递归算法设计示例

3.3.4 递归算法设计举例

递归的求解的过程均有这样的特征：

先将整个问题划分为若干个子问题，通过分别求解子问题，最后获得整个问题的解。

而这些子问题具有与原问题相同的求解方法，于是可以再将它们划分成若干个子问题，分别求解，如此反复进行，直到不能再划分成子问题，或已经可以求解为止。这种自上而下将问题分解、求解,再自下而上引用、合并，求出最后解答的过程称为递归求解过程。这是一种分而治之的算法设计方法。

递归算法设计先要给出递归模型，再转换成对应的C/C++语言函数。

求递归模型的步骤如下：

- （1）对原问题进行分析，假设出合理的“较小问题”；
- （2）假设“较小问题”是可解的,在此基础上确定“原问题”的解；
- （3）确定一个特定情况的解,由此作为递归出口。

【示例-1】采用递归算法求实数数组A[0..n-1]中的最小值。

假设f(A,i)函数求数组元素A[0]～A[i]中的最小值。

当i=0时，有f(A,i)=A[0]；

假设f(A,i-1)已求出，则f(A,i)=MIN(f(A,i-1),A[i])，其中MIN()为求两个值较小值函数。

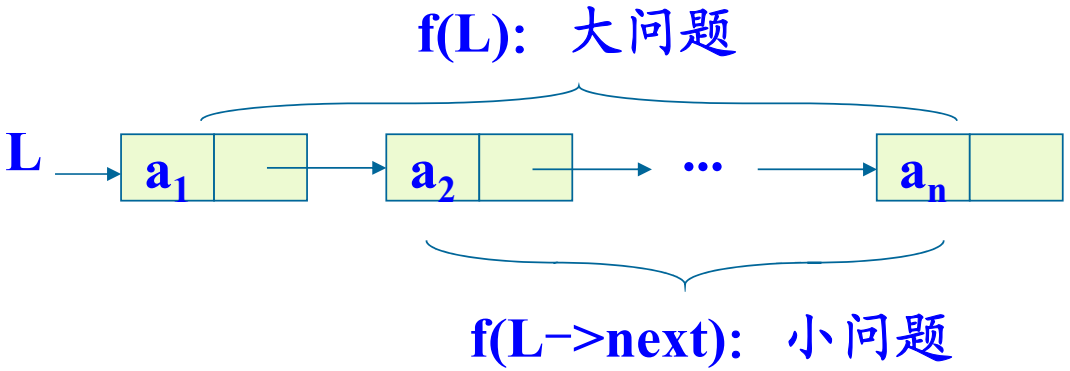
因此得到如下递归模型：

f(A,i)=A[0]	当i=0时
f(A,i)= MIN(f(A,i-1),A[i])	其他情况

由此得到如下递归求解算法：

```
float f(float A[], int i)
{  float m;
   if (i==0)
      return A[0];
   else
   {   m=f(A,i-1);
       if (m>A[i])
          return A[i];
       else
          return m;
   }
}
```

【示例-2】设计不带头结点的单链表的相关递归算法



（1）求单链表中数据结点个数

递归模型如下：

f(L)=0	当L=NULL
f(L)=f(L->next)+1	其他情况

递归算法如下：

```
int count(LNode *L)
{
    if (L==NULL)
        return 0;
    else
        return count(L->next)+1;
}
```

（2）正向显示以L为首结点指针的单链表的所有结点值

递归模型如下：

f(L)↔不做任何事件　　当L=NULL

f(L)↔输出L->data;f(L->next)　　其他情况

递归算法如下：

```
void traverse(LNode *L)
{
   if (L==NULL)
      return;
   printf("%d ",L->data);
   traverse(L->next);
}
```

（3）反向显示以L为首结点指针的单链表的所有结点值

递归模型如下：

f(L)↔不做任何事件　　当L=NULL

f(L)↔f(L->next);输出L->data;　　其他情况

递归算法如下：

```
void traverseR(LNode *L)
{
   if (L==NULL)
      return;
   traverseR(L->next);
   printf("%d ",L->data);
}
```

（4）删除以L为首结点指针的单链表中值为x的第一个结点

递归模型如下：

f(L,x)↔不做任何事件	当L=NULL
f(L,x) ↔ 删除L所指结点	当L->data=x
f(L,x)↔f(L->next,x)	其他情况

递归算法如下：

```
void del(LNode *&L, ElemType x)
{  LNode *t;
   if (L==NULL) return;
   if (L->data==x)
   {   t=L;
       L=L->next;
       free(t);
       return;
   }
   else del(L->next,x);
}
```

（5）删除以L为首结点指针的单链表中值为x的所有结点

递归模型如下：

f(L,x)↔不做任何事件　　当L=NULL
f(L,x) ↔ 删除L所指结点;f(L->next,x)　　当L->data=x
f(L,x) ↔f(L->next,x)　　其他情况

递归算法如下：

```
void delall(LNode *&L,ElemType x)
{  LNode *t;
   if (L==NULL) return;
   if (L->data==x)
   {   t=L;
       L=L->next;
       free(t);
       delall(L,x);
   }
   else delall(L->next,x);
}
```

（6）输出以L为首结点指针的单链表中最大结点值

递归模型如下：

f(L)=L->data　　当L中只有一个结点

f(L)=MAX(f(L->next),L->data)　　其他情况

递归算法如下：

```
ElemType maxv(LNode *L)
{
   ElemType m;
   if (L->next==NULL)
       return L->data;
   m=maxv(L->next);
   if (m>L->data)
       return m;
   else
       return L->data;
}
```

（7）输出以L为首结点指针的单链表中最小结点值。

递归模型如下：

f(L)=L->data　　当L中只有一个结点

f(L)=MIN(f(L->next),L->data)　　其他情况

递归算法如下：

```
ElemType minv(LNode *L)
{
    ElemType m;
    if (L->next==NULL)
        return L->data;
    m=minv(L->next);
    if (m>L->data) return L->data;
    else return m;
}
```

（8）释放L为首结点指针的单链表的所有结点

递归模型如下：

f(L)↔不做任何事件　　当L=NULL

f(L)↔f(L->next);释放L所指结点；　其他情况

递归算法如下：

```
void Destroy(LNode *L)
{
   if (L==NULL)
      return;
   Destroy(L->next);
   free(L);
}
```

END