度数和$2(n-1) > n$, 即$T$至少含有一个度数$\geq 2$的分支点$v$.

$T$是树, 是连通的, 将该分支点$v$删去可得$T - \{v\}$不是连通的, 所以$v$是割点. ($\text{如果}v$是悬挂点, $T - \{v\}$还是连通的, 所以要先证明$d(v) \geq 2$)

**推论** 树中分支点必为割点, 树中边均为桥.

- 树是连通性最弱的连通图($m = n - 1$), 但也是最经济的连通图.
  - 在保证连通性的前提下, 边没办法更少了.

# 无向图

例 8.1 已知一棵无向树$T$中4度, 3度, 2度的分支点各一个, 其余的顶点均为树叶, 问$T$中有几片树叶?

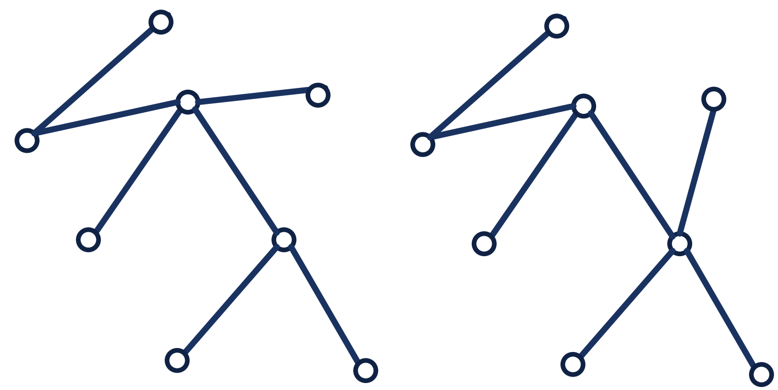
解 设$T$中有$x$片树叶. 则$T$的阶数$n = 1 + 1 + 1 + x = 3 + x$. 由$m = n - 1$可得$m = 2 + x$. 每个树叶都是1度, 由握手定理有总度数

$$4 + 2x = 2m = 4 + 3 + 2 + x = 9 + x,$$

解得$x = 5$片树叶.

例 8.2 满足例8.1中度数列(4, 3, 2, 1, 1, 1, 1, 1)的无向树在同构的意义下是惟一吗?

解 在同构意义下不惟一, 如图所示.



# 生成树

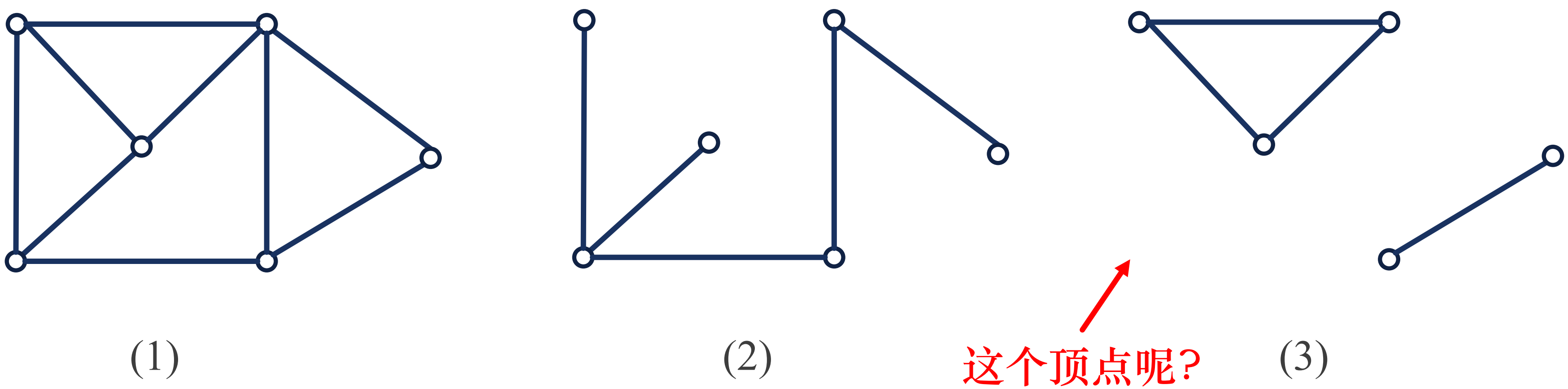
**定义 8.2**

设$G = \langle V, E\rangle$是无向连通图, $T$是$G$的生成子图并且为树, 则称$T$是$G$的生成树. 对任意的边$e \in E(G)$, 若$e \in E(T)$, 则称$e$为$T$的树枝, 否则称$e$为生成树$T$的弦. 称所有弦的集合的导出子图$G[E(G) - E(T)]$为生成树$T$的余树, 记作$\bar{T}$.

- 余树不一定连通, 也不一定不含回路, 因此不一定是树.
- 一般来说, 连通图的生成树不是唯一的.
- 一个图$G$与它的生成树的差别在于$G$可能包含有回路, 而生成树不包含有回路.

# 生成树

例 下图中, (2)是(1)的一颗生成树, (3)是(1)的余树.

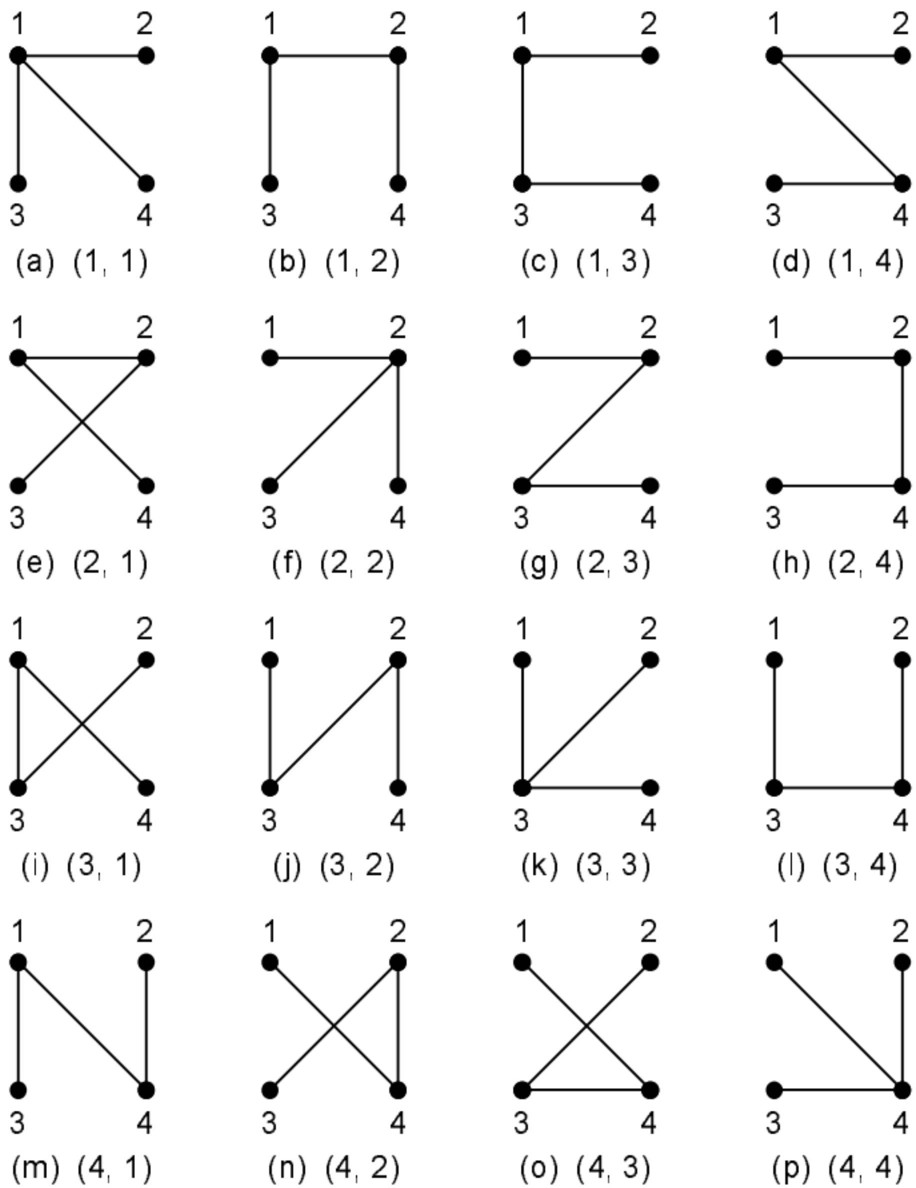


# 课堂练习

画出$K_4$的所有生成树. 其中非同构的有几种?

解

16个生成树, 其中2种非同构




Source: https://www.csie.ntu.edu.tw/~kmchao/tree07spr/counting.pdf

# 生成树

**定理 凯莱公式**

$K_n$的生成树的个数是$n^{n-2}$.

**定理 8.3**

任何无向连通图$G$都存在生成树.

**证明**

- 若$G$中无回路，由于$G$是连通的，所以$G$是树，所以$G$就是自己的生成树.
- 若$G$中有回路$C$，在$C$中任意删去一条边，不影响图的连通性. 若所得图中还有回路, 就继续在回路中删除一条边直到没有回路. 设最后的图为$G'$, $G'$是连通的且无回路, 所以$G'$是$G$的生成树.

# 基本回路

**定理**

设$T$是无向连通图$G$中的一棵生成树，$e$为$T$的任意一条弦，则$T \cup e$构成$G$的一个初级回路，该回路只含一条弦，其余边均为树枝，而且不同的弦$e$对应的初级回路是不同的.

**证明** 设$e = (u, v)$为$T$中的任意一条弦.

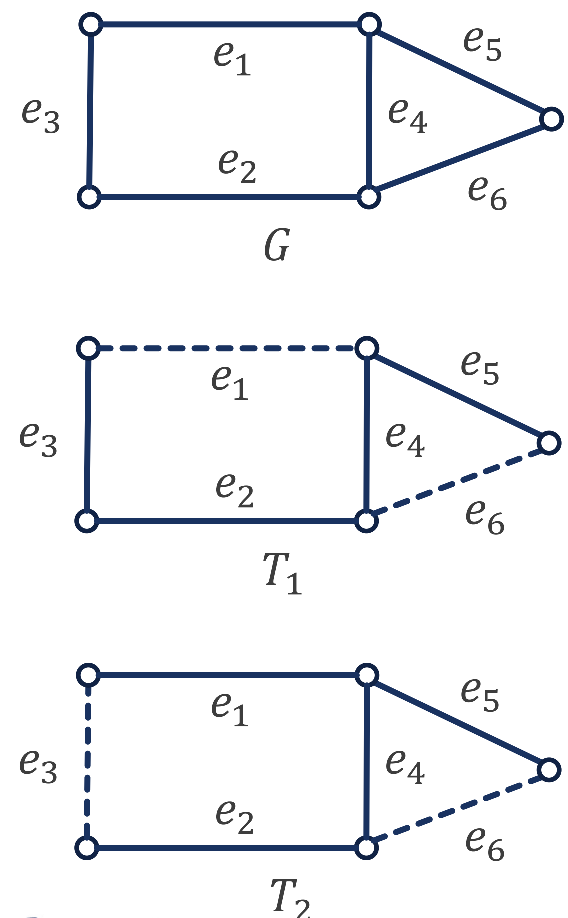
$G$中无回路，但在$G$中任何两个不相邻的顶点之间增加一条新边，就得到唯一的一条初级回路.

- 由于$e$是弦，所以$u$和$v$在$T$中不相邻.
- 由定理8.1(5)可知，$u$与$v$之间增加一条新边，也就是$e$，放入$T$中后得到一条唯一的初级回路. 该回路中只含一条弦，其余边均为树枝.
- 当$e_1, e_2$为任意不同的弦时，对应的初级回路是不同的，因为：
  - $e_2$不在$e_1$对应的初级回路$C_{e_1}$中，因为初级回路$C_{e_1}$中只含一条弦，就是$e_1$；
  - $e_1$不在$e_2$对应的初级回路$C_{e_2}$中，因为初级回路$C_{e_2}$中只含一条弦，就是$e_2$.

# 基本回路

**定义 8.3**

设$G$是$m$条边的$n$阶无向连通图，$T$是$G$的一棵生成树，$T$的$m-n+1$条弦为$e_1, e_2, \dots, e_{m-n+1}$. $G$中仅含$T$的一条弦$e_r$的回路$C_r$称作对应弦$e_r$的基本回路，$\{C_1, C_2, \dots, C_{m-n+1}\}$称作$G$对应$T$的基本回路系统.




**例 8.3** 求该图$G$的两个生成树$T_1, T_2$的基本回路系统.

**解** $T_1$: $\{e_1e_4e_2e_3, e_6e_4e_5\}$; $T_2$: $\{e_3e_1e_4e_2, e_6e_4e_5\}$.

- 基本回路的个数和弦数是相同的，都等于$m-n+1$.

# 基本割集

$G$是连通的, 且$G$中任何边均是桥.

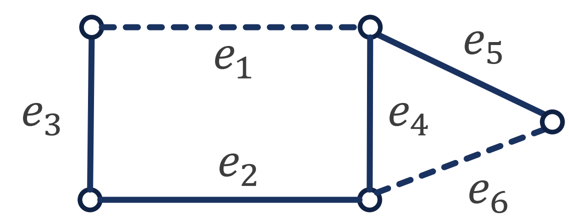
- 设$T$是连通图$G$的一棵生成树, $e$是一条树枝. 根据定理8.1(6), $T-e$是不连通的, 它有两个连通分支$T_1$和$T_2$. 于是, 由$e$和$T$的弦就可以构成一个$G$的割集.

**定义 8.4**

设$T$是$n$阶连通图$G$的一棵生成树, $e'_1, e'_2, \dots, e'_{n-1}$为$T$的树枝. 设$S_r$是只含树枝$e'_r$其余边均为弦的$G$的割集, 则称$S_r$为$G$的对应树枝$e'_r$的基本割集, 并称$\{S_1, S_2, \dots, S_{n-1}\}$为$G$对应$T$的基本割集系统.

- 基本割集的个数和树枝数是相同的, 都等于$n-1$.

**例** 在该图中, 对应$e_5, e_4, e_2, e_3$的基本割集系统为$\{\{e_5, e_6\}, \{e_4, e_1, e_6\}, \{e_2, e_1\}, \{e_3, e_1\}\}$.



# 基本割集

求树枝$e$对应的基本割集的方法如下:

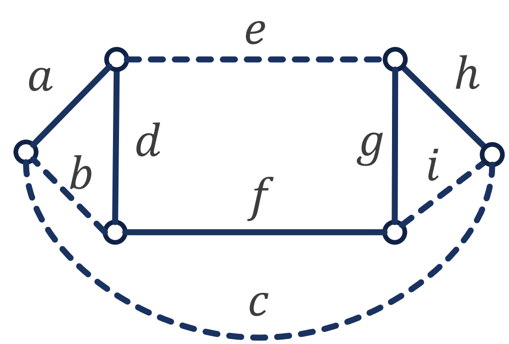
- 将$e$从$T$中删除, 得到$T-e$的两个连通分支为$T_1, T_2$, 设$T_1$和$T_2$的顶点集为$V_1, V_2$.
- 则$e$对应的基本割集为$e \cup S_e$, 其中

$$S_e = \{e' \mid e' \in E(G) \wedge e'\text{的一个端点在}V_1\text{中,另一个端点在}V_2\text{中}\}.$$

# 基本回路与基本割集

例 8.4 求该图$G$的生成树$T$的基本回路系统和基本割集系统.

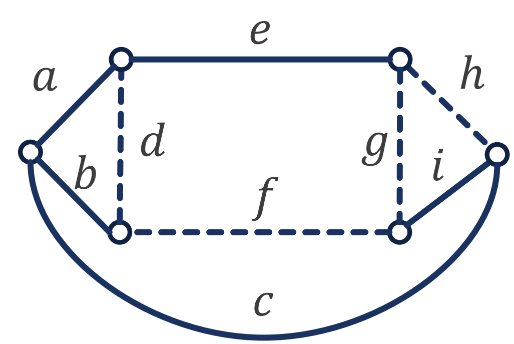
解 首先确定这两个系统中元素的个数. $G$的顶点数$n=6$, 边数$m=9$, 基本回路个数为$m-n+1=4$, 基本割集个数为$n-1=5$.




- $T$的4条弦$e$，$b$，$c$，$i$对应的基本回路为$edfg, bda, cadfgh, igh$.
- $T$的5条树枝$a$, $d$, $f$, $g$, $h$对应的基本割集为$\{a,b,c\}$, $\{d,e,b,c\}$, $\{f,e,c\}$, $\{g,e,i,c\}$, $\{h,i,c\}$.

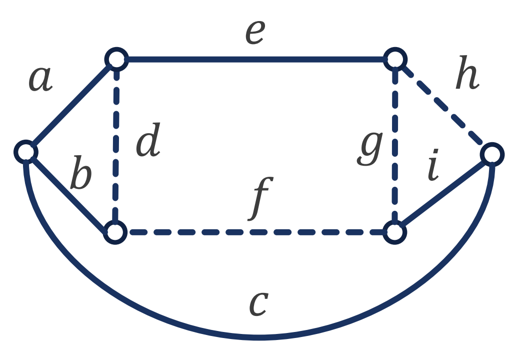
# 课堂练习

求该图$G$的生成树$T$的基本回路系统和基本割集系统.



# 课堂练习

求该图$G$的生成树$T$的基本回路系统和基本割集系统.




解 首先确定这两个系统中元素的个数. $G$的顶点数$n=6$, 边数$m=9$, 基本回路个数为$m-n+1=4$, 基本割集个数为$n-1=5$.

- $T$的4条弦$d$, $f$, $g$, $h$对应的基本回路为$dab, fbci, gicae, hcae$.
- $T$的5条树枝$a$, $e$, $b$, $c$, $i$对应的基本割集为 $\{a,d,g,h\}$ , $\{e,g,h\}$ , $\{b,d,f\}$ , $\{c,f,g,h\}, \{i,g,f\}$.

# 最小生成树

- 想给城镇之间铺路, 把若干城市连接起来 (成为连通图), 怎样才能使所用的线路总长最短 (或时间最少, 资源最少) 呢? 问题的实质就是求带权的最小生成树问题.

**定义 8.5**

对图$G$的每条边$e$附加一个实数$w(e)$, 称$w(e)$为边$e$的权. $G$连同附加在各边上的权称为带权图, 常记作$G = \langle V, E, W\rangle$. 设$G_1 \subseteq G$, 称$\sum_{e\in E(G_1)} w(e)$为$G_1$的权, 记作$W(G_1)$.

**定义 8.6**

设无向连通带权图$G = \langle V, E, W\rangle$, $G$的所有生成树中$W(T)$最小的生成树$T$称为$G$的最小生成树.

# 避圈法

- 设$n$阶简单无向连同带权图$G = \langle V, E, W \rangle$有$m$条边. 下面介绍求最小生成树的算法: 避圈法 (Kruskal算法).

  1. 将$m$条边按权从小到大顺序排列, 设为$e_1, e_2, \ldots, e_m$.
  2. 令$T = \emptyset$.
  3. 依次取边并判断$T \cup e_i$是否有回路.
  4. 若无回路, 则更新$T \leftarrow T \cup e_i$; 若有回路, 则弃去$e_i$.

- 那么, 为何该算法总能够得到最小生成树? 该算法是否是最高效的呢?

# 避圈法

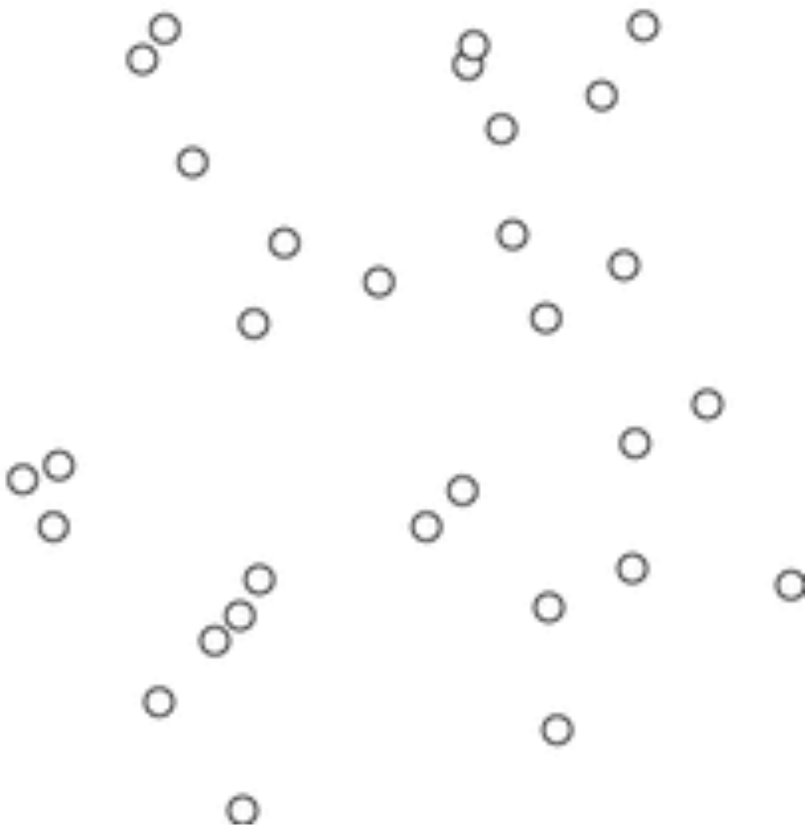
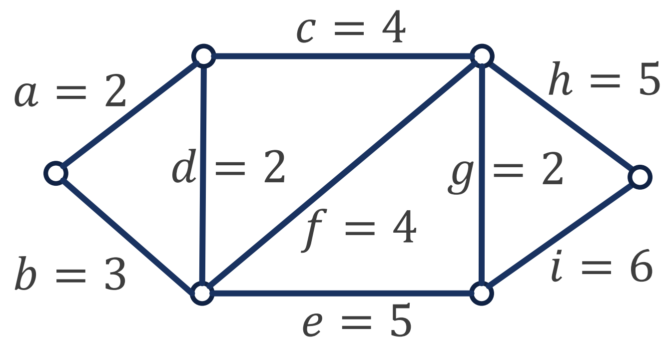




Image source: https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

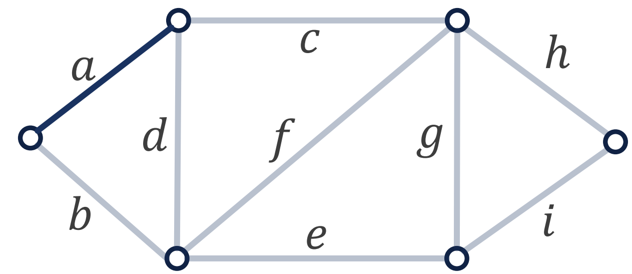
# 避圈法

例 8.5 通过避圈法求该图的最小生成树.

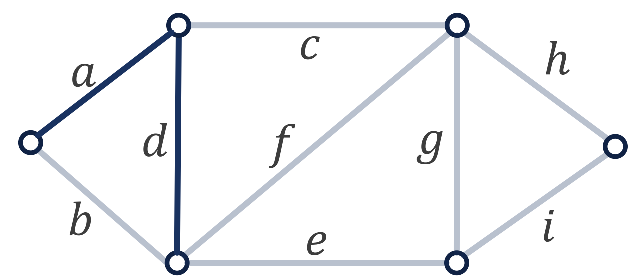



解

先对边的权进行排序：$a, d, g, b, c, f, e, h, i$.




(1) 加入$a$, $T = \{a\}$.




(2) 加入$d$, $T = \{a, d\}$.

# 避圈法

序列: $a, d, g, b, c, f, e, h, i$.

(3) 加入$g$, $T = \{a, d, g\}$.

(4) 加入$b$, 有回路, 扔掉.

(5) 加入$c$, $T = \{a, d, g, c\}$.

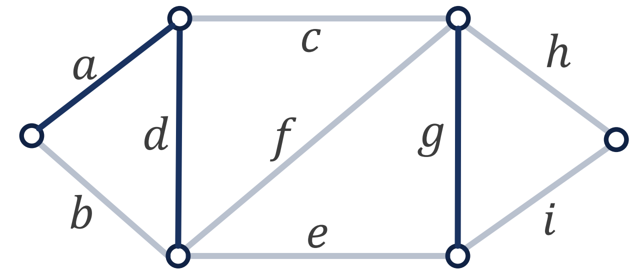
(6) 加入$f$, 有回路, 扔掉.

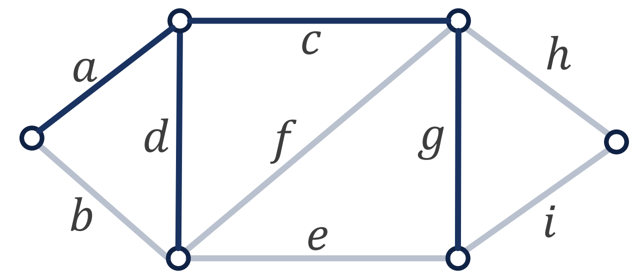
(7) 加入$e$, 有回路, 扔掉.

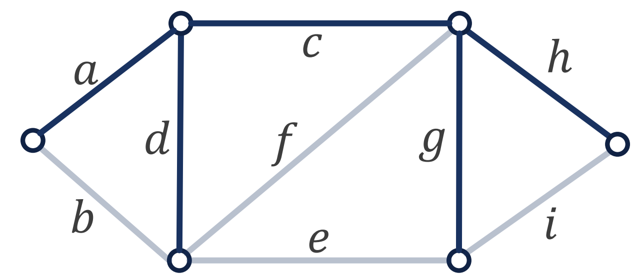
(8) 加入$h$, $T = \{a, d, g, c, h\}$.

(9) 加入$i$, 有回路, 扔掉.

最终该生成树的权为15, 是最小的.



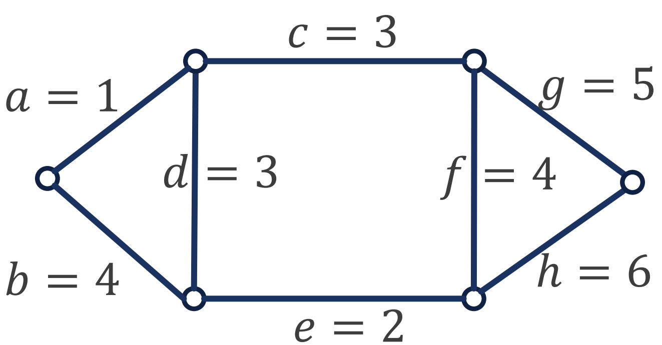
# 避圈法

- 对于计算机，判断加入新的边之后$T$是否有回路的简单方法是判断该边的两个端点是否在同一集合内.

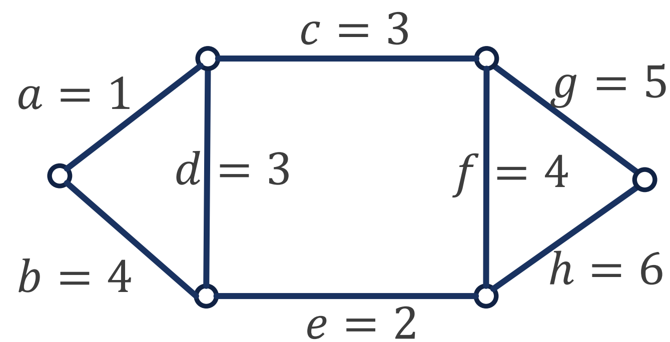
```
void kruskal (int n, int m,
              edge_set E,
              edge_set& F)
{
    index i, j;
    set_pointer p, q;
    edge e;
    sort the m edges in E by weight in nondecreasing order;
    F = ∅;
    initial(n);
    while (number of edges in F is less than n - 1){
        e = edge with least weight not yet consisdered;
        i, j = indices of vertices connected by e;
        p = find(i);
        q = find(j);
        if (!equal(p, q)){
            merge(p, q);
            add e to F;
        }
    }
}
```

# 课堂练习

通过避圈法求该图的最小生成树.



# 课堂练习

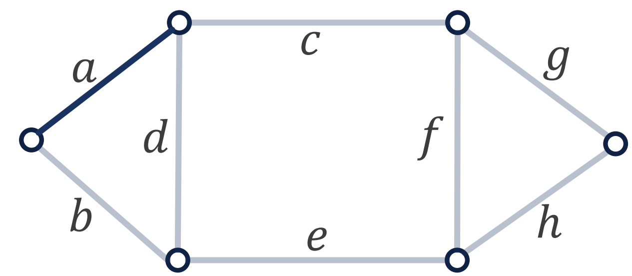



通过避圈法求该图的最小生成树.

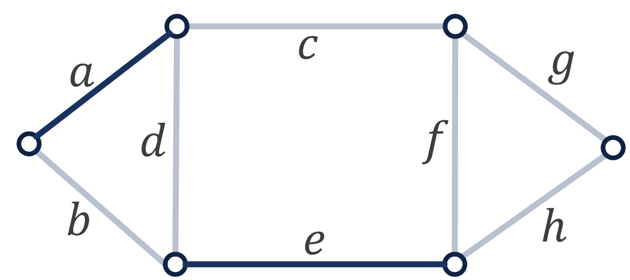
解

先对边的权进行排序：$a, e, c, d, b, f, g, h$.

(1) 加入$a$, $T = \{a\}$.




(2) 加入$e$, $T = \{a, e\}$.



# 课堂练习

序列: $a, e, c, d, b, f, g, h$.

(3) 加入$c$, $T = \{a, e, c\}$.

(4) 加入$d$, $T = \{a, e, c, d\}$.

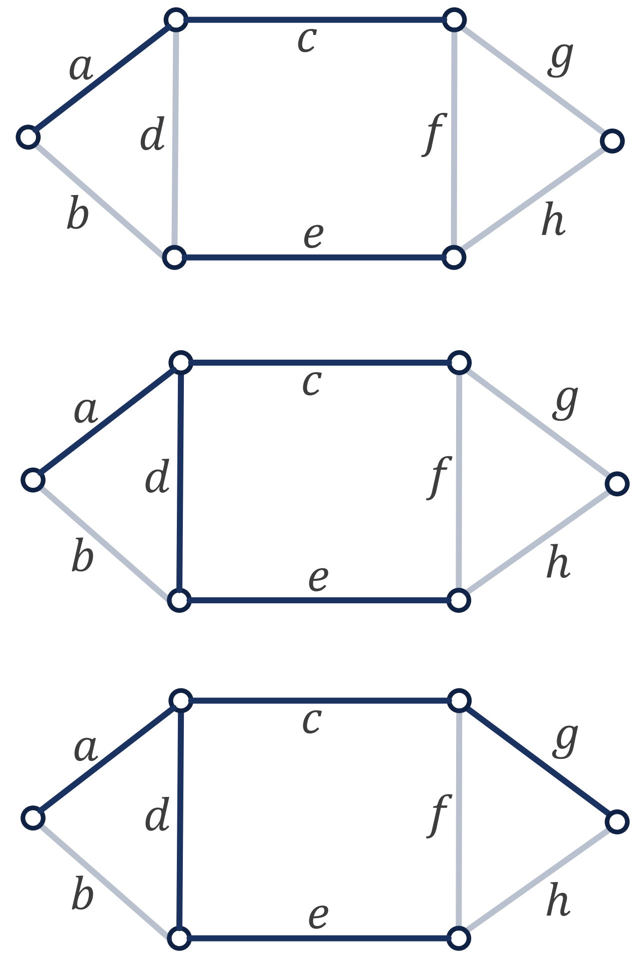
(5) 加入$b$, 有回路, 扔掉.

(6) 加入$f$, 有回路, 扔掉.

(7) 加入$g$, $T = \{a, e, c, d, g\}$.

(8) 加入$h$, 有回路, 扔掉.

最终该生成树的权为14, 是最小的.



# 8.2 根树及其应用

# 根树

- 若有向图$D$的基图是无向树，则称$D$为有向树. 在有向树中，最重要的是根树，它在数据架构，算法设计，数据库等专业课程中极其重要.

**定义 8.7**

设有向图树$T$是一颗非平凡树，如果$T$中有一个顶点的入度为0，其余顶点的入度均为1，则称$T$为根树.

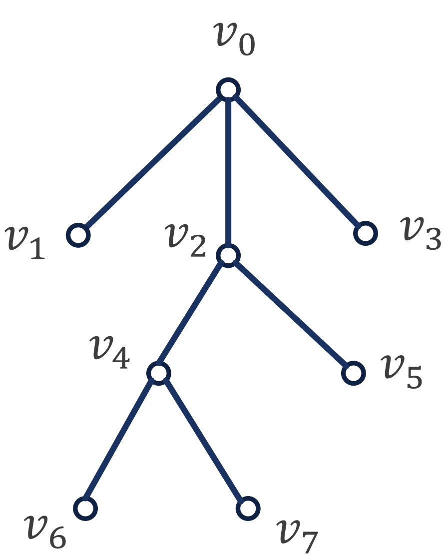
在根树中，入度为0的顶点称为树根；入度为1且出度为0的顶点称为树叶；入度为1且出度大于0的顶点称为内点. 内点和树根统称为分支点.

在根树中，从树根到任一顶点$v$的通路长度称为$v$的层数，记作$l(v)$；称层数相同的顶点在同一层上，层数最大顶点的层数为树高，记作$h(T)$.

# 根树

- 在根树中，由于各有向边的方向的一致性， 因而画根树时，省掉有向边的箭头，并将树根放在最上方，边的方向一律向下或斜下方.

例 该图为一颗根树.




$v_0$为树根，$v_1$，$v_3$，$v_5$，$v_6$，$v_7$均为树叶，$v_2$，$v_4$为内点，$v_0$，$v_2$，$v_4$为分支点.

各顶点层数:

- $l(v_0) = 0$;
- $l(v_1) = l(v_2) = l(v_3) = 1$;
- $l(v_4) = l(v_5) = 2$;
- $l(v_6) = l(v_7) = 3$.

树高为3，即$h(T) = 3$.

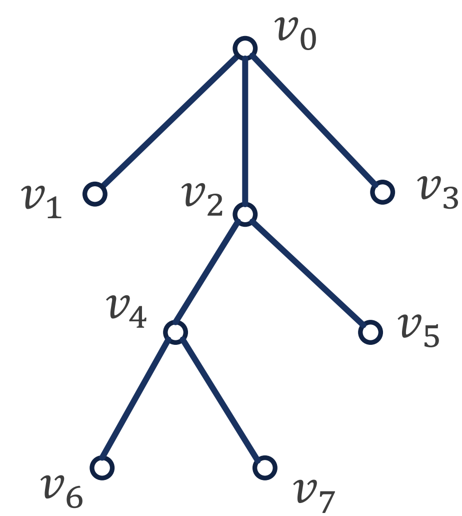
# 根树

**定义**

若顶点$a$邻接到顶点$b$，则称$b$为$a$的儿子，$a$为$b$的父亲；

若$b, c$的父亲相同，则称$b, c$为兄弟；

若$a \neq d$，而$a$可达$d$，则称$a$为$d$的祖先，$d$为$a$的后代.




**例** 该树中，$v_1, v_2, v_3$是兄弟，它们的父亲是$v_0$；

$v_4$和$v_5$是兄弟，它们的父亲是$v_2$；

$v_6$和$v_7$是兄弟，它们的父亲是$v_4$；

$v_0$以外的所有顶点都是$v_0$的后代，$v_0$是它们的祖先.

# 有序树, 正则树, 完全树

**定义 8.8**

如果将根树$T$的每一层上的顶点都规定次序, 则称$T$为有序树.

**定义 8.9**

设$T$为一棵非平凡的根树,

(1) 若$T$的每个分支点至多有$r$个儿子, 则称$T$为$r$元树;

(2) 若$T$的每个分支点都恰有$r$个儿子, 称$T$为$r$元正则树;

(3) 若$r$元树$T$是有序的, 则称$T$为$r$元有序树;

(4) 若$T$是$r$元正则树, 且所有树叶的层数均为树高, 则称$T$为$r$元完全正则树;

(5) 若$T$是$r$元完全正则树, 且$T$是有序的, 则称$T$为$r$元完全正则有序树.

- 完全树一定是正则树, 但是正则树不一定是完全的.

# 2叉树

- 在所有的$r$元树中, 2元树最重要, 2元树又称2叉树.

**定义**

对一棵根树的每个顶点都访问一次且仅访问一次称为遍历.

**定义**

设$T$为一棵根树, $a$是$T$的一个非根顶点, 称$a$及其后代的导出子图$T'$为$T$的以$a$为根的根子树.

- 2元正则有序树的每个分支点的两个儿子导出的根子树分别称为该分支点的左子树和右子树.

# 2叉树的遍历

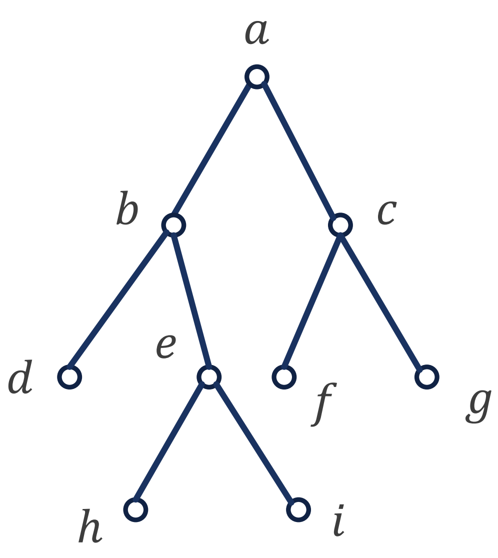
- 设$T$为一棵2元正则有序树，按对树根，左子树，右子树的不同的访问顺序主要有以下3种遍历方法：

(1) 中序遍历法：访问次序为左子树，树根，右子树；

(2) 前序遍历法：访问次序为树根，左子树，右子树；

(3) 后序遍历法：访问次序为左子树，右子树，树根。

- 这里的**前中后**指的是**树根在访问次序中的位置**。




例 该根树按中序，前序，后序遍历的结果分别为：

$$\big(db(hei)\big)a(fcg),$$

$$a\big(bd(ehi)\big)(cfg),$$

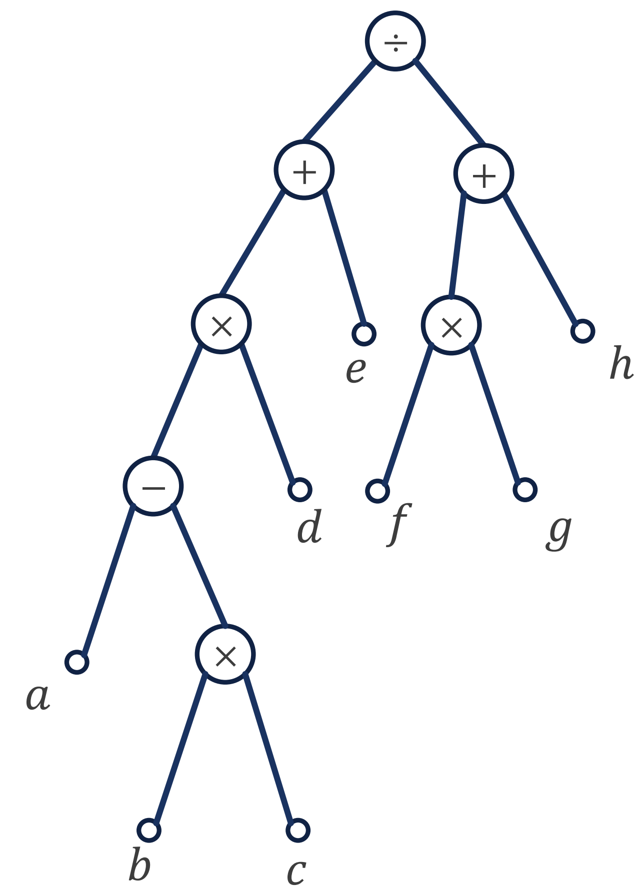
$$(d(hie)b)(fgc)a.$$

# 2叉树的遍历

例 8.6(1) 利用2元有序树表示下面算式:

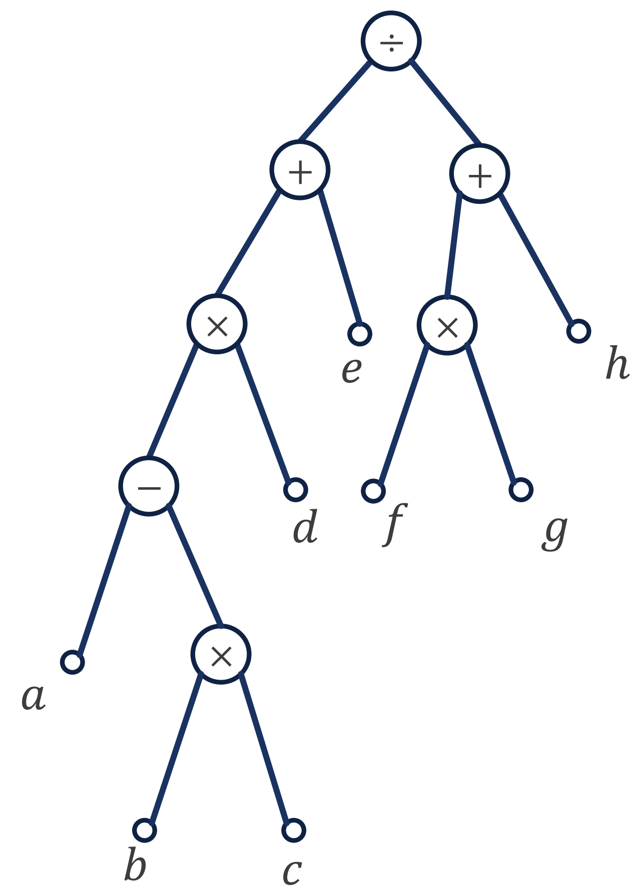
$$\big((a-b\times c)\times d+e\big)\div(f\times g+h).$$

解 将运算符都作为分支点，运算数都作为树叶, 可以表示为该二元有序树.



# 2叉树的遍历

例 8.6(2) 用3种遍历法访问该2元有序树，写出访问结果.




解

- 中序遍历:

$$\Big(\Big(\big(a-(b\times c)\big)\times d\Big)+e\Big)\div\big((f\times g)+h\big)$$

- 前序遍历:

$$\div\Big(+\Big(\times\big(-a(\times bc)\big)d\Big)e\Big)\big(+(\times f g)h\big)$$

- 后序遍历:

$$\Big(\Big(\big(a(bc\times)-\big)d\times\Big)e+\Big)\big((f g\times)h+\big)\div$$

- 中序遍历的结果去掉一些多余的括号后原式结果相同，所以中序遍历访问的结果是还原算式.

# 2叉树的遍历

- 对于前序遍历的访问结果, 可将全部括号去掉, 得到:

$$\div + \times - a \times bcde + \times fgh.$$

对这个表达式规定, 从右到左, 每个运算符与它后面紧邻的2个数进行运算. 这种运算符在参加运算的数的前面的表达方式称为前缀符号法, 又称作波兰符号法.

- 对于后序遍历的访问结果, 可将全部括号去掉, 得到:

$$abc \times - d \times e + fg \times h + \div.$$

对这个表达式规定, 从左到右, 每个运算符与它前面紧邻的2个数进行运算. 这种运算符在参加运算的数的后面的表达方式称为后缀符号法, 又称作逆波兰符号法.

# 课堂练习

给定该算式

$$\Big(\big(a\times(b+c)\big)\times d-e\Big)\div(f+g)\div\big(h\times(i+j)\big)$$

(1) 将以上算式存入一棵2元正则有序树中;

(2) 分别写出上式的波兰符号法和逆波兰符号法表达的形式.

# 课堂练习

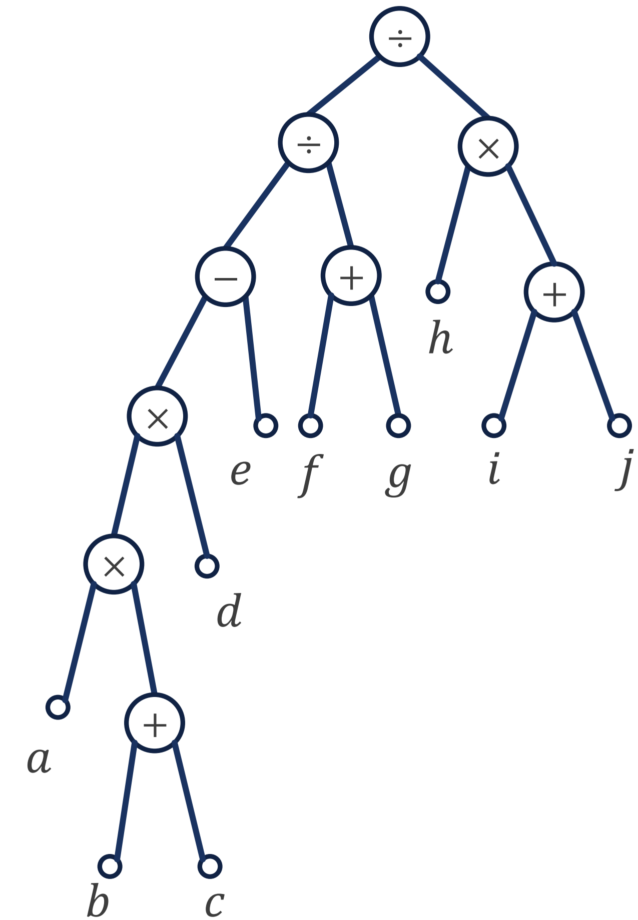
解

波兰符号法表达的形式为

$$\div\div-\times\times a+bcde+fg\times h+ij$$

逆波兰符号法表达的形式为

$$abc+\times d\times e-fg+\div hij+\times\div$$



# 作业

p172

2(2)

3

4

5

6

8

10

# 谢谢

有问题欢迎随时跟我讨论