《离散数学》

10-树 (Tree)

杨启哲

上海师范大学信机学院计算机系

2024 年 11 月 11 日

主要内容

- 树的基本概念
- 最小生成树的计算
- Huffman 树

树的基本概念

树的概念

我们赖考察一类特殊的无向图：

定义 1 **[树].**

给定一个无向图 $G = (V, E)$，如果其不含任何回路，我们称其为林，如果 G 是连通的，则称其为树。

定义 2 **[树的顶点度数].**

树的顶点度数为 1 的顶点称为叶子，其余顶点称为内部顶点。

树上的边是割边

回顾一下割边的定义：

定义 3 **[割边].**

设 $G=(V,E)$ 是一个图，如果 $e\in E$，且 $G-e$ 的连通分分支个数严格大于 G，则称 e 为 G 的一个割边。

定理 4.

设 $G=(V,E)$ 是一个图，$e=(u,v)\in E$，则 e 是 G 的割边当且仅当 e 不属于 G 的任何一个回路上。

推论 5.

设 $G=(V,E)$ 是一棵树，则 G 中的任何一条边都是割边。

割边判定的证明

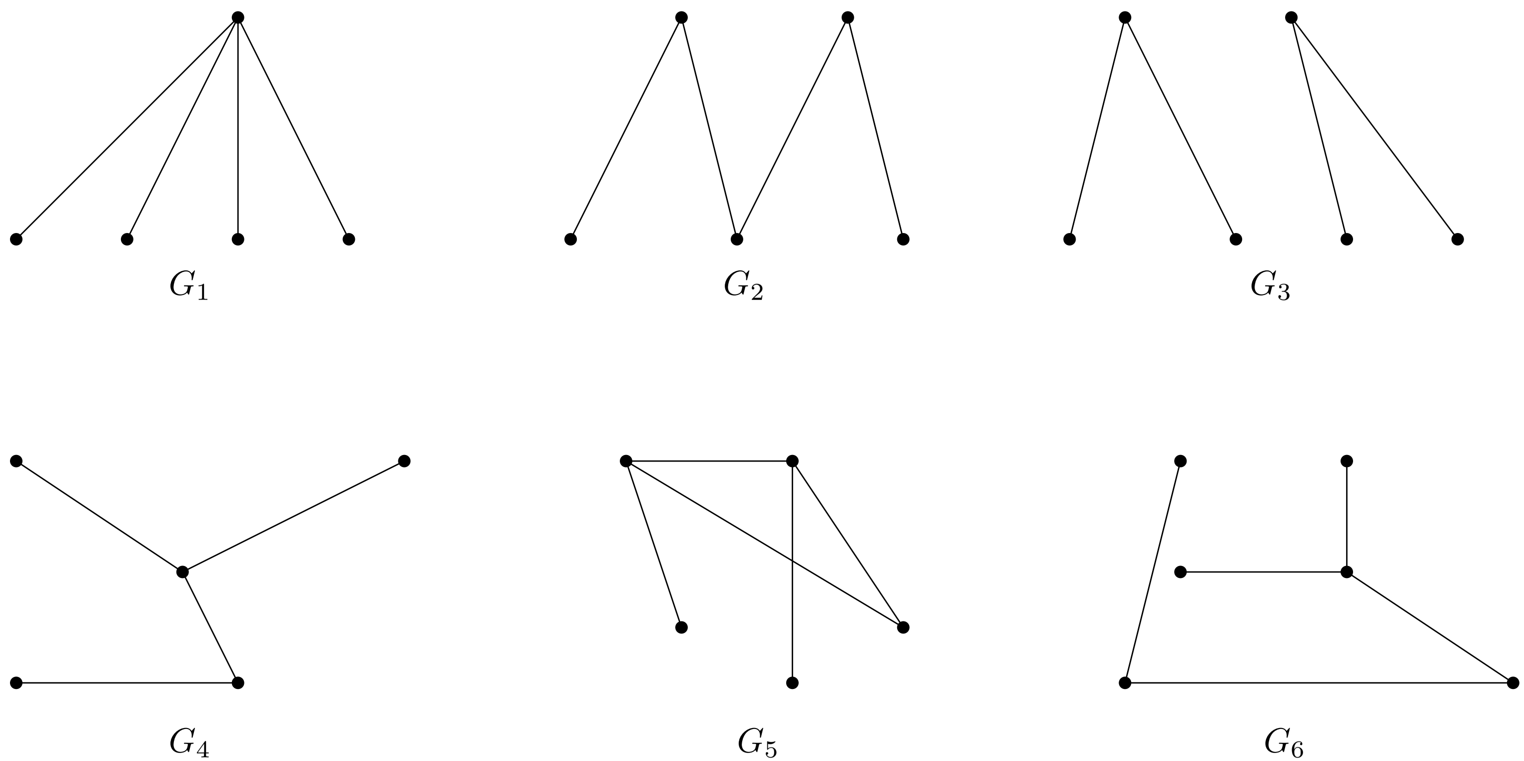
定理4的证明.

$\Rightarrow$ 由于 e 是割边，因此 $G - e$ 的连通分支个数严格大于 G，即 u 和 v 一定属于两个不同的分支。如果 e 属于 G 中某条回路，则依旧会存在一条 u 到 v 的通路，从而 u, v 依旧在一个连通分支里，矛盾。。

$\Leftarrow$ 假设 e 不属于 G 的任何一个回路上，我们考虑 $G - e$ 的连通分支个数。如果 e 不是割边，则 $G - e$ 和 G 的连通分支个数是相同的，因此 u, v 一定在同一个连通分支里，即在 $G - e$ 中存在一条 u 到 v 的通路 π，从而 $\pi + e$ 将成为一条 G 中的回路，矛盾。

$\square$

树的一些例子



- G_1，G_2，G_4, G_5 都是树。
- G_1 是林，G_5 不是树或者林。

树的等价定义

定理 6.

设 T 是结点数 $n \geqslant 2$ 的树，下列定义是等价的：

1. T 连通且无回路。
2. T 连通且每条边都是割边。
3. T 连通且有 $n-1$ 条边。
4. T 有 $n-1$ 条边且无回路。
5. T 的任意两节点间有唯一通路。
6. T 没有回路，但在任意两条边间加一条边都会产生回路。

$1 \Rightarrow 2 \Rightarrow 3$ 的证明

证明.

$1 \Rightarrow 2$ 由于 T 连通，并且没有任何一条回路，从而由定理4，T 中的每一条边都是割边。

$2 \Rightarrow 3$ 我们对 n 进行归纳，令 $V(T)$，$E(T)$ 分别表示 T 的顶点集和边集的个数。
$V(T) = 2$ 时我们有 $E(T) = 1$，从而命题成立。
假设命题对 $V(T) \leqslant k$ 成立，考察 $V(T) = k + 1$ 的情况。由于 T 连通，因此 T 中的任一条边都是割边，即 $T - e$ 将其分解成了两个连通分支 T_1, T_2。注意到 T_1, T_2 都是树，并且 $V(T_1), V(T_2) \leqslant k$，从而有归纳假设：

$$E(T) = E(T_1) + E(T_2) + 1 = V(T_1) - 1 + V(T_2) - 1 + 1 = V(T) - 1 = k$$

即命题对 $k + 1$ 也成立。

□

$3 \Rightarrow 4$ 的证明

证明.

$3 \Rightarrow 4$ 反设 T 存在一条回路 C，不妨记为：

$$C = v_0 v_1 \cdots v_k v_0$$

由于 T 是连通的，从而不在 C 上的任何顶点一定存在跟一条跟 C 中顶点的边

从而 T 至少有：

$$E(T) \geqslant V(T) - V(C) + k = n - k + k = n$$

矛盾！

$\square$

$4 \Rightarrow 5$ 的证明

上海师范大学 Shanghai Normal University

证明.

存在性 先证明对于树种的任何两点 u, v，其是连通的。反设其不连通，则 u, v 分属两个不同的连通分支 T_1, T_2，并且由假设 T_1, T_2 也不存在回路，由上述证明可知 T_1 和 T_2 恰好有 n_1, n_2 条边，从而 T 有:

$$E(T) = E(T_1) + E(T_2) = V(T_1) - 1 + V(T_2) - 1 = V(T) - 1 = n - 2 < n - 1$$

与条件矛盾。

唯一性 再证唯一性。反设存在两条不同的通路 π_1, π_2，则 $\pi_1 + \pi_2$ 中至少会存在一条回路。从而任意两节点间的道路存在且唯一。

$\square$

$5 \Rightarrow 6 \Rightarrow 1$ 的证明

证明.

$5 \Rightarrow 6$ 该方向是显然的。

$6 \Rightarrow 1$ 只需要证明 T 是连通的，反设其不连通，即存在两个不同的连通分支 T_1, T_2。令 v_1 是 T_1 中的一个顶点，v_2 是 T_2 中的一个顶点，则由连通分支的定义，v_1, v_2 之间不存在通路，现增加一条 (v_1, v_2) 的边，并不会产生新的回路，从而与假设矛盾。

$\square$

简单总结

1. 树是极小的连通图，减少一条边就不再连通。
2. 树是极大的连通无回路图，增加一条边就会产生回路。

树的其他性质

定理 7.

给定一棵树 T，其一定存在树叶节点。

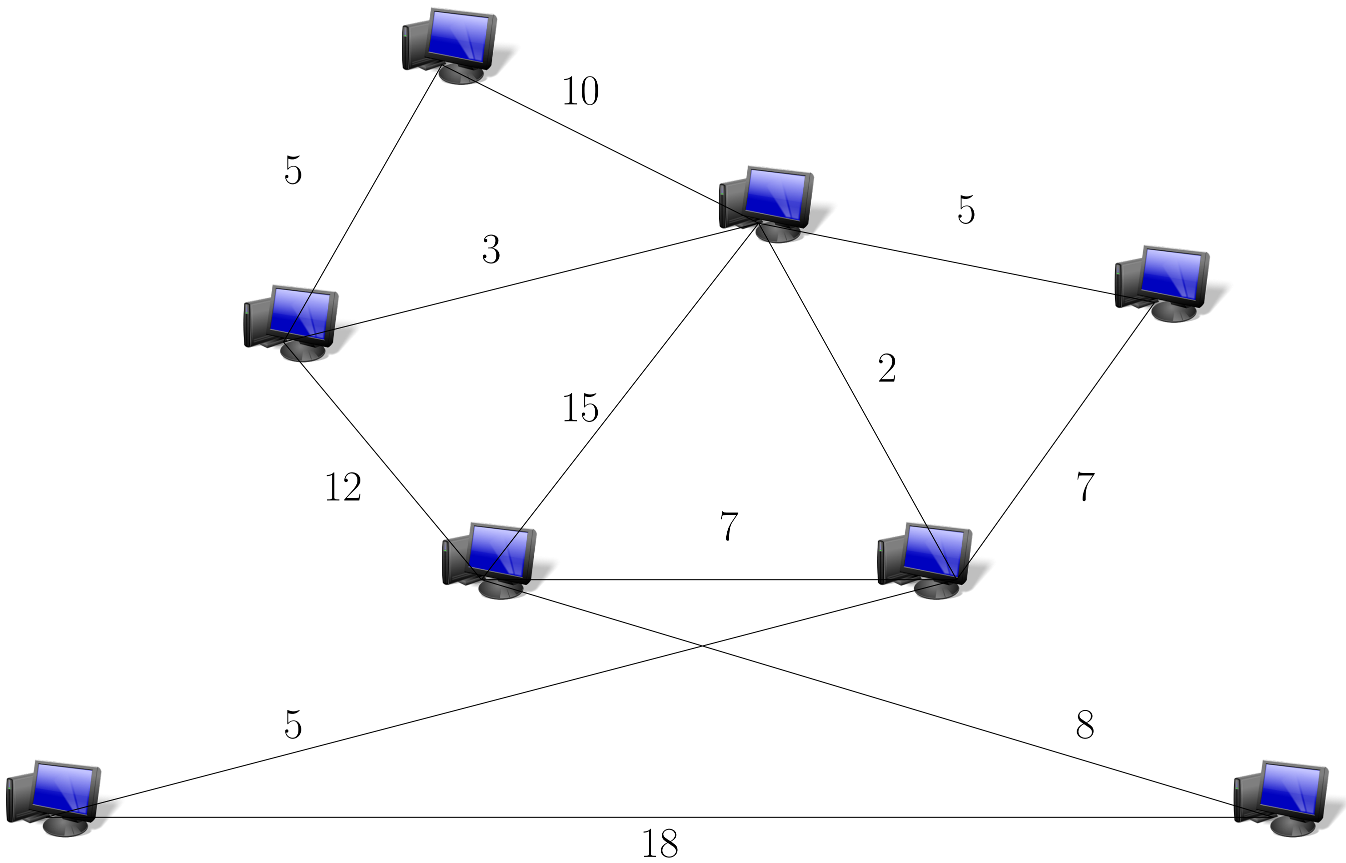
证明. 由于 T 是连通的，因此任何一个顶点的度数至少为 1，反设其没有任何树叶节点，则 T 中的任何一个顶点的度数至少为 2，从而：

$$|V(T)| - 1 = |E(T)| = \frac{1}{2} \cdot \sum_{v \in V(T)} \deg(v) \geqslant |V(T)|$$

产生矛盾。 □

最小生成树的概念 (I)

假设你现在需要将一组计算机连接起来，你需要在这些计算机之间铺设光纤，每条光纤的长度是不同的，你需要找到一种铺设方案，使得铺设光纤的总长度最小。



最小生成树的定义

我们可以将上述例子抽象成一张有权重的无向图，则问题转化为：

问题 8.

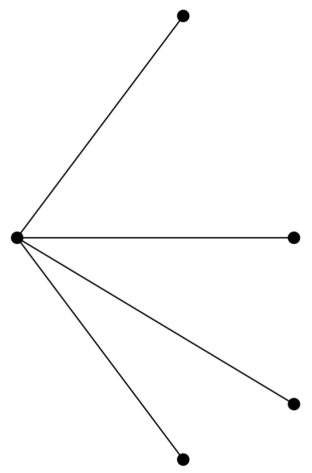
给定一个含权重的无向图 $G = (V, E, \omega)$，我们需要选出足够多的边集 T，使得在其子图 (V, T) 中任何两个顶点之间都有一条路径，且这些边的权重之和最小。

这其实就是在求 G 的一个生成子图 T，满足 T 是最小的连通的图，显然 T 会是一棵树。我们称这样的 T 为图 G 的生成树。

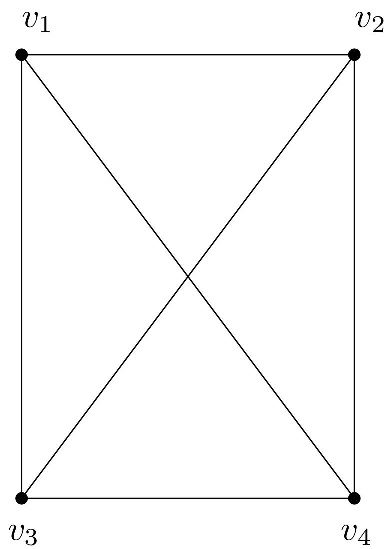
定义 9 **[生成树，*spanning tree*].**

给定一个无向图 $G = (V, E)$，如果 T 是 G 的生成子图并且是一棵树，则称其为图 G 的生成树。特别的，如果图的边是带权重的，我们则称所有边权重之和最小的生成树为图 G 的最小生成树。

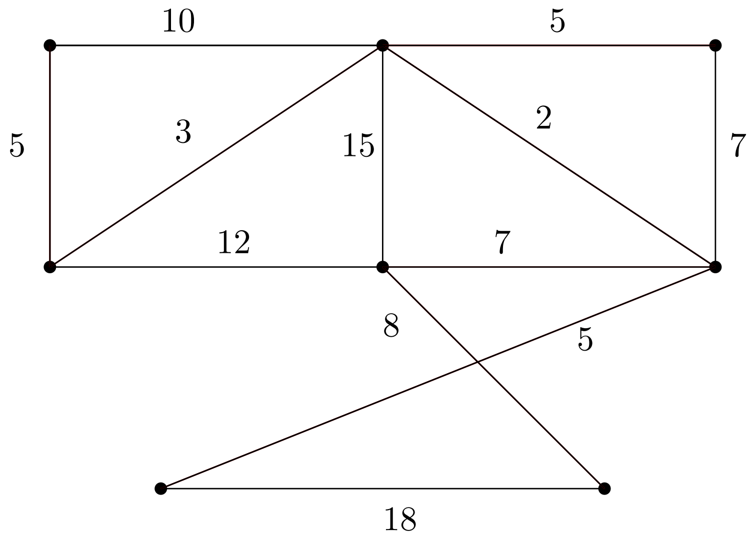
生成树的简单例子



G_1



G_2



G_3

- G_1 的生成树是什么?
- G_2 有多少不同的生成树?
- G_3 的最小生成树是什么?

生成树的基本性质

定理 10.

任何无向连通图都存在生成树。

证明.

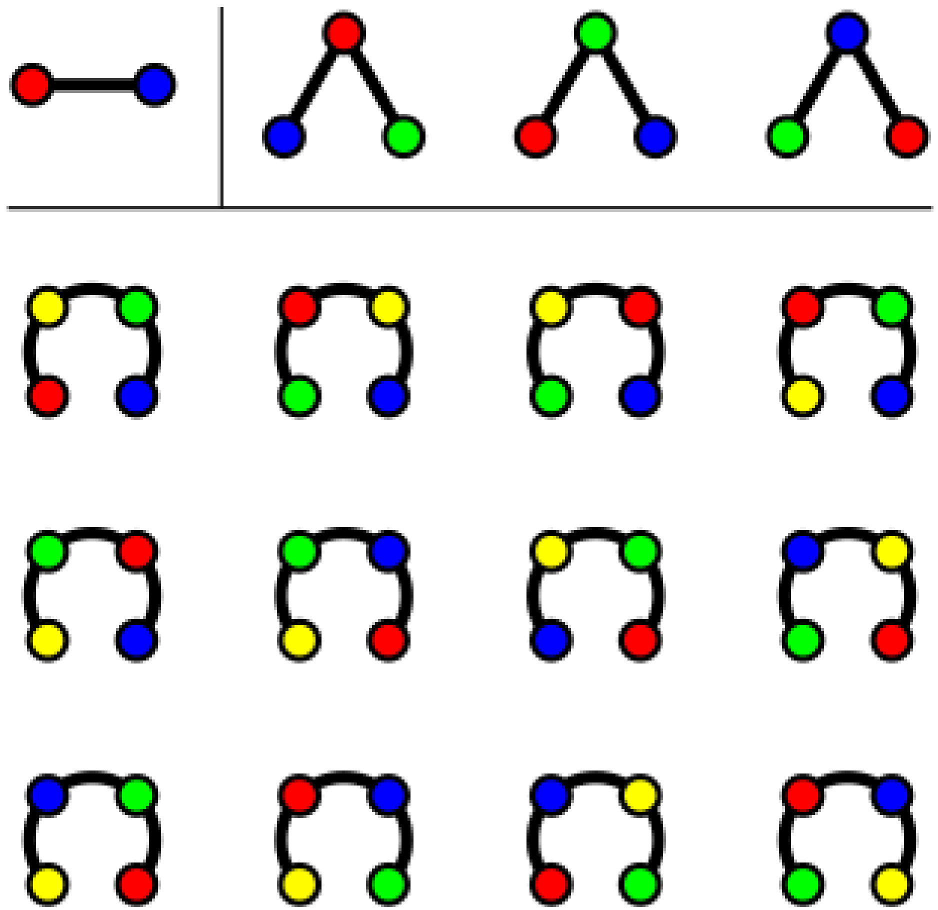
- 若图 G 不存在回路，则 G 本身就是一棵树。
- 如果 G 存在一个回路 π，则删去 π 中的一条边依旧是连通的；重复该过程直至没有新的回路。显然最后生成的图 G' 便是 G 的一棵生成树。

□

Cayley 公式 (I)

假设我们现在对树上每个顶点做个标号，即不同的标号的顶点视作不同的顶点，

这样 n 个顶点的树同构意义下不同的树的个数有多少个?



Cayley 公式 (II)

定理 11 [Cayley's formula].

给定一个正整数 n，其不同的标号的树的个数为 n^{n-2}。

补充说明

实际上 Cayley 公式计算的便是 n 个不同顶点的完全图的生成树的个数。

Cayley 公式有很多种证明，我们课上就介绍一种最为简单的证明-构造映射。

首先我们令这些所有 n 个顶点的生成树组成集合 $\mathcal{V}_n$.

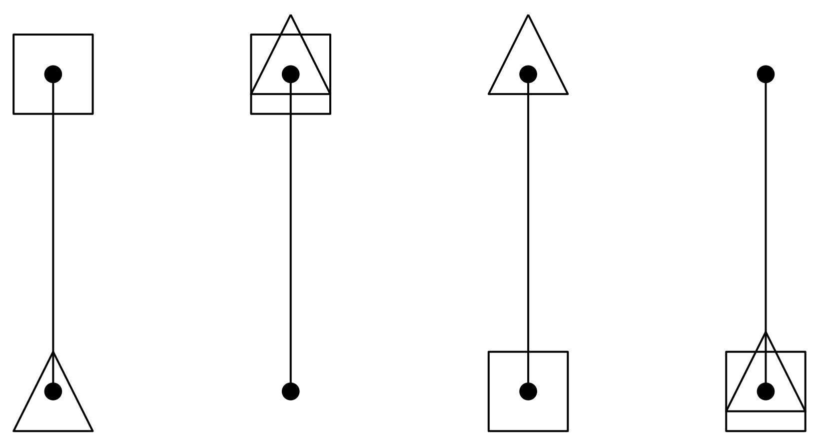
Cayley 公式的证明 (I)

考察一个 n 个点的生成树，我们给其加上两个特殊的标记：

- 一个顶点标记上 □.
- 一个顶点标记上 △

这里我们允许一个顶点上既有 □ 也有 △. 记由这些带有 □ 和 △ 标记的生成树组成的集合为 $\mathcal{T}_n$，显然我们有：

- $|\mathcal{T}_n| = n^2 \cdot |\mathcal{V}_n|$.



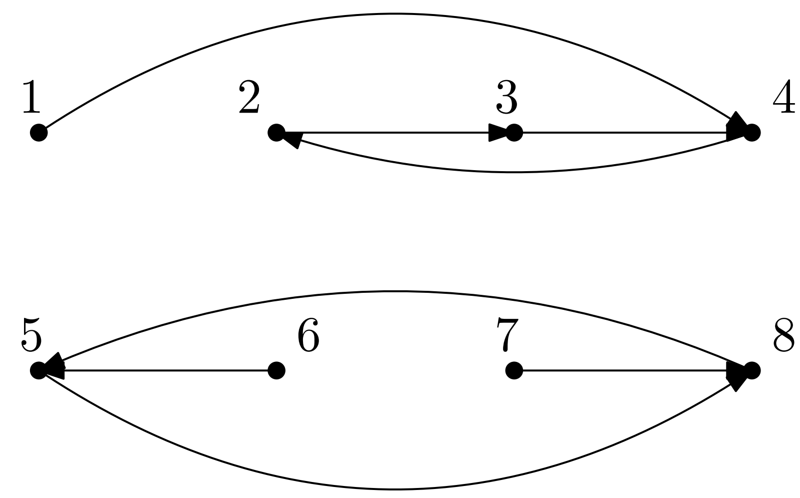
Cayley 公式的证明 (II)

注意到 $\{1,2,\ldots,n\}\to\{1,2,\ldots,n\}$ 的不同映射有 n^n 个，记这些映射组成的集合为 $\mathcal{F}_n$，下面我们给出一个 $\mathcal{F}_n$ 到 $\mathcal{T}_n$ 的双射 f，从而完成 Cayley 公式的证明。

双射 f 的构建： 对于 $\mathcal{F}_n$ 中的任何一个映射 $\mathfrak{f}$，我们设计如下的有向图 $G_{\mathfrak{f}}$:

- 图 $G_{\mathfrak{f}}$ 共有 n 个顶点，分别代表 $1,2\ldots,n$.
- 如果 $\mathfrak{f}(k)=j$，则在图中增加一条由 k 到 j 的有向边。

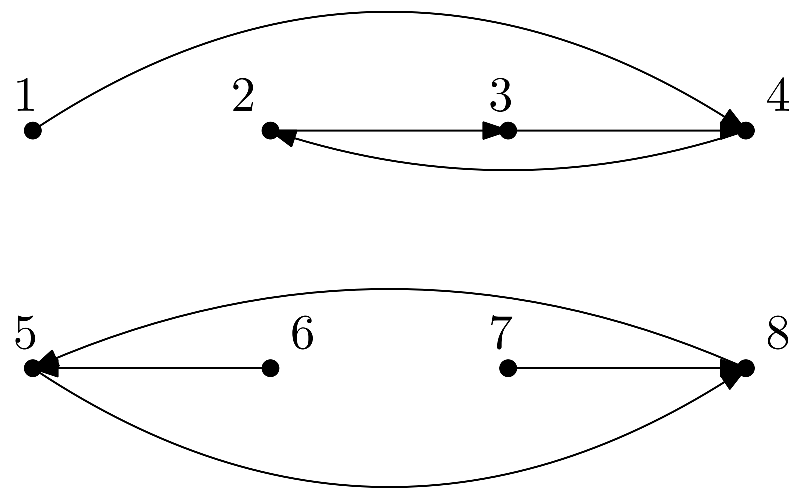
1	2	3	4	5	6	7	8
4	3	4	2	8	5	8	5



Cayley 公式的证明 (III)

图 G_f 的性质： 显然图 G_f 具有如下性质：

- 一共有 n 条边。
- 假设其有 k 个不同的单向连通分支，则每个分支上一定都会存在一个初级回路。

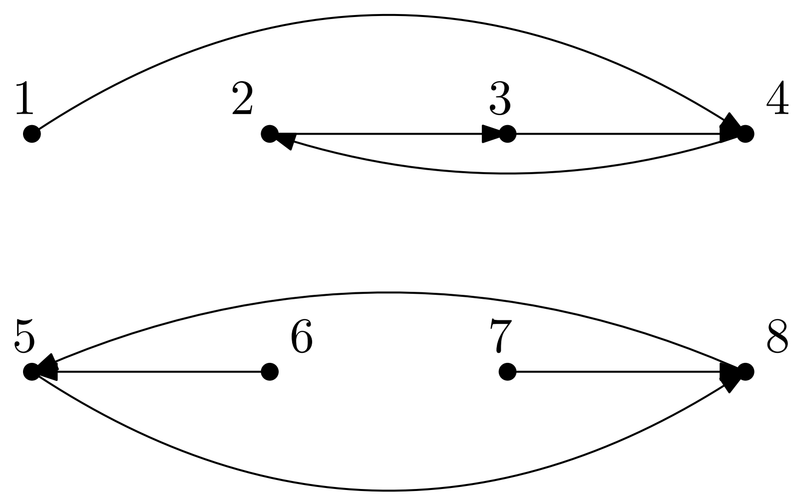


Cayley 公式的证明 (IV)

现在我们取每个单向连通分支里极大的初级回路上的顶点，并且令其从小到大排列为：

$$V_{\mathfrak{f}} = \{v_1,\ v_2, \cdots, v_k\}$$

则我们可以证明 $\mathfrak{f}$ 在 $V_{\mathfrak{f}}$ 的限制下存在一个双射。



2	3	4	5	8
3	4	2	8	5

Cayley 公式的证明 (V)

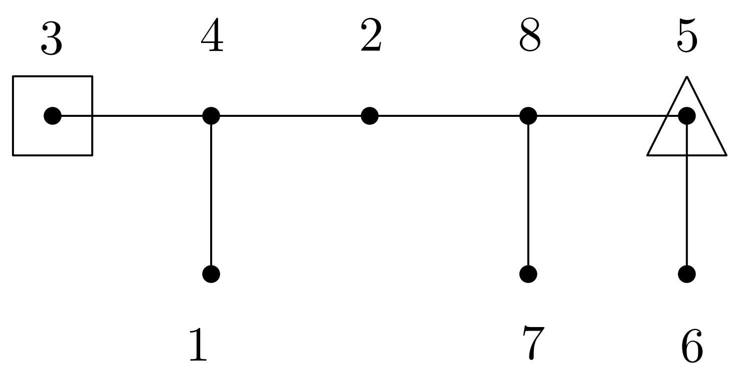
上海师范大学 Shanghai Normal University

现在我们来给出我们的映射 f，即 $\mathfrak{f}$ 对应的生成树。

- 从 $\mathfrak{f}(\nu_1)$ 开始，依次联结 $\mathfrak{f}(\nu_2),\ldots \mathfrak{f}(\nu_k)$.
- 在 $\mathfrak{f}(\nu_1)$ 上标记 □，在 $\mathfrak{f}(\nu_k)$ 上标记 △.
- 对于不在 $\mathfrak{f}(V_{\mathfrak{f}})$ 上的点，如果 $\mathfrak{f}(\nu)=u$，则在构造的生成树中连接一条 u 到 ν 的边。

2	3	4	5	8
3	4	2	8	5

1	2	3	4	5	6	7	8
4	3	4	2	8	5	8	5



Cayley 公式的证明 (VI)

显然我们构造的映射 f 满足:

- 每个映射 f 产生一个 $\mathcal{T}_n$ 中不同的生成树。
- 每个 $\mathcal{T}_n$ 中不同的生成树对应 $\mathcal{F}_n$ 中不同的映射。

因此 fs 是一个双射，从而我们有:

$$|\mathcal{V}_n| = |\mathcal{T}_n| = \frac{|\mathcal{F}_n|}{n^2} = n^{n-2}$$

$\square$

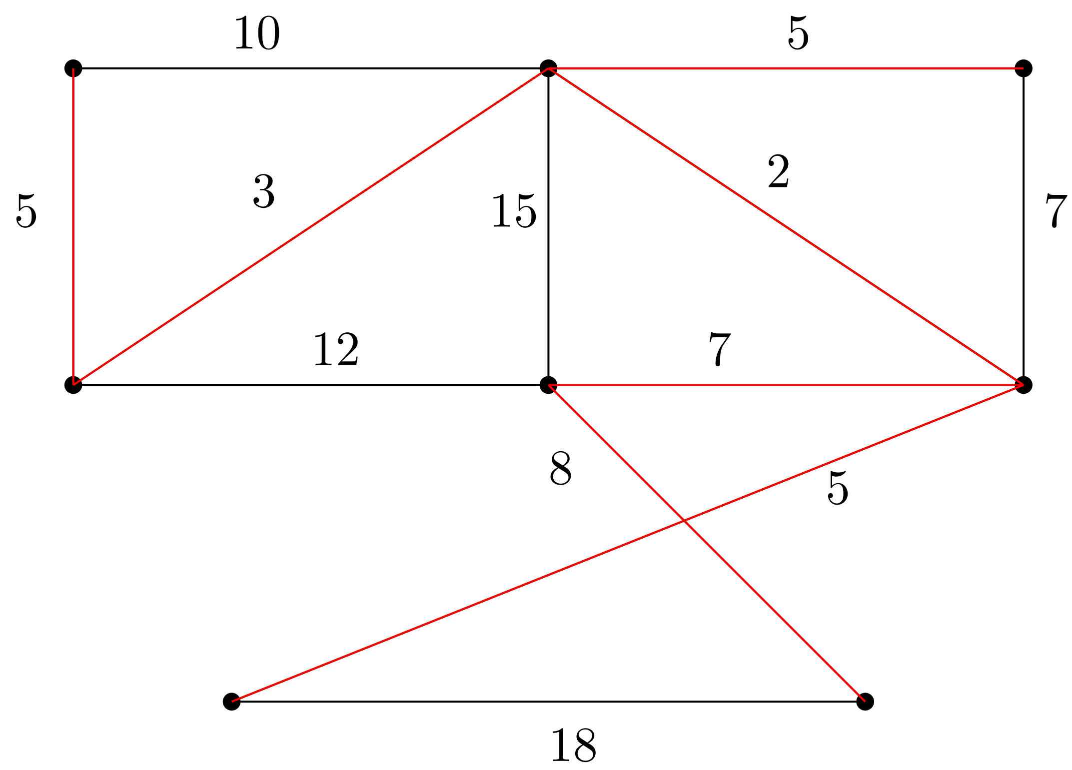
最小生成树的计算

最小生成树-例子

上海师范大学 Shanghai Normal University

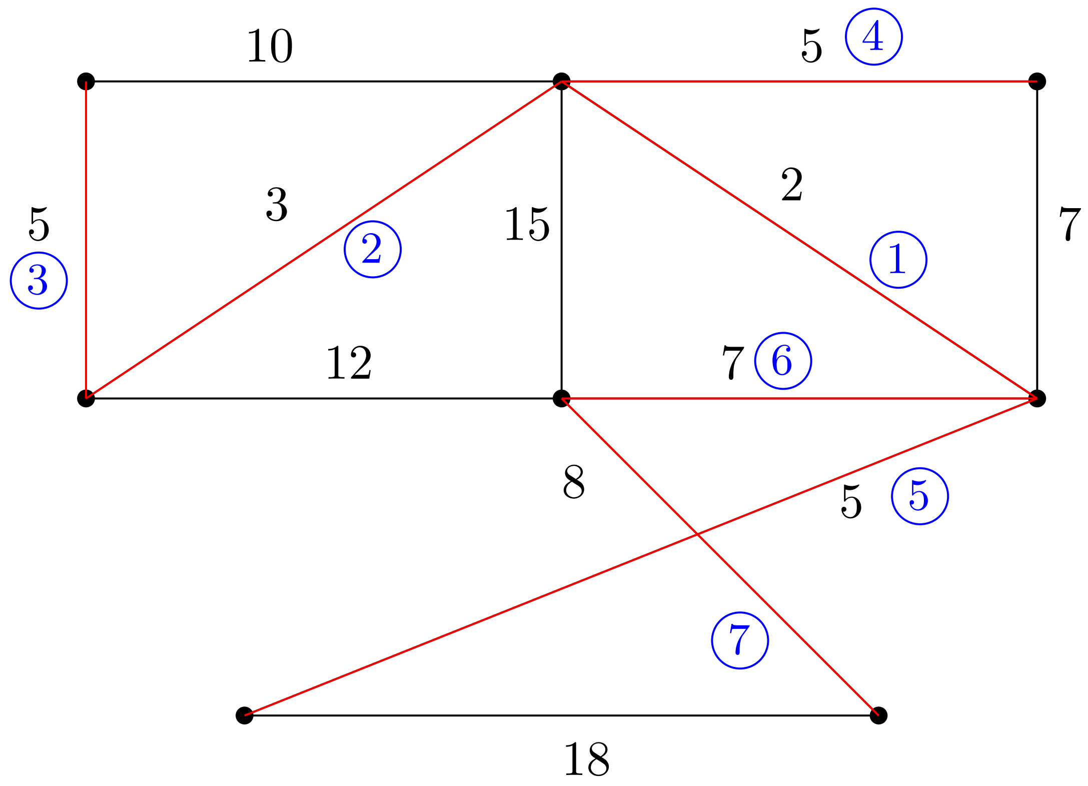
回顾一下之前的例子:

- 其中红色边集即为对应的最小生成树。



最小生成树-贪心策略

- 既然要求最小权重的生成树，那我每次选取当前权重最小的边, 只要保证不成圈就行。



这就是 Kruskal 算法。

Kruskal 算法

算法：Kruskal

输入： 含权连通无向图 $G = (V, E)$，$V = \{1, 2, \ldots, n\}$

输出： G 生成的最小生成树所组成的边集 T

按非降序的权重 E 进行排序，得到 $E = \{e_1, e_2, \ldots, e_m\}$
$T = \emptyset$
for $i = 1$ to m **do**
    if $T \cup \{e_i\}$ 不成圈 **then**
        $T = T \cup \{e_i\}$
return T

Kruskal 算法正确性分析

定理 12.

Kruskal 算法能够正确的求出最小生成树。

证明. 令算法找到的生成树为 T，其边的加入顺序为 $\{e_1, e_2, \dots e_{n-1}\}$。设 T^* 为 G 的最小生成树，我们来证明 $\omega(T) = \omega(T^*)$。

定义 $T_k = \{e_1, e_2, \dots e_k\}$，我们对 k 归纳证明：$T_k$ 是某个最小生成树边集 T^* 的子集。初始情况是 $k = 0$，显然成立。假设 $\leqslant k-1$ 成立，考察 k 时的情况，我们有：

- 若 $e_k \in T^*$，则命题成立。
- 若 $e_k \notin T^*$，由归纳假设 $T_{k-1} \subseteq T^*$，从而 T^* 恰好包含了一个包括边 e_k 的回路，并且其满足存在一条边 $e' \in T^* \setminus T_k$ 使得：
  - $\omega(e') \geqslant \omega(e_k)$

  注意到如果 $\omega(e') > \omega(e_k)$，则可以构造一棵权重更小的树 $T^* - \{e'\} + \{e_k\}$，从而 $\omega(e') = \omega(e_k)$。由此可以证明 $T^{**} = T^* - \{e'\} + \{e_k\}$ 依旧是一颗最小生成树，从而我们有：$T_k \subseteq t^{**}$，即命题对 k 也成立。

Kruskal 算法的时间分析

上海师范大学 Shanghai Normal University

- Kruskal 算法对边排序需要 $O(m \log m)$ 的时间。
- 判断是否有圈可以利用并查集，一共至多执行 $2m$ 次 Find 操作和 $n-1$ 次 Union 操作，因此总共耗费 $O(m \log^* n)$ 的时间。
- 一共会往 T 里增加 $n-1$ 条边。

因此，算法的运行时间为 $O(m \log m) = O(m \log n)$.

定理 13.

Kruskal 算法可以在 $O(m \log n)$ 内求出 G 的最小生成树。

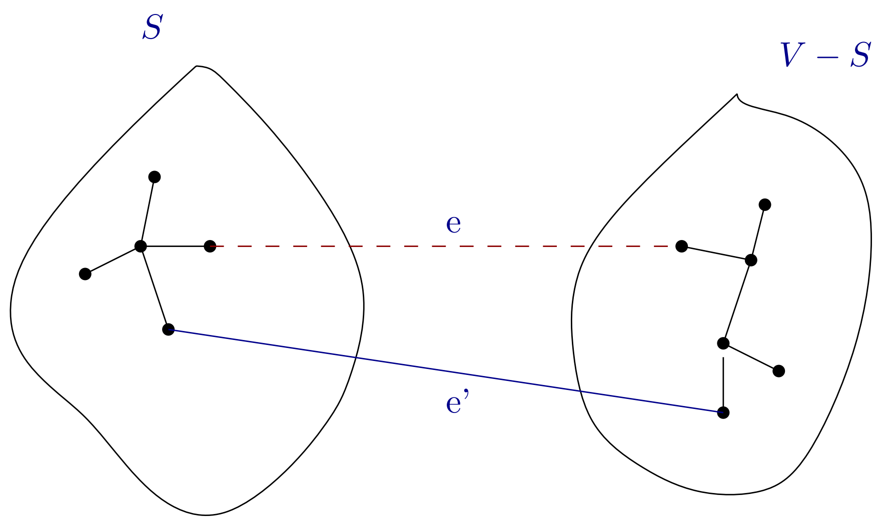
Kruskal 算法正确的另一视角-分割性质

我们换个角度再来理解下 Kruskal 算法的正确性。

假设为了构造最小生成树，我们已经选择了一些边，这些边将图上的顶点划分成了若干个部分，下面性质说明，跨越这些部分中的最短边也是某个最小生成树的一部分。

分割性质

设 $G=(V,E,\omega)$ 是一个含权重的连通无向图，X 是 G 的某个最小生成树一的部分，令 S 是 V 的一个子集，满足 X 中没有横跨 S 和 $V-S$ 的边，设 e 是 G 中连接 T 中的一个顶点和 $V-T$ 中的一个顶点的最短边，则 $X\cup\{e\}$ 是 G 的某个最小生成树的一部分。



最小生成树的另一算法-Prim 算法

算法：Prim

输入： 含权连通无向图 $G = (V, E)$，$V = \{1, 2, \ldots, n\}$

输出： G 生成的最小生成树所组成的边集 T

$T = \emptyset$，$X = \{1\}$，$Y = \{V\} - \{1\}$
for $y \leftarrow 2$ to n **do**
    if $(1, y) \in E$ **then**
        $n(y) = 1$ ▷ $n(y)$ 记录当前最短边的另一端点
        $c(y) \leftarrow \omega(1, y)$ ▷ $c(y)$ 记录当前最短边的权重
    else $c(y) \leftarrow \infty$
for $j \leftarrow 1$ to $n - 1$ **do**
    从 Y 中选取 $w(y)$ 最小的点 u
    $T = T \cup \{(u, n(u))\}$
    $X = X \cup \{u\}$，$Y = Y - \{u\}$
    for $w \in Y \wedge (y, w) \in E$ **do**
        if $\omega(y, w) < c(w)$ **then**
            $n(w) \leftarrow y$，$c(w) \leftarrow \omega(y, w)$
return T

Prim 算法分析

- 由分割性质，Prim 算法的正确性是显然的。
- Prim 算法与 Dijkstra 算法的流程基本相同，所以其复杂性是一样的，取决于优先队列的实现。
  - 如果使用普通数组，时间为 $O(n^2)$.
  - 如果使用 2 叉堆，时间为 $O(m \log n)$.

定理 14.

使用二分堆作为优先队列的实现时，Prim 算法可以在 $O(m \log n)$ 内求出 G 的最小生成树。

Huffman 树

根树的概念

我们现在来介绍有向图中一类特殊的树-根树。

定义 15.

若一颗有向树 T 满足存在一个顶点入度为 0，其余顶点入度为 1，则这样的有向树被称作根树。入度为 0 的节点一般称作根节点，没有出度的节点一般称作叶子节点，其余节点一般被称作内部节点。从根节点到任意顶点 v 的路径的长度称作 v 的层数（或者高度、深度），最大层数称作树的高度（深度）。

树中节点的关系

在一棵树中对于任两个节点 v_i 和 v_j：

- 如果存在 v_i 到 v_j 的一条路径，则称 v_i 是 v_j 的祖先节点，v_j 是 v_i 的后代节点。
- 如果存在 v_i 到 v_j 的一条边，则称 v_i 是 v_j 的父亲节点，v_j 是 v_i 的儿子节点。

特殊的根树

常见的根树我们会限制每个父节点的儿子个数，即：

定义 16 **[r 叉树].**

对于一个根树来说，如果每个非叶子节点的出度至多为 r，则称该树为 r 叉树：如果每个非叶子节点的出度恰好为 r，则称该树为 r 叉正则树；如果一个 r 叉正则树的所有叶子节点都在同一层，则称其为完全 r 叉正则树。

根树也有子树的概念：

定义 17 **[子树].**

对于一个根树 T，对于 T 中任何一个节点 v，称以 v 和其后代的导出子图 T_v 是 T 以 v 为根的子树。特别的，对于标定顺序的 2 叉树来说，每个分支点的两个儿子节点导出的子树分别称为该分支点的左子树和右子树。

2 叉树 (I)

2 叉树 (II)

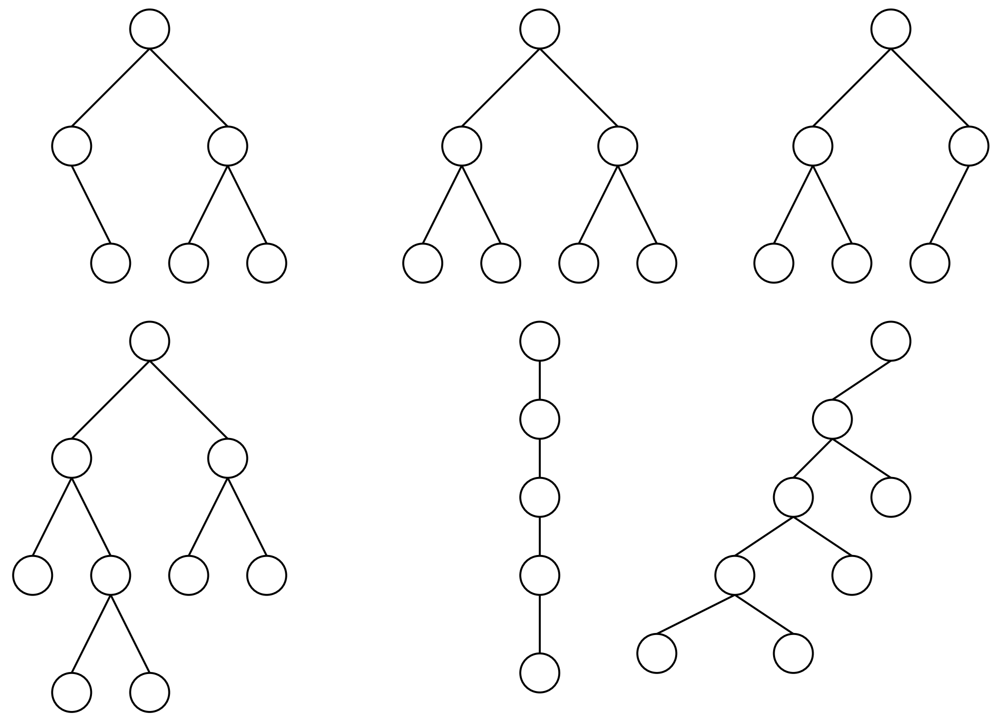
定义 18 **[2 叉树的另一种定义].**

2 叉树是顶点的一个有限集合，该集合或者为空，或者由一个根节点和两棵不相交的分别称为根节点的左子树和右子树的 2 叉树组成。

重温 2 叉树的种类

- 满 2 叉树：除了叶节点外，每个节点都有两个子节点的 2 叉树称为满 2 叉树。
- 完全 2 叉树：所有叶子在同一层的满 2 叉树称为完全 2 叉树。
- 几乎完全的 2 叉树：除了最后一层外，每一层都是满的，且最后一层上的叶子都尽可能地靠左的 2 叉树称为几乎完全的 2 叉树。

2 叉树 (III)



2 叉树的性质

1. 在 2 叉树中，第 j 层的顶点数最多是 2^j。
2. 令 2 叉树 T 的顶点数是 n，高度是 h，则有：$n \leqslant \sum_{j=0}^{h} 2^j = 2^{h+1} - 1$。
3. 任何 n 个顶点的 2 叉树的高度至少是 $\lfloor \log n \rfloor$，最多是 $n - 1$。
4. 有 n 个顶点的几乎完全的或完全 2 叉树的高度是 $\lfloor \log n \rfloor$。

最优 2 叉树

现在我们假设对于一颗 2 叉树 T，我们对其每个叶子节点赋予一个权重 w，则我们称这样的 2 叉树为赋权 2 叉树.

定义 19 **[赋权 2 叉树的权重].**

给定一颗赋权 2 叉树 T，设其叶子节点为 $v_1, \dots, v_k$，相应的权重为 $w_1, \dots w_k$，则 T 的权重定义为：

$$W(T) = \sum_{i=1}^{k} w_i \cdot l(v_i)$$

其中 $l(v_i)$ 表示 v_i 的层数。特别的，在所有的叶子节点为 $v_1, \dots, v_k$、相应的权重为 $w_1, \dots w_k$ 的 2 叉树中，权重最小的 2 叉树被称作最优 2 叉树。

我们下面用一个实际的问题来展示最优 2 叉树的作用，以及如何求出相应的最优 2 叉树。

文件压缩

我们现在来介绍一个问题-文件压缩。

问题 20.

假设现在有一个字符型文件，我们希望将其尽可能的压缩文件，但能很容易的重建文件。我们知道的信息有，文件中一共有 n 个字符，分别为 $\{c_1, \ldots, c_n\}$，每个字符出现的次数为 $f(c_1), f(c_2), \ldots, f(c_n)$。我们的目标是找到一种压缩方式 τ，令该压缩方式下，c_i 转换成的字符长度为 $\tau(c_i)$，使得最后文件的总长度最小，即最小化：$\sum_{i=1}^{n} f(c_i) \cdot \tau(c_i)$ 的值.

定长压缩

一个很自然的方法是定长编码压缩，比如假设一共有 $n = 2^k$ 个不同的编码，则我们可以使用 k 位的 01 串来编码每个字符。比如如果文章有 4 个不同的字符 $\{A, B, C, D\}$，则我们可以使用 $00, 01, 10, 11$ 去表示，

文件压缩

定长编码似乎非常有道理，但我们考虑下面这个情况，两个文件由 4 个字符 $\{A, B, C, D\}$ 组成，但其出现次数分别是:

- $f(A) = 25, f(B) = 25, f(C) = 25, f(D) = 25.$
- $f(A) = 1, f(B) = 1, f(C) = 1, f(D) = 97.$

两种情况此时都会用一个 400 位的 01 串表示。但对于第二种情况，是否可能还存在更简单的编码?

如果我们令:

$$A : 100,\ B : 101,\ C : 11,\ D : 0$$

用这种编码的话，第二个文件只需要 105 位的 01 串就可以表达了。

通过选择合适的变长编码可以减小文件表示的数目!

编码-二义性

变长的编码有一个问题，就是可能会出现二义性。假设某个文件中的 a, b, c 分别用如下编码:

$$a : 10,\ b, 100,\ : 0$$

那么对于字符串 100100100:

- 其想表达的是 bbb?
- 还是 acbb?
- $\ldots$

为了避免歧义，我们引入前缀码的概念，即任何字符的编码都不是其他字符编码的前缀。

定义 21.

如果一个编码满足前缀码的性质，即任何一个字符的编码不会是其他某个字符编码的前缀，则称其为前缀码。

前缀码举例

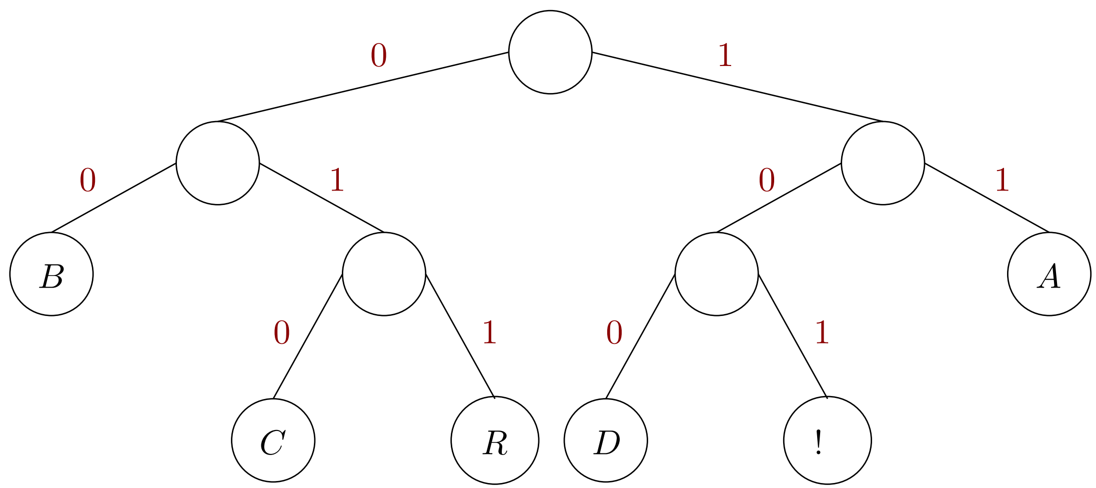
例 22.
考察下面堆字符的一个编码:

$$!:101,\ A:11, B:00,\ C:010,\ D:100,\ R:011$$

其是一个前缀码，对于任何一个由其编码的字符串，其意义是唯一的。

- 11000111101011100110001111101 对应的字符串为 ABRACADABRA!.

前缀码可以由一棵 2 叉树来表示，比如上面的例子对应的 2 叉树为:



Huffman 编码

接下来我们介绍一种构造前缀码的方法，即 Huffman 编码。

David Huffman

Robert Fano

Claude Shannon

- 其直观的思想是，我们希望出现次数多的字符编码尽可能的短。
- 对于前缀码，所有字符都是对应在叶子节点上的，因此算法优先选择出现次数少的字符，将其合并成父节点。

Huffman 算法

Shanghai Normal University 上海师范大学

算法：Huffman

输入: 一个 n 个字符的集合 $C = \{c_1, \ldots, c_n\}$ 和其字符对应出现的频度：$\{f(c_1), \ldots, f(c_n)\}$

输出: C 的一个 Huffman 编码对应的树 (V, T)

```
根据频度将所有字符插入最小堆 H
V ← C, T = ∅
for i = 1 to n − 1 do
    c₁ ← DeleteMin(H)
    c₂ ← DeleteMin(H)
    f(v) ← f(c₁) + f(c₂)          ▷ v 是一个构造出来的 c₁，c₂ 的父节点
    Insert(H, v)
    V ← V ∪ {v}
    T ← T ∪ {(v, c₁), (v, c₂)}
return (V, T)
```

时间复杂性：$O(n \log n)$!

Huffman 编码的效果

上海师范大学 Shanghai Normal University

	a	b	c	d	e	f
频度	45	13	12	16	9	5
定长编码	000	001	010	011	100	101
Huffman 编码	0	101	100	111	1101	1100

假设一共有 100000 个字符。

- 定长编码需要 300000 位。
- Huffman 编码需要 224000 位。

Huffman 编码是压缩率最高的无损编码。

Huffman 算法的正确性 (I)

引理 23.

令 C 是一个字母表。对其中每个字符 $c \in C$，$f(c)$ 为其频率。令 x, y 是其频率最低的两个字符，则存在一个 C 的最优前缀码，使得 x, y 的编码字符长度相同，且只差最后一个二进制不相同。

证明. 令 C' 为其的一个最优前缀码，令 a, b 是其中深度最大的兄弟叶节点，由对称性不妨假设：$f(x) \leqslant f(y)$，$f(a) \leqslant f(b)$，从而有：

$$f(x) \leqslant f(a), \quad f(y) \leqslant f(b)$$

我们将 x 与 a 的位置交换，y 与 b 的位置交换得到的新树 T' 满足：

$$\begin{aligned}
\omega(T') &= \omega(T) + f(a)\cdot d_T(x) + f(b)\cdot d_T(y) + f(x)\cdot d_T(a) + f(y)\cdot d_T(b) \\
&- (f(a)\cdot d_T(a) + f(b)\cdot d_T(b) + f(x)\cdot d_T(x) + f(y)\cdot d_T(y)) \\
&= \omega(T) + (f(a)-f(x))(d_T(a)-d_T(x)) + (f(b)-f(y))(d_T(b)-d_T(y)) \leqslant 0
\end{aligned}$$

从而我们可以调整成使得 x, y 的编码字符长度相同，且只差最后一个二进制不相同的最优前缀码。 □

Huffman 算法的正确性 (II)

引理 24.
令 C 是一个字母表。对其中每个字符 $c \in C$，$f(c)$ 为其频率。令 x, y 是其频率最低的两个字符。令 C' 是字母表 C 去掉 x, y 加入一个新的字符 z 后得到的字母表。C' 也定义了其字符的频率 f'，$f'(c)$ 与 $f(c)$ 相同，除了定义 $f'(z) = f(x) + f(y)$。则对于 C' 的一个最优前缀码对应的编码树 T'，将其中代表 z 的叶子节点替换成一个以 x, y 为孩子的内部节点得到新的树 T，则 T 是 C 的一个最优前缀码对应的编码树。

证明. 我们先来考察 T 和 T' 之间的关系，其代价不难验证存在下述关系：

$$\omega(T') = \omega(T) - f(x) - f(y)$$

Huffman 算法的正确性 (III)

上海师范大学 Shanghai Normal University

证明续. 反设 T 不是 C 的最优前缀码，即存在 T'' 使得 $\omega(T'') < \omega(T)$。则由上述引理，T'' 包含一对兄弟节点 x, y，我们将其和父节点替换成叶节点 z，并令 $f'(z) = f(x) + f(y)$，则在新构成的树 T''' 有：

$$\omega(T''') = \omega(T'') - f(x) - f(y) < \omega(T) - f(x) - f(y) = \omega(T')$$

与假设矛盾。 □

由上述两个引理不难得出：

定理 25.

Huffman 算法可以正确的构造出最优前缀码。

本章总结

本章总结

- 树的基本概念。
  - 树的等价定义。
  - 生成树的概念。
  - Cayley 公式。
- 最小生成树的计算。
  - Kruskal 算法。
  - Prim 算法。
- Huffman 编码。
  - 根树、最优 2 叉树的概念。
  - Huffman 算法。