**回路矩阵**

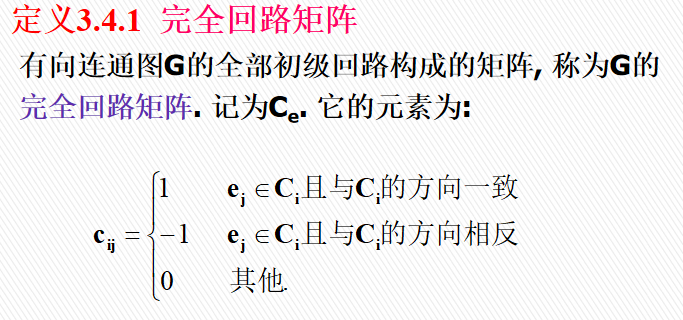
4月18日13:59

若通路(回路)中所有边各异, 则称为简单通路(简单回路), 否则称为复杂通路(复杂回路)。

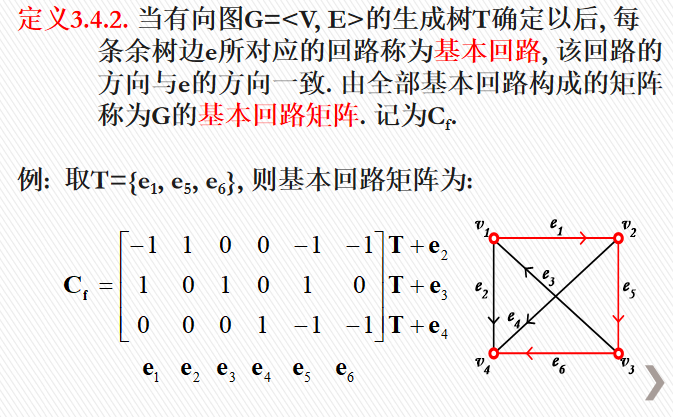
若简单通路中所有顶点各异, 则称为初级通路或路径。

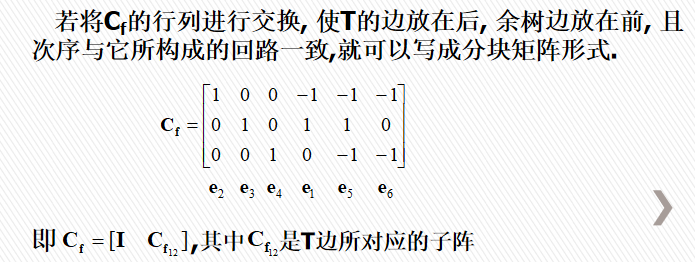
若简单回路中除起点终点外，所有顶点各异，则称为初级回路或圈。

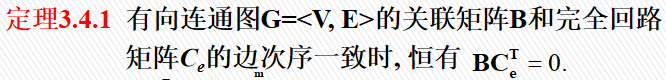
边不重复为简单，点不重复为初级。先满足简单才可能能称为初级。



初级回路：除起点和终点外，边不重复，点不重复。







首先，这个定理其实是对任意回路矩阵而言的。关联矩阵与完全回路矩阵的列是边，回路矩阵取转置之后就满足了矩阵可乘。

回路矩阵与关联矩阵只在意边（列）的排列，关联矩阵点的排布和回路矩阵具体回路的排布没有影响。

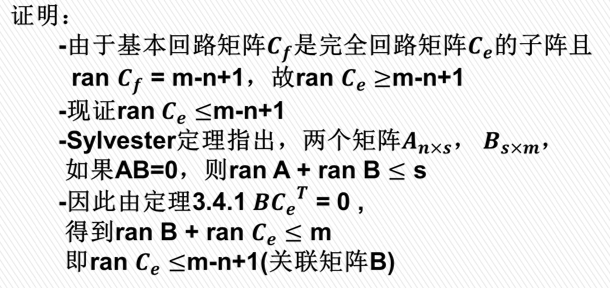
实际上，这个定理可以推广到：基本关联矩阵和任意回路矩阵上。

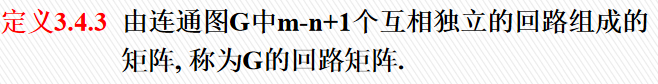
因为基本关联矩阵和回路矩阵分别是关联矩阵和完全回路矩阵的分块，整体为0，分块之积必为0；



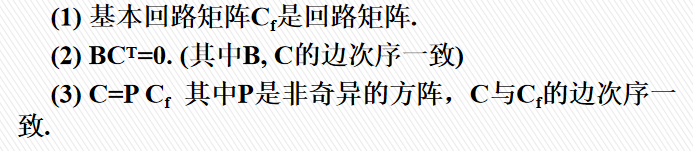
完全回路矩阵的Rank是m-n+1；这里之后会和其他几个矩阵的rank一起作对比；

其实证明也不复杂，顺便看看：



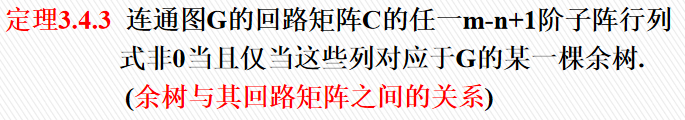


回路矩阵就是在完全回路矩阵里抽出线性无关的m-n+1行即可；



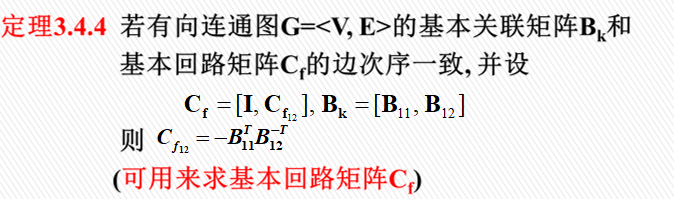
其实回路矩阵不过是对应着选择不同的树构建出的不同的基本回路矩阵罢了；

第三点的理解：因为C\_f是行满秩的，它的行空间和C就一样，所以C中任一行都可以被C\_f的各行线性表示。



对于每个回路矩阵，只有一个m-n+1阶的子阵det不为0，并不！

删去树上一条边，塞入一条余树边，其实完全有可能构成新的树。



B12是先取了逆再取负；注意到，用来求出了Cf12之后还要再把I加会去；

注意到，基本回路矩阵与基本割集矩阵都是是余树边在前。B11是n-1阶方阵。行对应n-1个点（其中被任意删除了一个），列对应着余树边们；树有n-1条边，余树有m-n+1条边；

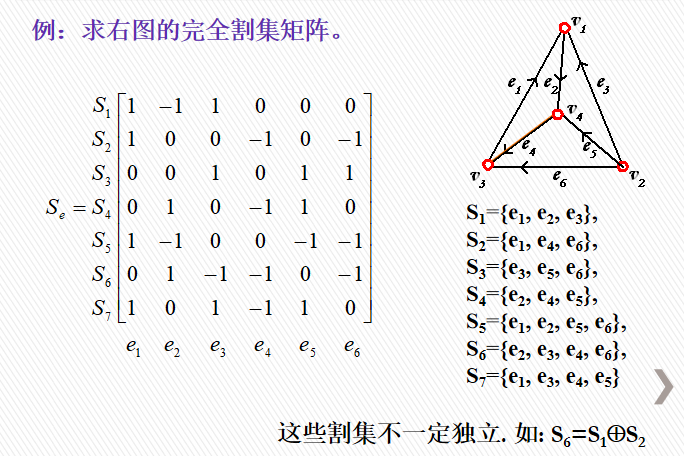
**割集矩阵**

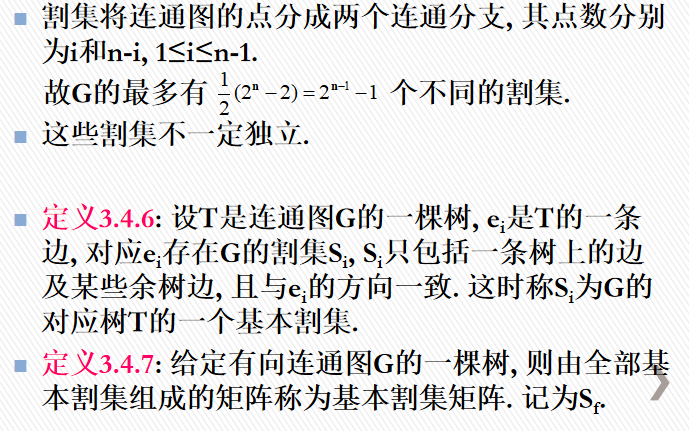
4月18日14:35



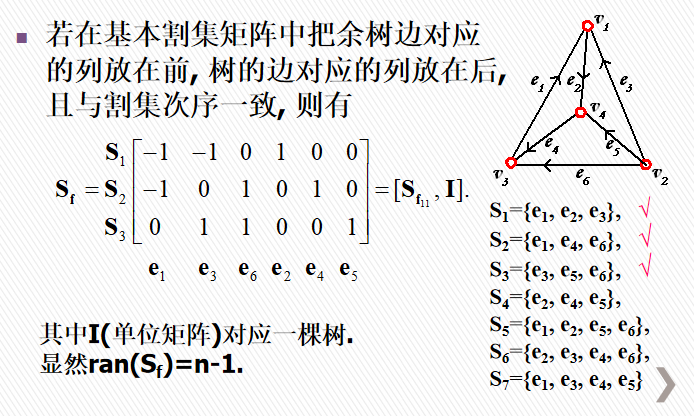
首先，割集是个集合，可以视为把一些点完全孤立所需要去掉的边；

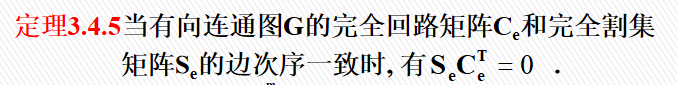
1. 割集的方向：人为设定，比如我们创立一个把v1独立的割集，设定指向v1为割集的正方向，那么e1就是+1，e2就是-1；或者说，割边连接两个连通支，就只有两个方向，从A到B或者从B到A，指定从A到B的为+，那从B到A就是-；



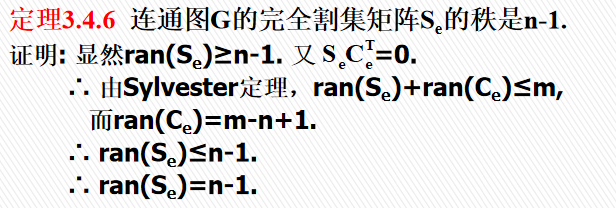


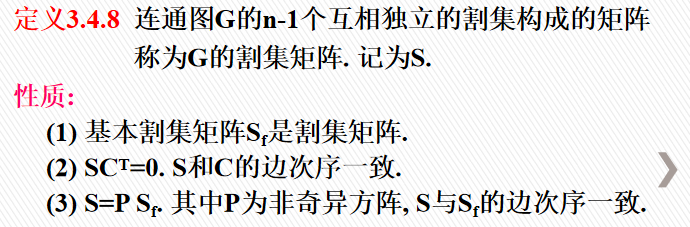
基本割集：仅仅含有一条树上的边和若关余树边，且设定树上的边的方向为割集的正方向；

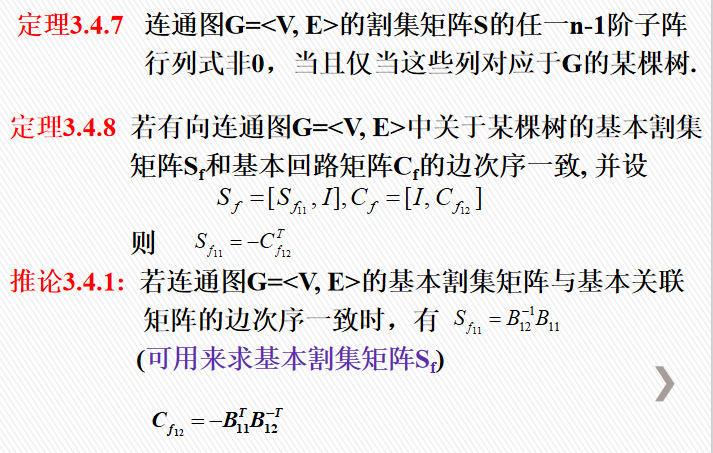




同样的，这个定理可以扩展到任意回路矩阵和基本割集矩阵；







**最小支撑树**

4月18日15:05

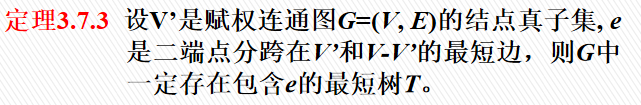


Prim算法：

构造三个集合A、B、C，B当中初始存有所有的顶点，且所有顶点初始化权值为+infinite，A初始化为空集用于以后存点，C用于存储边，初始化为空集；

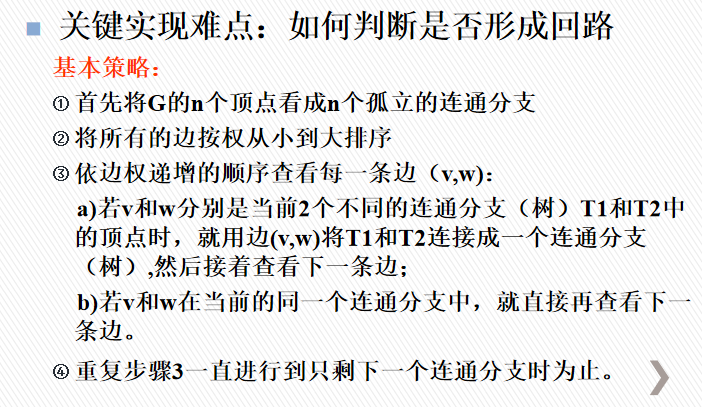
随意选择B中一个节点为起始节点加入A，不妨视为V1，用V1更新所有与V1相连的且在B中的节点的权值，并记录父节点为v1。

选择权值最小的点（这个选择方式和Dijkstra相同）（不妨为V2），加入A，并把（V1,V2）加入C。之后用V2到其连接点的长度与原来的权作比较。（这里和Dijkstra的更新原则不太一样，前者是b的权加上bk的长，Prim直接是bk的长）如果某点的权减小，那么把权重记为V2到该点的长度，并将该点的父节点为V2，再选择权值最小的点和对应的边加入A和C，依次更新；

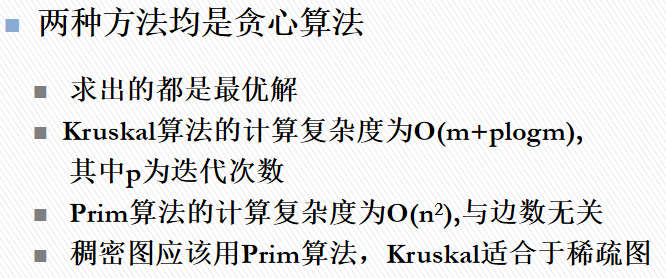


Kruskal算法（并查集算法）：

不断选择全局上的最短边，如果这条边与之前选择的边构成了回路就删去，直到所有边都被选择了。（因为其他边被删完了）

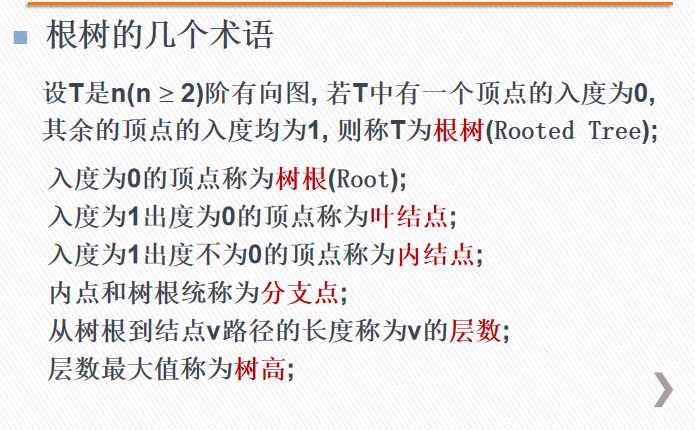


翻译一下，如果这条长度最短的边的起点和终点都在我们构造的连通支里，那就删去。反之则添加，并且扩大连通支。



**Huffman树**

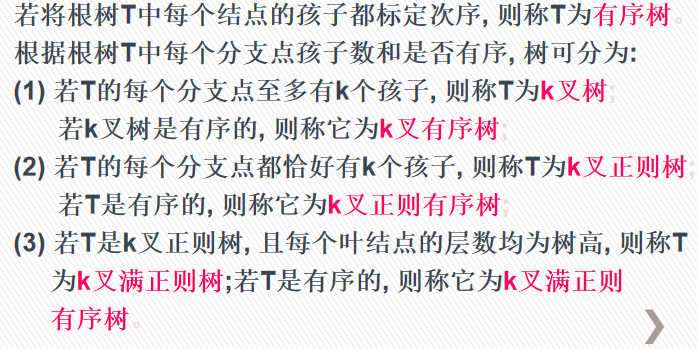
4月18日16：21



注意当我的树的边权不为1时，层数并不是简单的边的条数；



注意到-->符号表示间接连接；

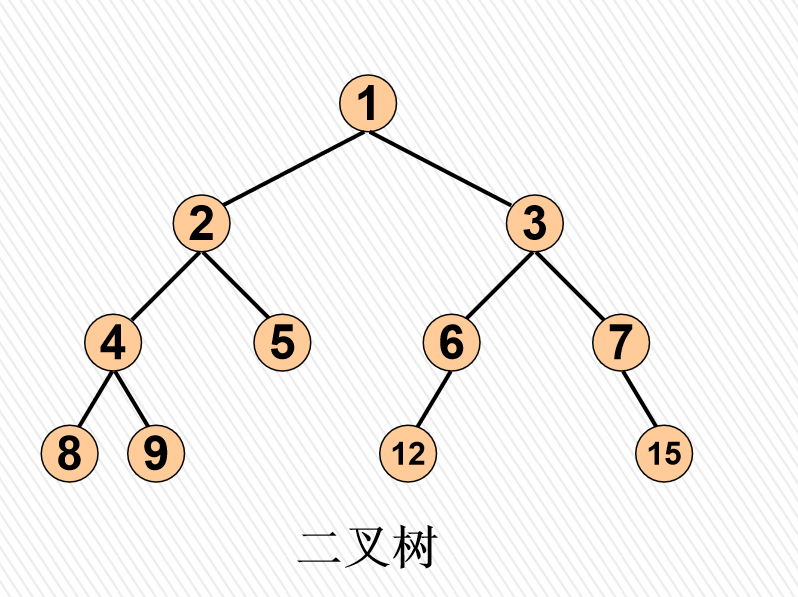


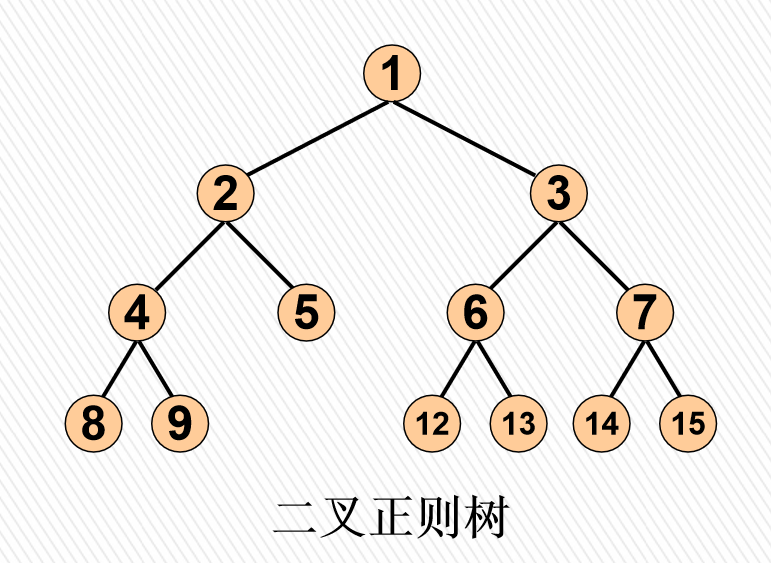
满：每个叶节点的层数=树高；

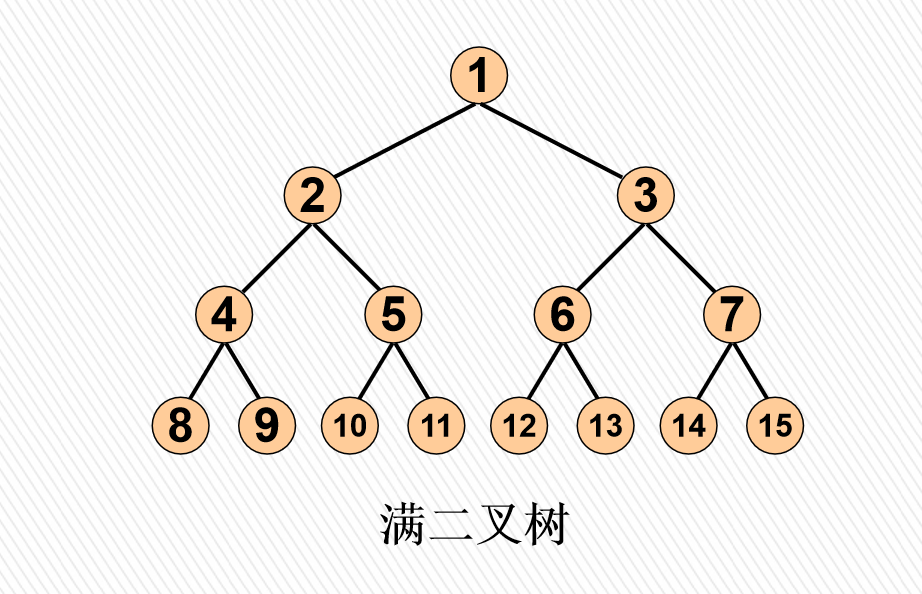
正则：每个分支点恰好k个孩子；（注意到，叶节点不是分支点，所以二叉树不一定是满的）

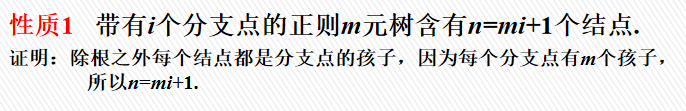
k叉：最多k个孩子；

有序：孩子都有序；

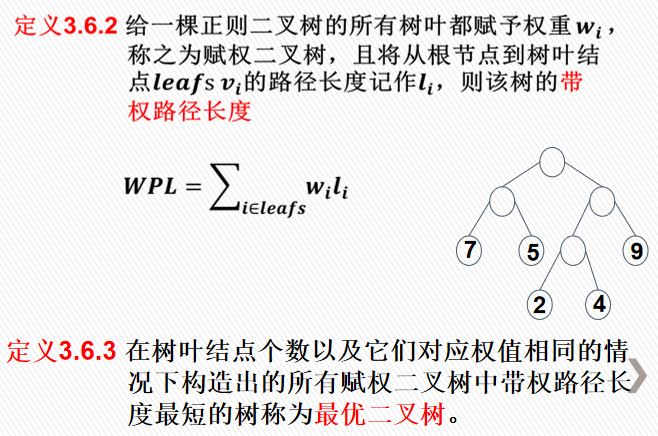




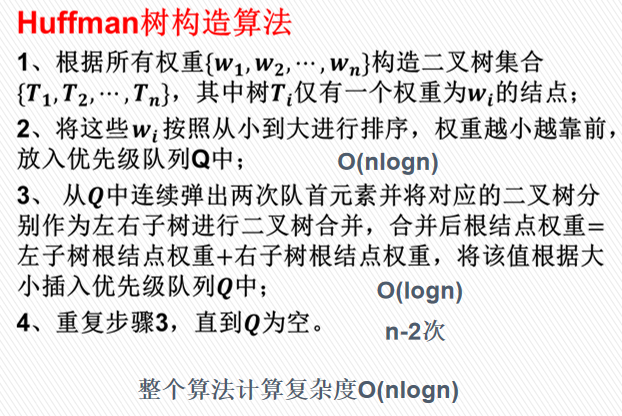




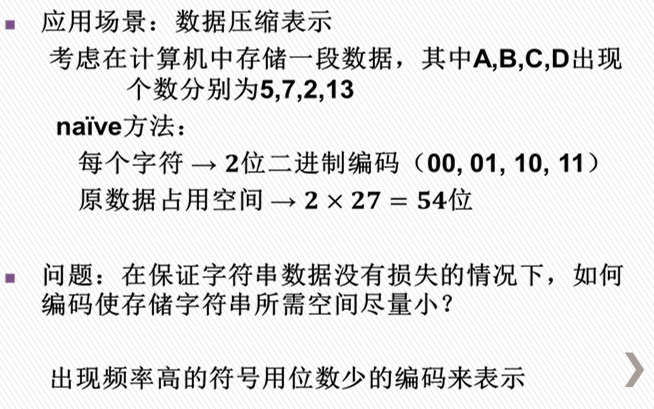


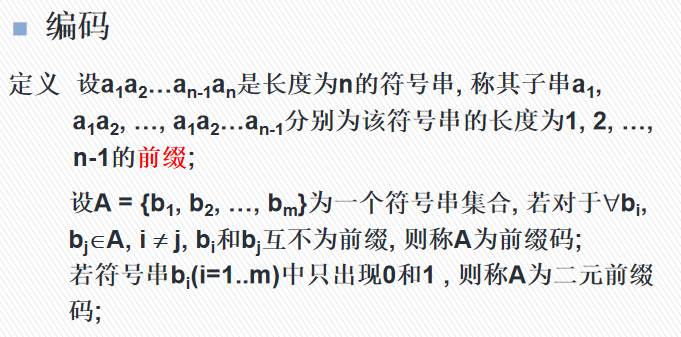


权重赋值只在叶节点上；WPL=所有叶子的权\*叶子的长度之和；

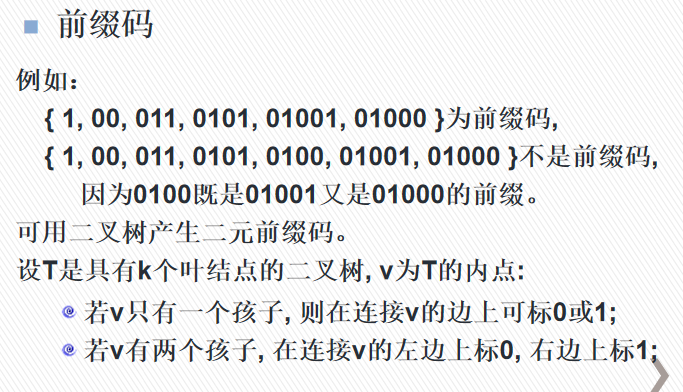


弹出两个最小的叶子，然后合并为新的叶子放回去；

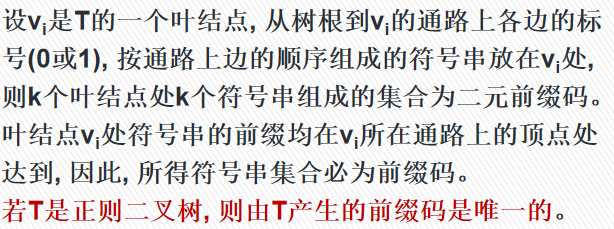




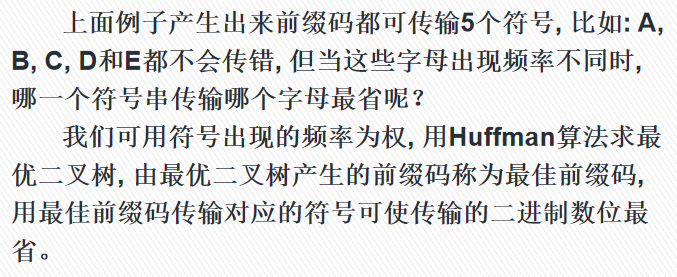
前缀码的定义：字符串集合中的元素不互为前缀；



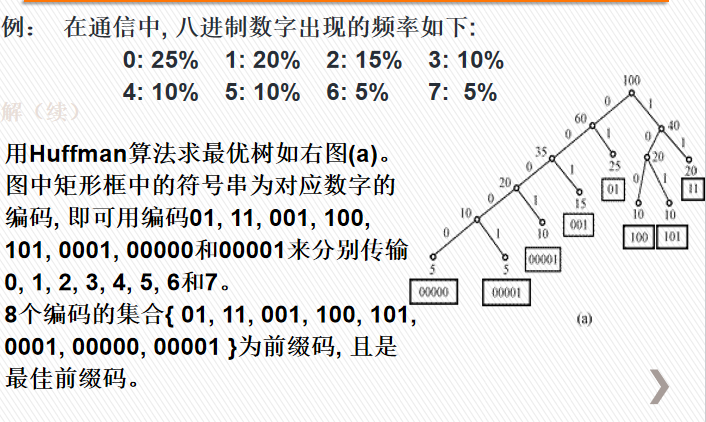
这个其实可以用树的尾部来联想下；



不正则，尾部叶片只有一个，它路上可以写1也可以0，故而不是前缀码不唯一。当然，一棵树正则的部分是基本唯一的，因为我们的编码需要符合：有两个儿子时，左0右1。



也就是先用Huffman算法构造出最优二叉树，接着把这棵树赋值为前缀码；





这里的意思是我的那棵按照概率传递信息，用编码的位数（也就是叶子的长度）\*权重的求和。