第三章 树III

计算机系网络所：张小平

清華大學
Tsinghua University

主要内容

- 3.1 树的有关定义
- 3.2 基本关联矩阵及其性质
- 3.3 支撑树的计数
- 3.4 回路矩阵与割集矩阵
- **3.5 支撑树的生成**
- 3.6 Huffman树
- 3.7 最短树
- 3.8 最大分枝

Tsinghua University

支撑树的生成

- 如何得到一个图的支撑树?
  - 树的计数!
  - 树的生成?

支撑树的生成

基本概念：

- 用t表示G的一棵树。假定t_1，t_2是连通图G的两棵树，t_1中共有k条边不属于t_2，则称t_1和t_2的**距离**为k，记做 $d(t_1, t_2) = k$
- 显然，t_2和t_1的距离也是k

支撑树的生成

- 定义3.5.1 设t_1和t_2是连通图G距离为1的两棵树， $t_1 - t_2 = (e)$ ， $t_2 - t_1 = (e')$ 。

  则 $t_2 = t_1 \oplus (e, e')$ 称为 t_1 到 t_2 的

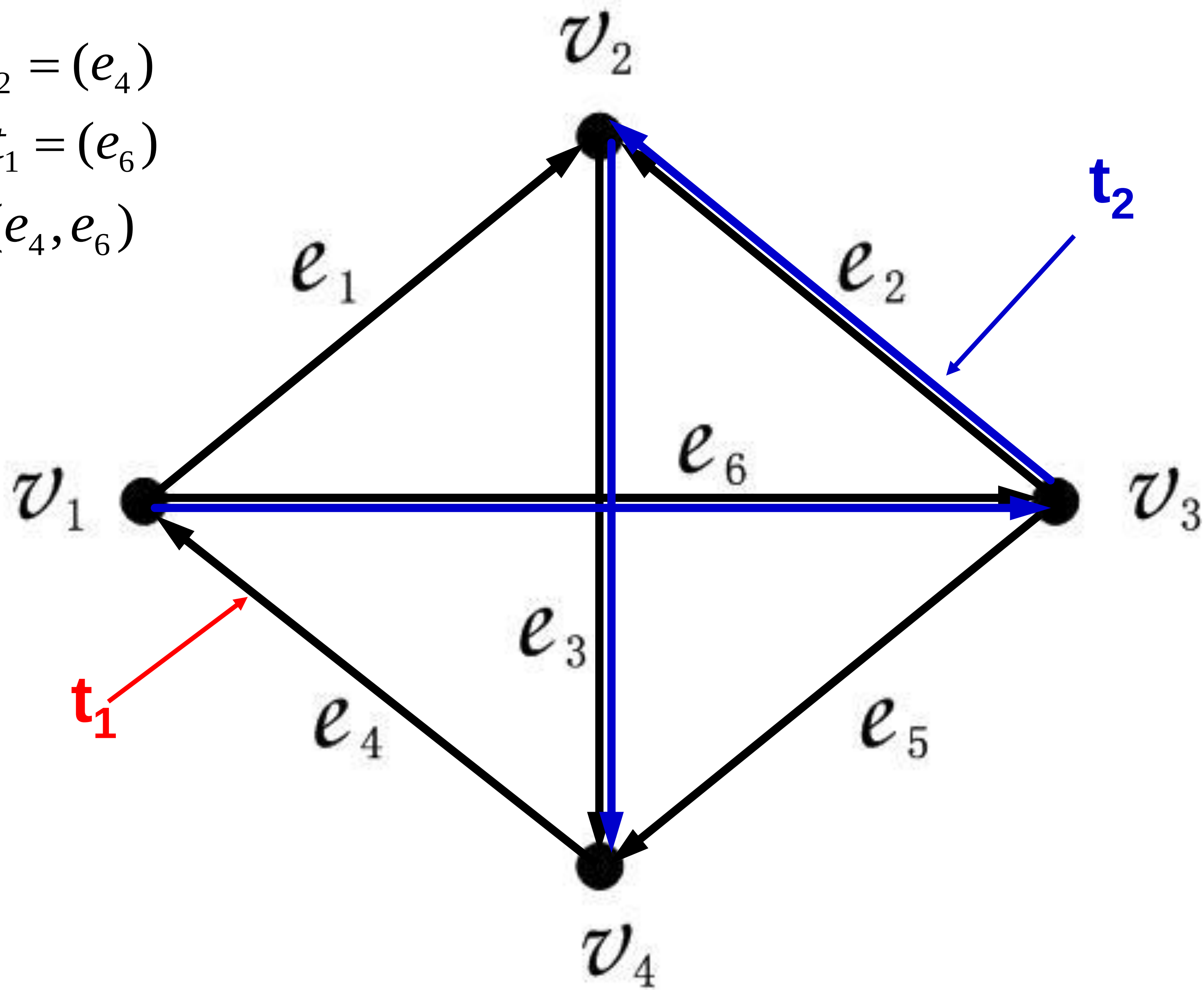
  基本树变换

例：

$$t_1 - t_2 = (e_4)$$

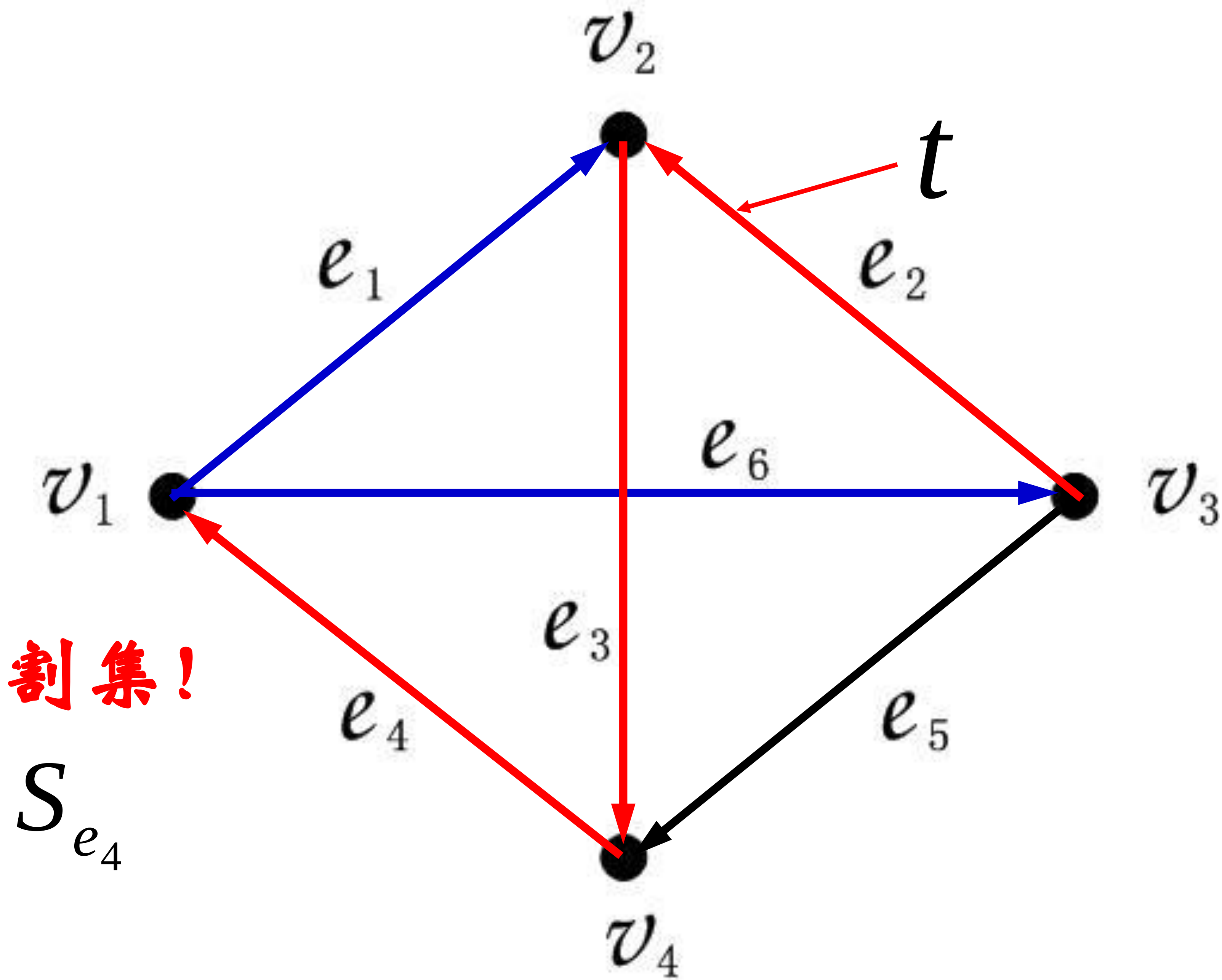
$$t_2 - t_1 = (e_6)$$

$$t_1 \oplus (e_4, e_6)$$



支撑树的生成

- 猜想：假如给定图G的一棵树t_1，某树边为e，则在$t_1 - e$后再增加一条余树边，是否可以生成另外一棵树？



v2
t
e1
e2
e6
v1
v3
e3
割集！
e4
e5
S e4
v4

支撑树的生成

- 定理3.5.1 令 t_1和 t_2是G中距离为1的两棵树，且

$t_1 - t_2 = (e)$，$t_2 - t_1 = (b)$，则$b \in S_e(t_1)$；

反之，若$b \in S_e(t_1)$，则$t_2 = t_1 \oplus (e,b)$是树！

支撑树的生成

证明：$\boldsymbol{t_1 - t_2 = (e)}$，$\boldsymbol{t_2 - t_1 = (b)}$ ⟹ $\boldsymbol{b \in S_e(t_1)}$

- 设$t_1 = (e, a_1, a_2, \cdots, a_k)$，$t_2 = (b, a_1, a_2, \cdots, a_k)$
- 若$b \notin S_e(t_1)$，由于$S_e(t_1)$不含t_1的其它树枝边 $a_1, \cdots, a_k$，因此$S_e(t_1)$也不含 t_2的任何边
- 这样假如删除$S_e(t_1)$，G中仍然保留有t_2，即G仍然为连通图，与$S_e(t_1)$是割集矛盾。

支撑树的生成

$t_1=(e,a_1,a_2,\cdots,a_k)$

- 证明（续）：若$b\in S_e(t_1)$ $\Longrightarrow$ $t_1\oplus(e,b)$是树
  - 若$b\in S_e(t_1)$，而$S_e(t_1)$只含t_1的唯一树枝e，因此$b\neq a_i,(i=1,2,\cdots,k)$，因此 $b\notin t_1$
  - 如果树t_1删除边e，之后添加$S_e(t_1)$内的边b，则将仍然保持连通，即$t_1\oplus(e,b)$仍然会保持连通
  - 此时，$t_1\oplus(e,b)$ 有n个结点，$n-1$条边，保持连通，因此是一棵树。

证毕！

Tsinghua University

支撑树的生成

- 定理3.5.1 令 t_1和 t_2是G中距离为1的两棵树，且$t_1 - t_2 = (e)$，$t_2 - t_1 = (b)$，则$b \in S_e(t_1)$；

  反之，若$b \in S_e(t_1)$，则$t_2 = t_1 \oplus (e,b)$是树！

问题：如何从一棵树构造距离为1的另一棵树？

用树枝边对应基本割集中任一条边取代该树枝边即可

支撑树的生成

- 定理3.5.2 给定G的一棵树t_0，令$t_1, t_2, \cdots, t_p$

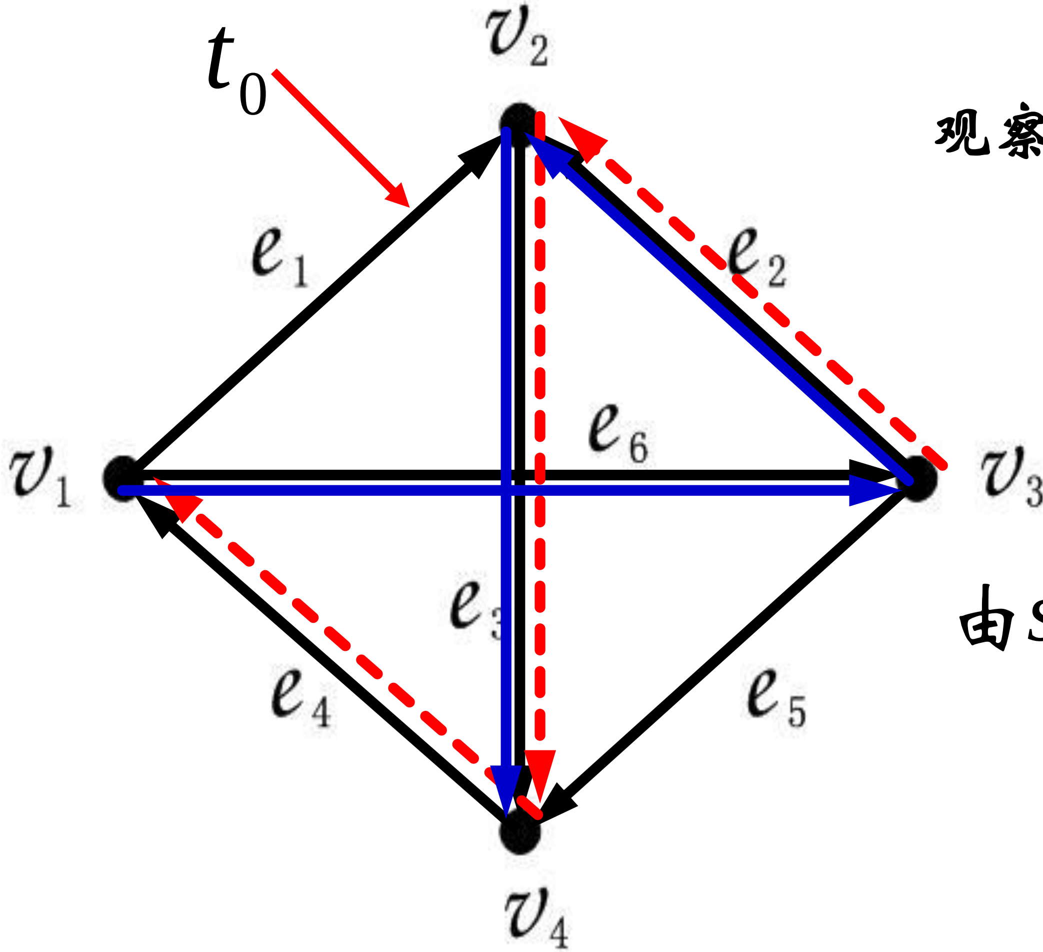
是G中全部满足 $\begin{cases} t_0 - t_i = (e) \\ t_i - t_0 = (b_i) \end{cases} \quad (i = 1,2,\cdots,p)$

的树，则$s_e(t_0) = (e, b_1, b_2, \cdots, b_p)$。

反之，若 $b_i \in s_e(t_0)$，

则 $t_i = t_0 \oplus (e, b_i)$ 是树！

例：求与t_0距离为1的全部树



e_1对应的基本割集？

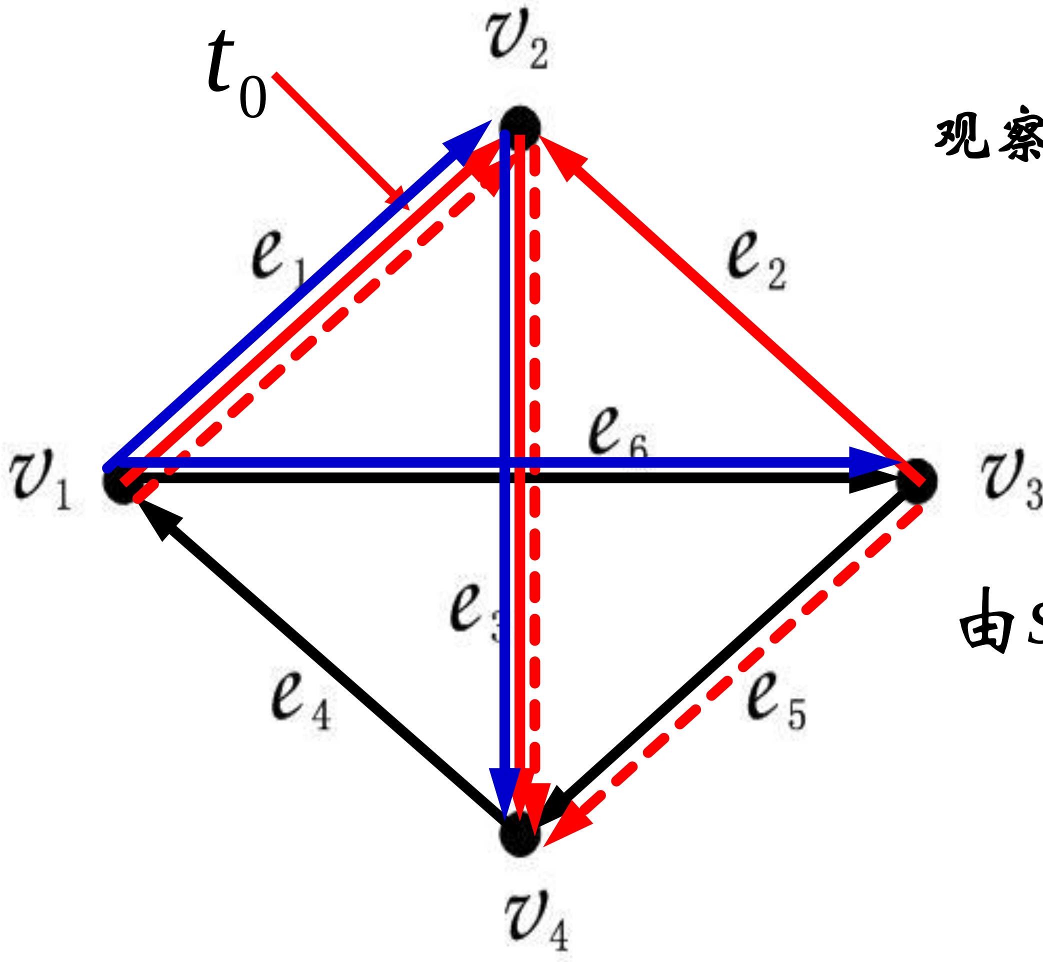
观察： $s_{e_1}(t_0)=(e_1,e_4,e_6)$

由$s_{e_1}(t_0)$：

$$t_1=(e_4,e_2,e_3)$$

$$t_2=(e_6,e_2,e_3)$$

例：求与t_0距离为1的全部树



e_2对应的基本割集？

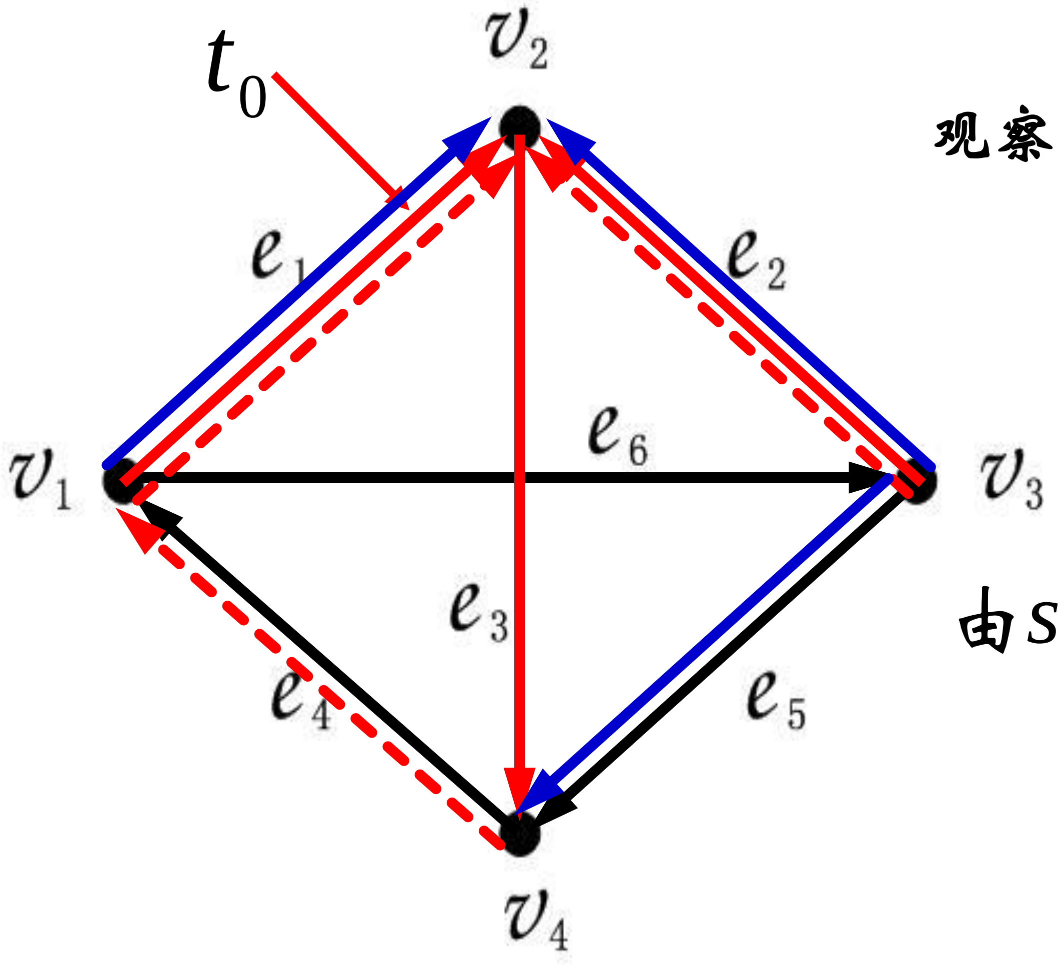
观察：$s_{e_2}(t_0)=(e_2,e_5,e_6)$

由$s_{e_2}(t_0)$：

$$t_3=(e_1,e_5,e_3)$$

$$t_4=(e_1,e_6,e_3)$$

例：求与t_0距离为1的全部树



e_3对应的基本割集？

观察：$s_{e_3}(t_0)=(e_3,e_4,e_5)$

由$s_{e_3}(t_0)$：

$$t_5=(e_1,e_2,e_4)$$

$$t_6=(e_1,e_2,e_5)$$

支撑树的生成

- 定义3.5.2 令

$$T^e = \{t_0 \oplus (e,b) | b \in s_e(t_0), b \neq e\}$$

其中T^e可表述为：

用e的基本割集每条边逐一替代e后生成的新树集合

支撑树的生成

- 归纳：
  - 给定G的一棵参考树t_0
  - t_0可以依据每条边e的基本割集生成新树集合
  - 这样生成的所有的树与t_0的距离为1

支撑树的生成

- 思考：用上述方法，我们可以生成与参考树距离为1的新树集合，但这样生成的新树集合是不是能涵盖与参考树距离为1的所有树？
- 答案是肯定的！

  假定参考树为t_0，t_1与其距离为1，不妨设$t_0 - t_1 = (e)$，$t_1 - t_0 = (b)$，则t_1一定可以通过e的基本割集生成。由于t_1的任意性，可知用上述方法，可覆盖所有距离为1的树

支撑树的生成

- 猜想：设$t_0=(e_1,e_2,\cdots,e_k)$是G中的参考树，则G中与t_0距离为1的树是否会恰在$T^{e_1},T^{e_2}\cdots,T^{e_k}$的某个集合中？

证明：

- 如果能够说明各集合无交集即可
- 观察T^{e_i}与T^{e_j}，前者中的树一定不包括e_i，但是一定包括e_j；后者中的树一定不包括e_j，但是一定包括e_i。
- 因此 $T^{e_i}\bigcap_{i\neq j}T^{e_j}=\phi$

支撑树的生成

- 定理3.5.3：设$t_0=(e_1,e_2,\cdots,e_k)$是G中的参考树，则G中与t_0距离为1的树恰在$T^{e_1},T^{e_2}\cdots,T^{e_k}$的某个集合中！

  证明：　（已证）

支撑树的生成－小结

- 给定图G的参考树t_0，如何得到与其距离为1的其他支撑树？

- 思考：如何用代数的方法，在给定参考树情况下，罗列出与其距离为1的所有树。

- 自学：如何得到与参考树距离为2的树。

主要内容

- 3.1 树的有关定义
- 3.2 基本关联矩阵及其性质
- 3.3 支撑树的计数
- 3.4 回路矩阵与割集矩阵
- 3.5 支撑树的生成
- **3.6 Huffman树**
- 3.7 最短树
- 3.8 最大分枝

Huffman树

提问：

如何在计算机中存储字符串“bdcdbdadbdcdd”？

编码问题：

a→00

b→01

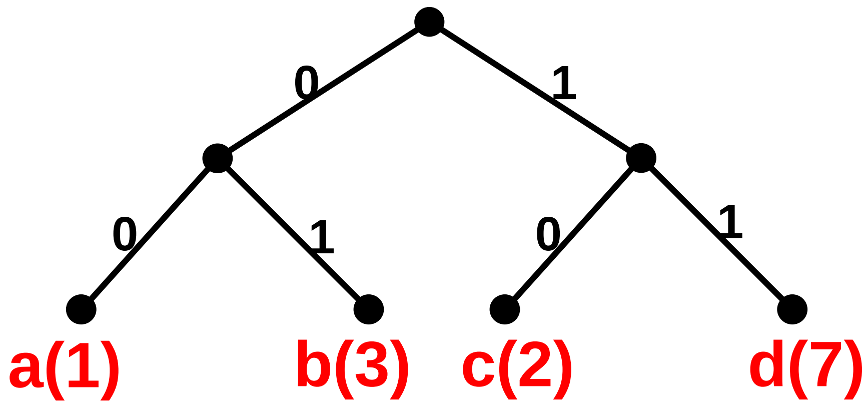
c→10

d→11

则在计算机中存储形式为26位长的二进制串

Huffman树

- 以树的形式表示这种编码方式



Huffman树

- 定义3.6.1 除树叶外，其余结点的正度最多为2的外向树称为二叉树。

  如果它们的正度都是2，则称为完全二叉树。

Huffman树

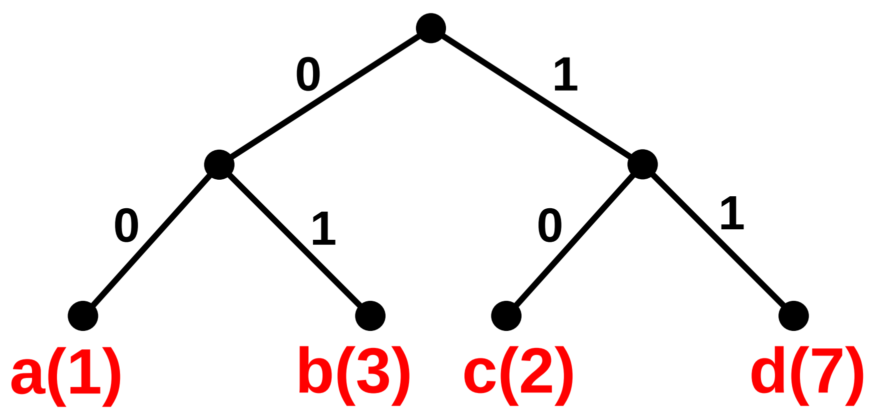
- 定义（续）：对于根结点为v_0的完全二叉树T，如果每个树叶结点v_i都赋予一个正实数w_i，则称之为**赋权二叉树**。从根到树叶的路径$P(v_0, v_i)$所包含的边数记为该路径的长度l_i，这样二叉树T带权的路径总长为$WPL = \sum_i w_i l_i$

  (Weighted Path Length of Tree)

Tsinghua University

Huffman树

- 以树的形式表示这种编码方式

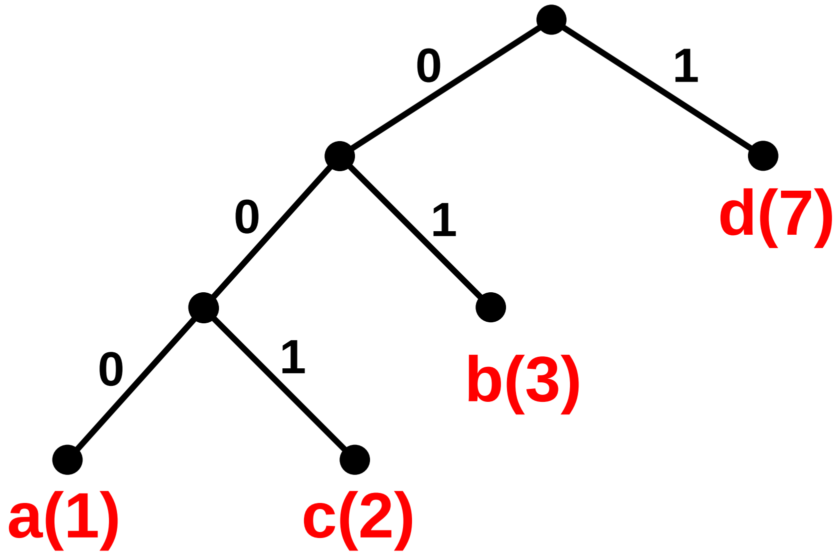


则图中从根结点到每一个树叶结点的带权总长度

即为计算机中的存储长度

Huffman树

- 假如给出另外一种编码方式



即**优化的编码节约了存储空间**

Huffman树

- 定义3.6.2：给定各个树叶的权值，可构造出许多不同的赋权二叉树，其中路径总长最小的二叉树，称为**最优二叉树**。

Huffman树

- 可优化的原因：
  - 字母出现的频度不同
  - 树叶的权值不同

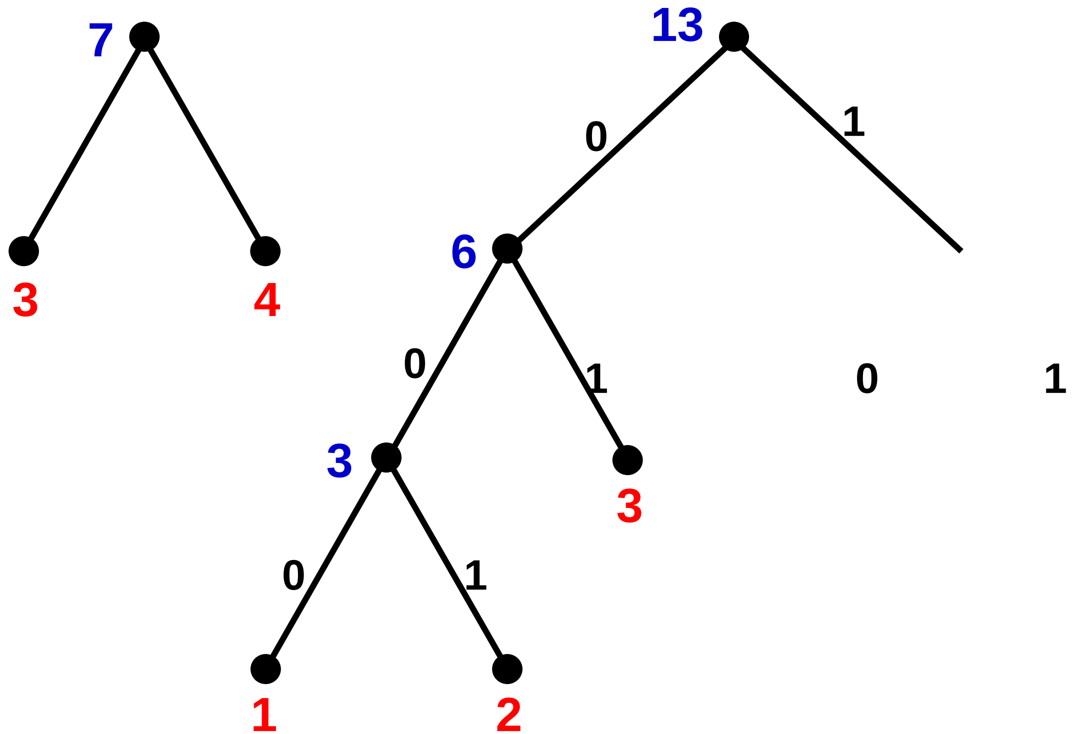
- 思考：
  - 给定n个树叶的权值，如何构造最优二叉树？

Huffman树

- $Huffman$ 树构造算法：
  - 1. 对权序列中的权值进行排序，满足
  $$w_{i_1} \le w_{i_2} \le \cdots \le w_{i_n}$$
  - 2. 计算$w_i = w_{i_1} + w_{i_2}$作为中间结点v_i的权，其左儿子是v_{i_1}，右儿子是v_{i_2}。在权序列中删去w_{i_1}，w_{i_2}，加入w_i，同时$n \leftarrow n-1$。

    若$n=1$，结束，否则转(1)

该算法构造出的二叉树具有什么特点?



Huffman树

- 定理3.6.1 由$Huffman$算法得到的二叉树是最优二叉树。

证明：

假定$n \geq 3$，不妨设各权值为$w_1 \leq w_2 \leq \cdots \leq w_n$

并设存在最优二叉树T。

则必定会有 $$l_1 = \max_i \{l_i\}$$

否则，必存在$l_k > l_1$

Huffman树

假如存在$l_k > l_1$，则令w_1与w_k互换位置，形成树T'

则有 $$WPL(T') = w_k \cdot l_1 + w_1 \cdot l_k + \sum_{i \neq 1,k} w_i \cdot l_i$$

而对于T，有 $$WPL(T) = w_k \cdot l_k + w_1 \cdot l_1 + \sum_{i \neq 1,k} w_i \cdot l_i$$

显然

$$\begin{aligned} WPL(T) - WPL(T') &= w_k \cdot l_k + w_1 \cdot l_1 - w_k \cdot l_1 - w_1 \cdot l_k \\ &= (w_k - w_1) \cdot (l_k - l_1) > 0 \end{aligned}$$

说明树T'比T具有更短的WPL，与T为最优树的前提矛盾。

因此必定有 $$l_1 = \max_i \{l_i\}$$

清華大学 Tsinghua University

Huffman树

可证，w_1必定存在兄弟结点。

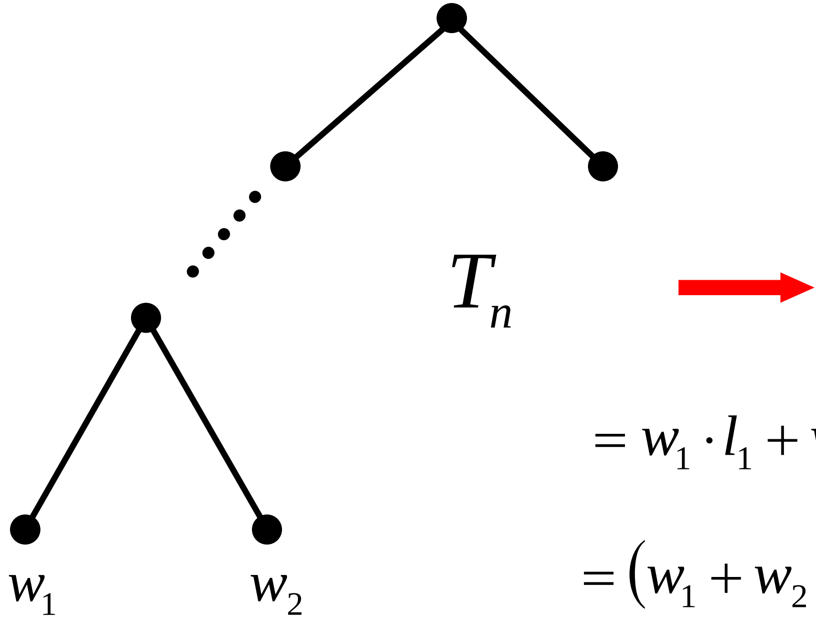
否则，其父亲结点路径更短

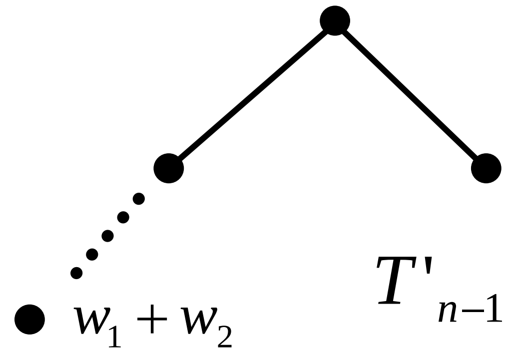
- 将w_1赋值给其父亲结点，将得到更短的WPL

那么，其兄弟结点是谁？

必定是w_2！为什么？

故，最优树T中，权最小的两个结点必然是兄弟结点

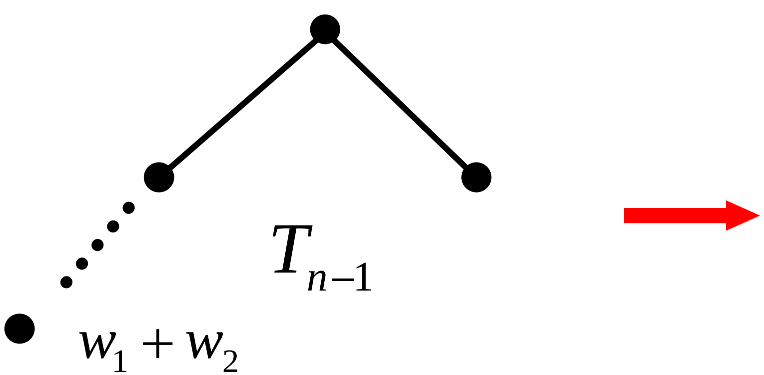


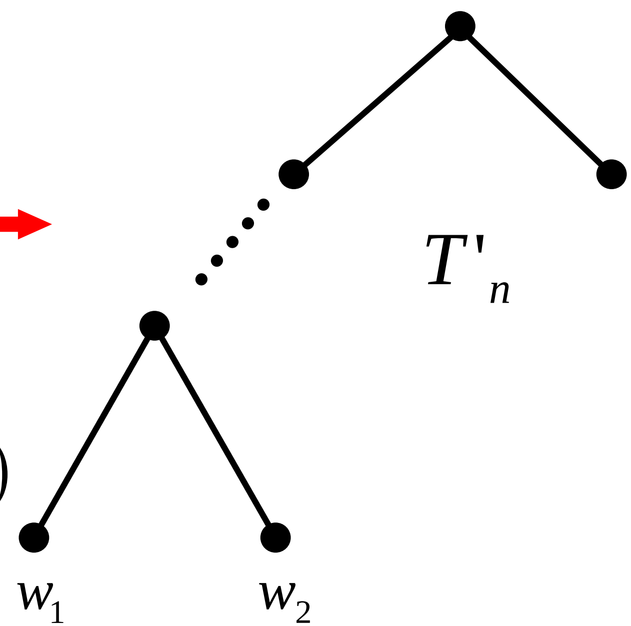


$$= w_1 \cdot l_1 + w_2 \cdot l_1 + \sum_{i \neq 1,2} w_i \cdot l_i$$

$$= (w_1 + w_2) \cdot (l_1 - 1) + \sum_{i \neq 1,2} w_i \cdot l_i + (w_1 + w_2)$$

$$= WPL(T'_{n-1}) + (w_1 + w_2)$$





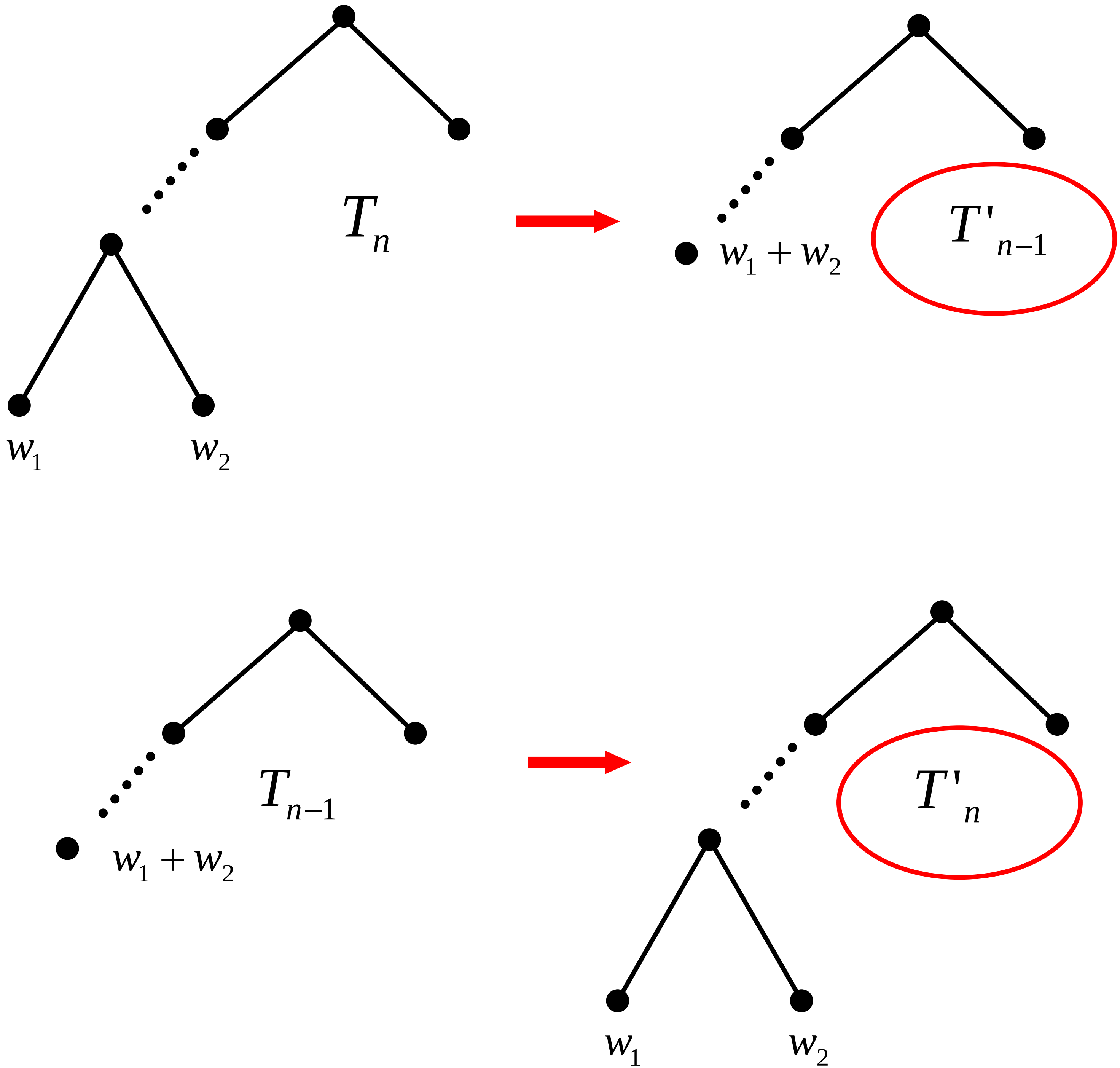
$$WPL(T'_n) = WPL(T_{n-1}) + (w_1 + w_2)$$

Huffman树

$$WPL\left(T'_n\right) = WPL\left(T_{n-1}\right) + \left(w_1 + w_2\right)$$

ΛIIV ΛI IV

$$WPL\left(T_n\right) = WPL\left(T'_{n-1}\right) + \left(w_1 + w_2\right)$$



w_1
w_2
T_n
$w_1 + w_2$
T'_{n-1}
T_{n-1}
$w_1 + w_2$
T'_n
w_1
w_2

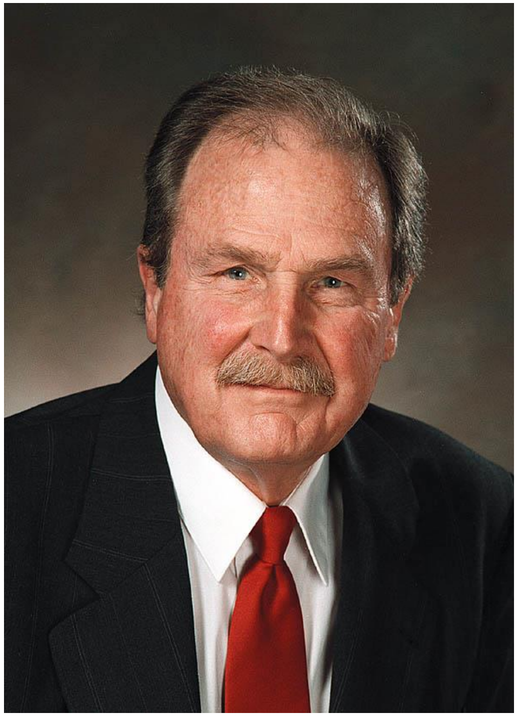
Huffman树

当只有两个结点时，由算法得到的树自然是最优树，由算法的分枝收缩展开过程可知，当结点数超过2时，得到的树仍然是最优树。

证毕！

Huffman树 – 小结

- Huffman树构造算法



David Albert Huffman

(1925 –1999)

Tsinghua University

主要内容

- 3.1 树的有关定义
- 3.2 基本关联矩阵及其性质
- 3.3 支撑树的计数
- 3.4 回路矩阵与割集矩阵
- 3.5 支撑树的生成
- 3.6 Huffman树
- **3.7 最短树**
- 3.8 最大分枝

Tsinghua University

最短树

- 供水问题：为给一些乡村联合供水，必须在各村之间建造管线系统，各村之间建造管线的成本已知，那么如何建造管线才能实现造价最低？

- 公司建网问题

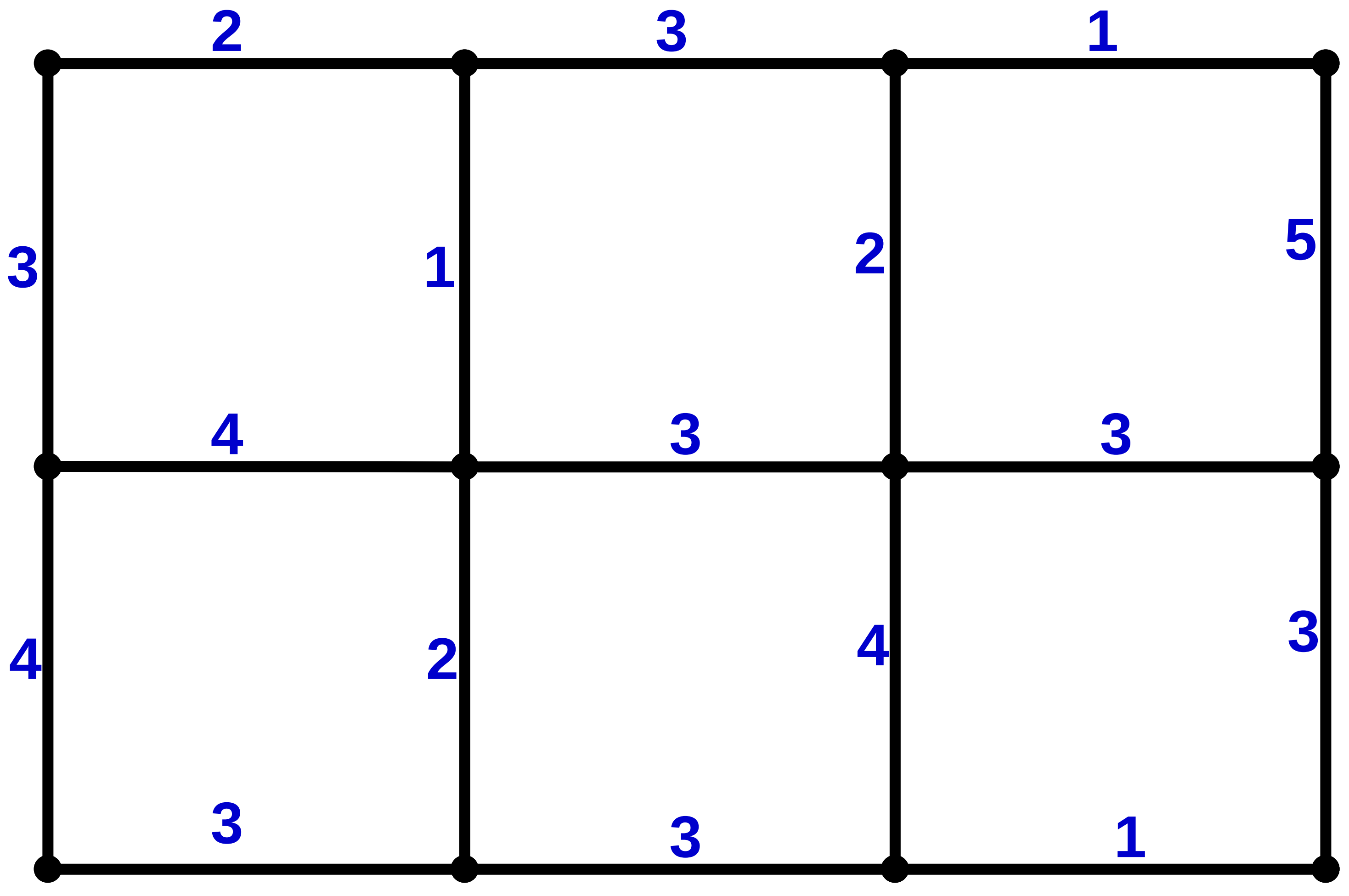
最短树

- 问题描述：在赋权连通图中，计算其总长最小的支撑树，称为最短树问题，或最小生成树问题。
  - Kruskal算法
  - Prim算法
  - 破圈法

最短树

- Kruskal(克鲁斯卡尔)算法：

- 算法思想：
  - 将边权值从小到大排序
  - 将边从小到大逐一加入T中，如果出现回路，则跳过当前边，直到出现树为止


2
3
1
3
1
2
5
4
3
3
4
2
4
3
3
3
1

最短树

思考：

- 对于连通图，逐边加入T(期间避免产生回路)的过程，是否一定会得到支撑树？

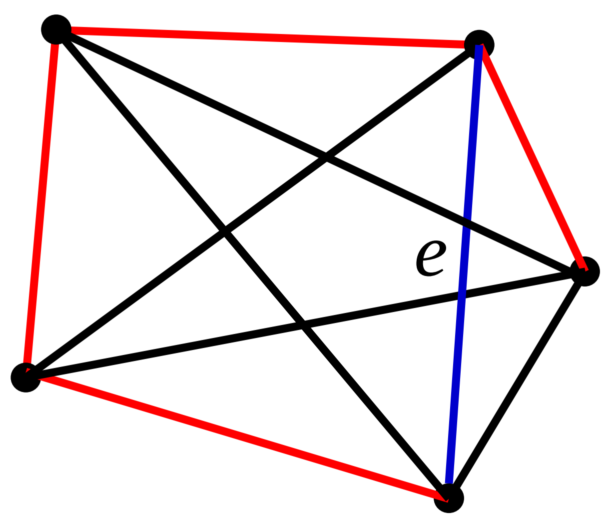
对于一个图，如果有n-1条边，不含回路

根据树的等价性质（4），可知必定是一棵树，而且是支撑树

最短树

- 定理3.7.1 $T=(V,E')$是赋权连通图$G=(V,E)$的最短树，当且仅当对任意的余树边$e\in E-E'$，回路$C^e(C^e\subseteq E'+e)$满足其边权

$$w(e)\geq w(a),\ a\in C^e(a\neq e)$$



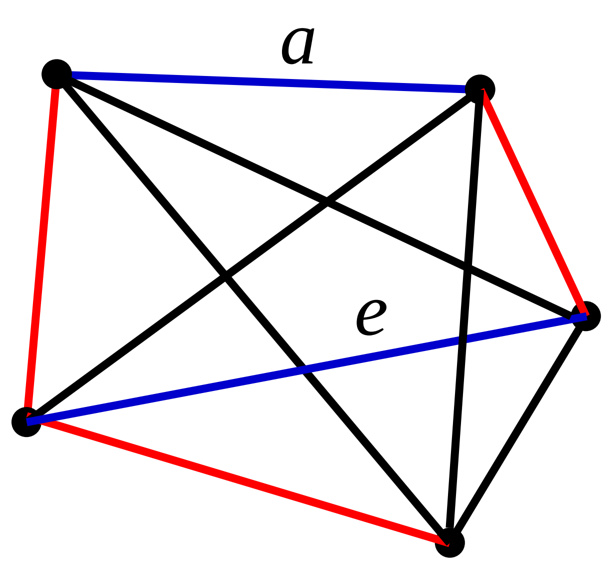
最短树

- 证明：（必要性）

  T为最短树 $\Rightarrow$ $w(e) \geq w(a),\ a \in C^e (a \neq e)$

  – 若T为最短树，则一定不存在余树边e，使得

  $w(e) < w(a),\ \ a \in C^e (a \neq e)$



- 充分性：
  - 树T任一条余树边e，都满足$w(e) \geq w(a), a \in C^e (a \neq e)$

    ⇒ T一定是最短树
  - 假设T不是最短树，则应该存在最短树T'不同于T
  - 则T'中一定存在树枝边e是树T的余树边，且e和树T的树枝边可构成回路C^e。且根据已知条件，e为C^e中最长边
  - 考察T'中，e所对应的割集S，它与C^e除e是公共边外，至少还会存在一条公共边a，且$w(e) \geq w(a)$
  - 构造树 $T^* = T' - e + a$，则T^*应该是比T'更短的树，矛盾！
  - 因此，如果存在最短树T'，它不能和树T有不同的边
  - 也就是说，树T一定就是最短树 ∎

最短树

- 定理3.7.1 $T=(V,E')$是赋权连通图$G=(V,E)$的最短树，当且仅当对任意的余树边$e\in E-E'$，回路$C^e(C^e\subseteq E'+e)$满足其边权

$$w(e)\geq w(a),\ a\in C^e(a\neq e)$$

该定理保证了**Kruskal**算法的正确性！

最短树

- Kruskal(克鲁斯卡尔)算法：

a. T←Φ

b. 当|T|<n-1且E≠Φ时，

Begin

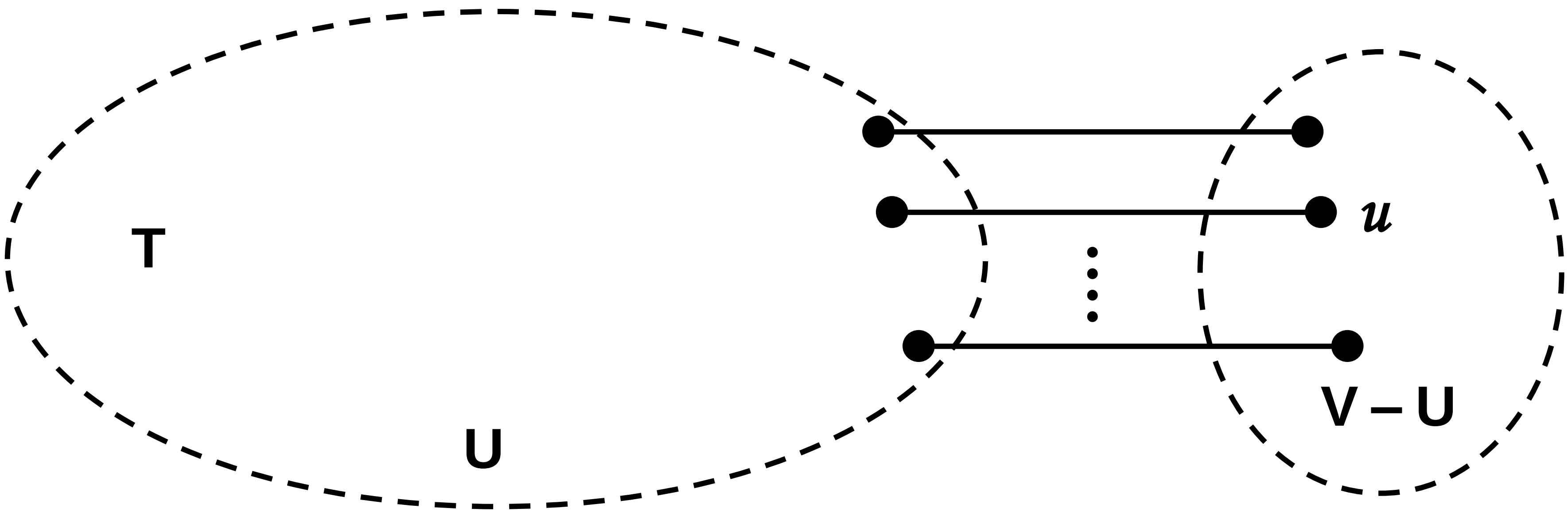
1. e ←E中最短边
2. E ←E-e
3. 若T+e无回路，则T ←T+e

End。

c. 若|T|<n-1，则原图为非连通图，否则T为最短树

最短树

- Prim算法思想：
  - 选取初始结点v，构成集合U，其余结点为$V-U$
  - 选取$V-U$中距离U最近的结点u，并入集合U，并将相应的边并入树T
  - 直到所有结点都进入U



最短树

- Prim(普林)算法：

$$t \leftarrow v_1,\ \ T \leftarrow \phi,\ \ U \leftarrow \{t\}$$

$$while\ \ U \neq V\ \ do$$

$$begin$$

$$w(t,u) = \min_{v \in V-U} \{w(t,v)\}$$

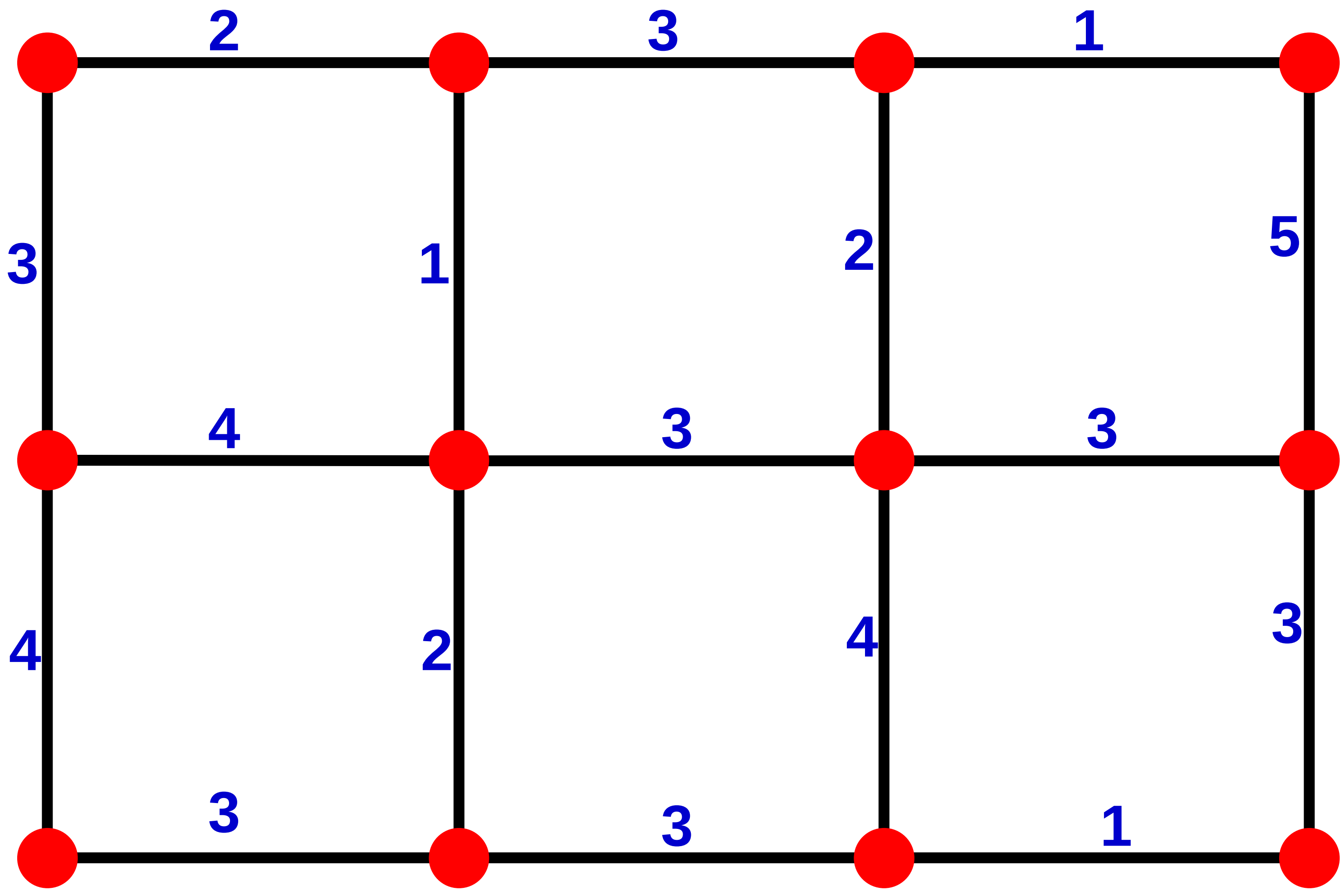
$$T \leftarrow T + e(t,u)$$

$$U \leftarrow U + u$$

$$for\ \ v \in V-U\ \ do$$

$$w(t,v) \leftarrow \min\{w(t,v), w(u,v)\}。$$

$$end。$$


2
3
1
3
1
2
5
4
3
3
4
2
4
3
3
3
1

最短树

- 定理3.7.3 设U是赋权连通图$G=(V,E)$的结点真子集，e为二端点分跨在U与$V-U$的最短边，则G中最短树T一定包含e

证明：假如最短树T不包含e，则$T+e$必定有包含e的唯一回路。由于e分跨在U与$V-U$，则回路中必定存在e'也分跨在U与$V-U$，且e比e'短。

则 $T'=T-e'+e$ 仍然为树，但是比T短。

与前提矛盾。 证毕！

Tsinghua University

最短树

- 定理3.7.4 Prim算法的结果是得到赋权连通图G的一棵最短树。

证明：首先证明它是一棵支撑树。

– 采用归纳法：

– 初始时，$U=\{v_1\}$，T=Φ，显然T为U导出的树

– 设$|U|=i$ 时，T为U导出的树，则边数为$i-1$

– 则当U中增加一个结点u时，T中增加一条与u相关联的边(边的另一端点为T中结点)，此时T结点数为$i+1$，边数为i，故为导出树

清華大學 Tsinghua University

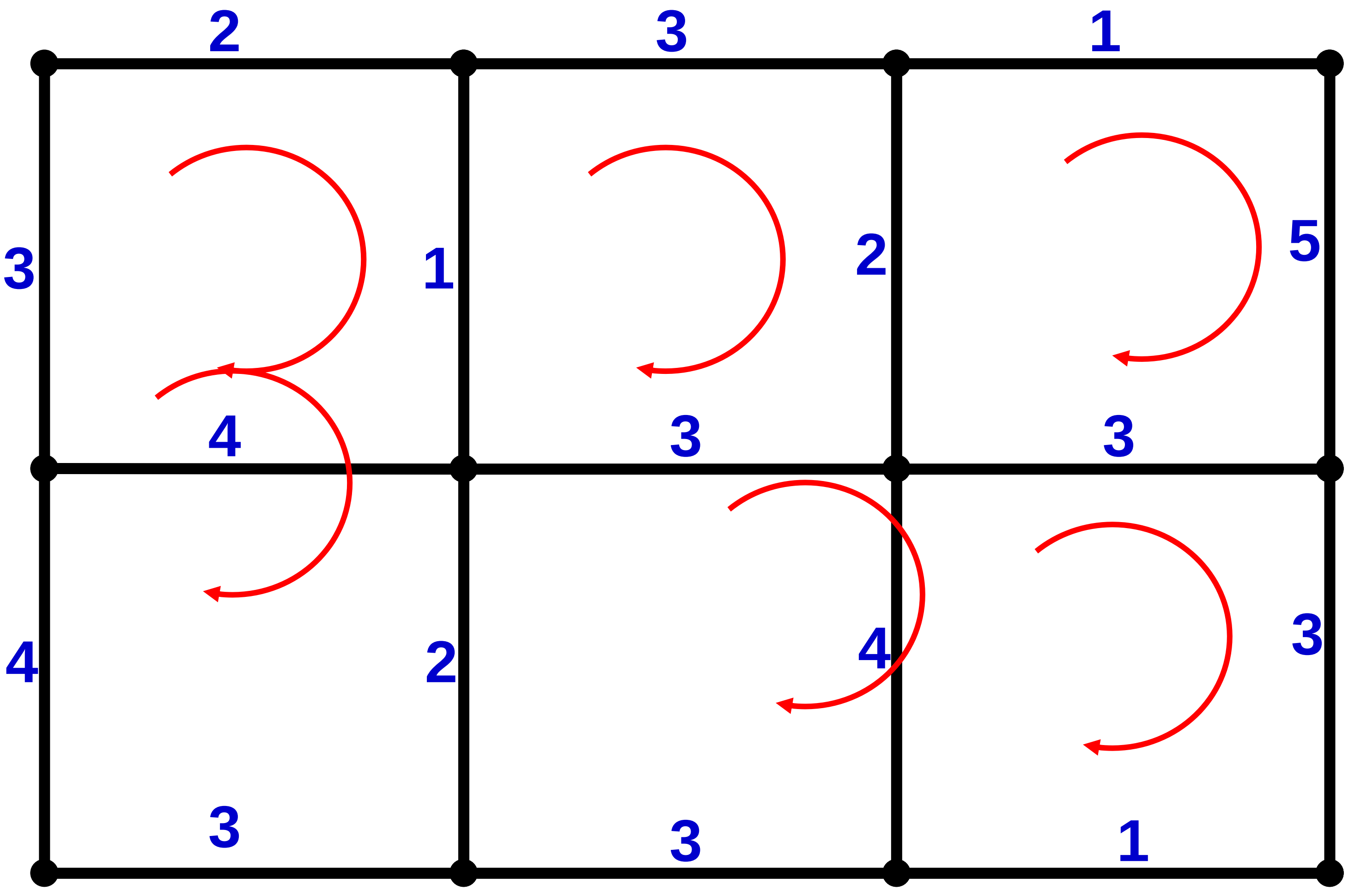
最短树

- 故最终T将是G的支撑树。
- 再证：T为最短树

（证明留做选作证明题）

最短树

- 破圈法：
  - Kruskal算法是在构造过程中尽量避免出现“圈”
    - 见圈避圈（避圈法）
  - 破圈法－见圈破圈



算法结束!

最短树

- 小结：
  - Kruskal 算法
    - 对稀疏图比较合适
  - Prim算法
    - 对稠密图比较合适
  - 破圈法
    - 对手工计算比较合适

思考：如何得到最大生成树？

主要内容

- 3.1 树的有关定义
- 3.2 基本关联矩阵及其性质
- 3.3 支撑树的计数
- 3.4 回路矩阵与割集矩阵
- 3.5 支撑树的生成
- 3.6 Huffman树
- 3.7 最短树
- **3.8 最大分枝**

清華大學 Tsinghua University

作业

- 课后 14题
- 选作题：

  求证：Prim算法的结果是得到赋权连通图G的一棵最短树。

- 下次课：习题课