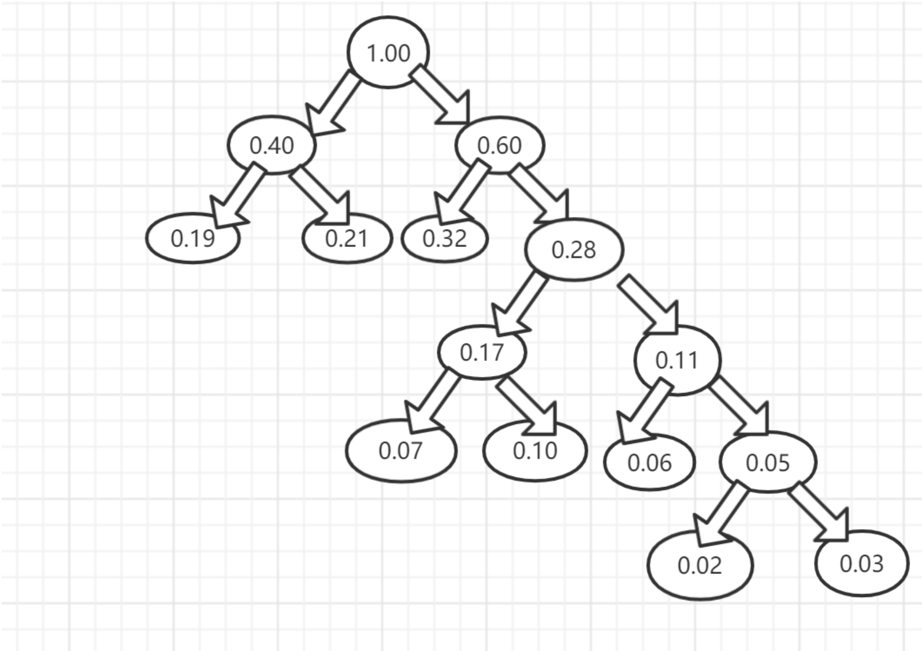
应用题

3.（1）（2）



对应的哈夫曼编码及等长编码如图所示：

字母编号	权重	哈夫曼编码	等长编码
1	0.07	1100	000
2	0.19	00	001
3	0.02	11110	010
4	0.06	1110	011
5	0.32	10	100
6	0.03	11111	101
7	0.21	01	110
8	0.10	1101	111

(3)对于上述实例，等长编码的构造更简单；但哈夫曼编码保证概率大的符号对应于短码，概率小的符号对应于长码而且所有的短码得到充分利用，实现了加权的最优。

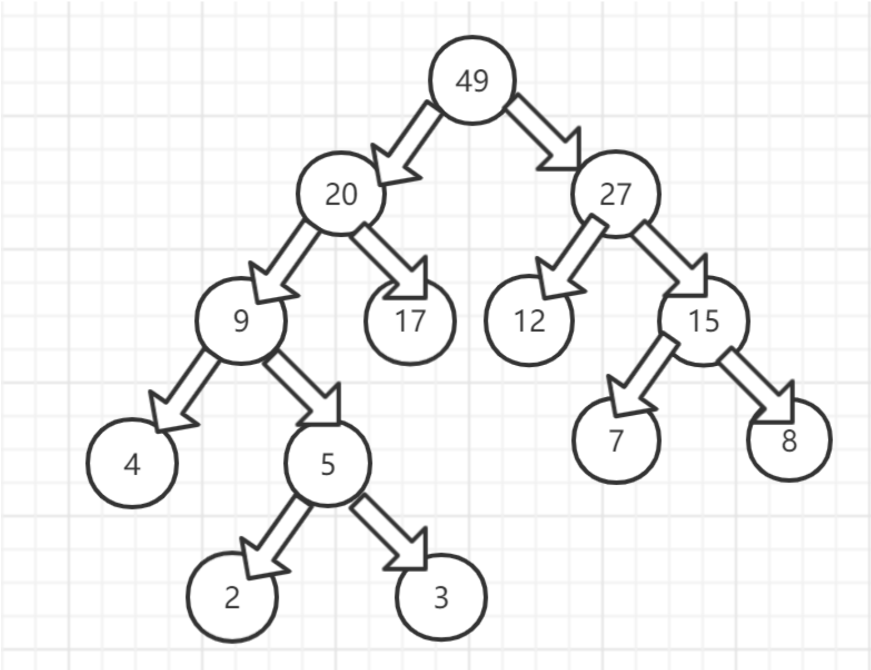
计算哈夫曼编码生成的树的带权路径长度：

2*(0.19+0.32+0.21)+4*(0.07+0.06+0.10)+5*(0.02+0.03)=2.61

计算等长编码生成的树的带权路径长度：

3*(0.19+0.32+0.21+0.07+0.06+0.10+0.02+0.03)=3

4.哈夫曼树如图



HT 树存储结构初态:

节点i	weight	parent	lchild	rchild
1	3	0	0	0
2	12	0	0	0
3	7	0	0	0
4	4	0	0	0
5	2	0	0	0
6	8	0	0	0
7	11	0	0	0
8	-	0	0	0
9	-	0	0	0
10	-	0	0	0
11	-	0	0	0
12	-	0	0	0
13	-	0	0	0
	-	0	0	0
15	-	0	0	0

HT 树存储结构终态:

节点i	weight	parent	lchild	rchild
1	3	8	0	0
2	12	12	0	0
3	7	10	0	0
4	4	9	0	0
5	2	8	0	0
6	8	10	0	0
7	11	11	0	0
8	5	9	5	1
9	9	11	4	8
10	15	12	3	6
11	20	13	9	7
12	27	13	2	10
13	47	0	11	12

算法设计题

（3）

```
void change(BiTree& T)
{
	if (T == NULL)
		return;
	else if (T->lchild == NULL && T->rchild == NULL)
//如果 T 为空或者 T 的左、右子树均为空，则返回
		return;
	else()
	{
		temp = T->lchild;
		T->lchild = T->rchild;
		T->rchild = temp;
	}
	change(T->lchild);//递归处理左子树及右子树
	change(T->rchild);
}
```

（4）

```
Int width(BiTree T)
{
	if (T == NULL)
		return 0;
	else
	{
		BiTreeQ[];
		front = 1;
		rear = 1;
		last = 1;
		temp = 0;
		maxw = 0;
```

```
        Q[rear] = T;
        while (front <= last)
        {
            p = Q[front++];
            temp++;
            if (p->lchild != NULL)
                Q[++rear] = p->lchild;
            if (p->rchild != NULL)
                Q[++rear] = p->rchild;
            if (front > last)
            {
                last = rear;
                if (temp > maxw)
                    maxw = temp;
                temp = 0;
            }
        }
        return maxw;
    }
}
```

（7）