20220613-数据结构

1.学习内容

1.1 数据结构

二叉编码树

1.学习内容

1.1 数据结构

二叉编码树

个字符串的编码串互不为前缀，即便出现无法解码的错误，也绝对不致歧义，这类编码方案即所谓的“前缀无歧义编码”（prefix-free code），简称PFC编码。

表5.1 Σ = { 'A', 'E', 'G', 'M', 'S' }的一份二进制编码表

字符	A	E	G	M	S
编码	00	01	10	110	111

表5.3 Σ = { 'A', 'E', 'G', 'M', 'S' }的另一份编码表

字　符	A	E	G	M	S
编码	00	01	10	11	111

任一编码方法都可描述为一棵二叉树：从根节点出发，每次向左或向右对应于一个0或1比特位。由根节点到每个节点的唯一通路，可以为各节点v赋予一个互异的二进制串，称作根通路串，记作rps(v)。|rps(v)|=depth(v)就是v的深度。

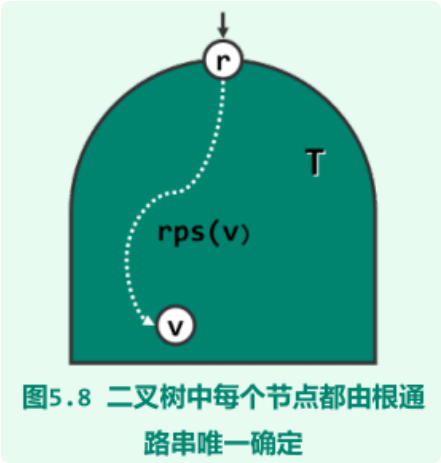

图5.8 二叉树中每个节点都由根通路串唯一确定

要实现PFC编码，只需所有字符对应于叶节点。而在解码时，从前向后扫描该串，同时在树中相应移动：起始时从树根出发，视各比特位的取值相应地向左或向右深入下一层，直到抵达叶节点。这一解码过程可以在编码串接收的过程中实时进行，不必等到所有比特位都到达，因此这类算法属于在线算法。

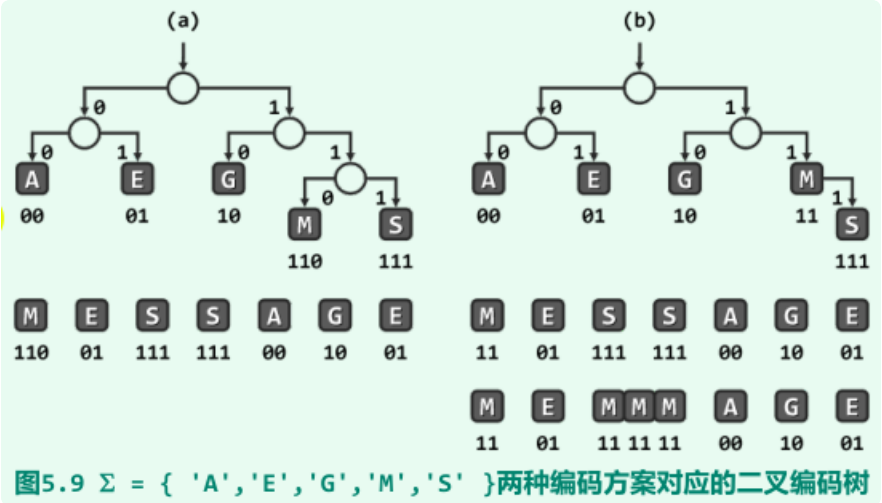

图5.9 Σ = { 'A','E','G','M','S' }两种编码方案对应的二叉编码树

二叉树的实现：

C++ 复制代码

```cpp
#define BinNodePosi(T) BinNode<T>* //节点位置
#define stature(p)((p)?(p)->height:-1)//节点高度
typedef enum {RB_RED,RB_BLACK} RBColor;//节点颜色

template <typename T> struct BinNode
{
    T data;
    BinNodePosi(T) parent;//父节点
    BinNodePosi(T) lc;//左孩子
    BinNodePosi(T) rc;//右孩子
    int height;
    int npl;
    RBColor color;
    //构造函数
    BinNode():
        parent(NULL),lc(NULL),rc(NULL),height(0),npl(1),color(RB_RED){}
    BinNode(T e,BinNodePosi(T) p=NULL,BinNodePosi(T)
lc=NULL,BinNodePosi(T) rc=NULL,int h=0,
        int l=1,RBColor c=RB_RED):
        data(e),parent(p),lc(lc),rc(rc),height(h),npl(l),color(c){}
    //操作接口
    int size();//统计当前节点后代总数
    BinNodePosi(T) insertAsLC(T const&);
    BinNodePosi(T) insertAsRC(T const&);
    BinNodePosi(T) succ();
    template<typename VST> void travLevel(VST&);
    template<typename VST> void travPre(VST&);
    template<typename VST> void travIn(VST&);
    template<typename VST> void travPost(VST&);
    bool operator<(BinNode const& bn) { return data < bn.data; }
    bool operator==(BinNode const& bn) { return data == bn.data; }
};
```