

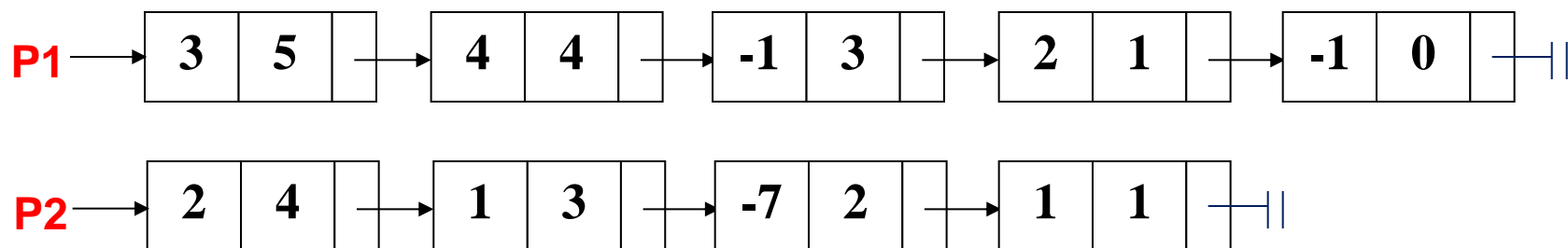
2.4 应用: 多项式加法运算

$$\begin{array}{r} P1 = 3X^5 + 4X^4 - X^3 + 2X - 1 \\ + P2 = 2X^4 + X^3 - 7X^2 + X \\ \hline P = 3X^5 + 6X^4 - 7X^2 + 3X - 1 \end{array}$$

主要思路：相同指数的项系数相加，其余部分进行拷贝。

多项式加法运算

采用不带头结点的单向链表，按照指数递减的顺序排列各项



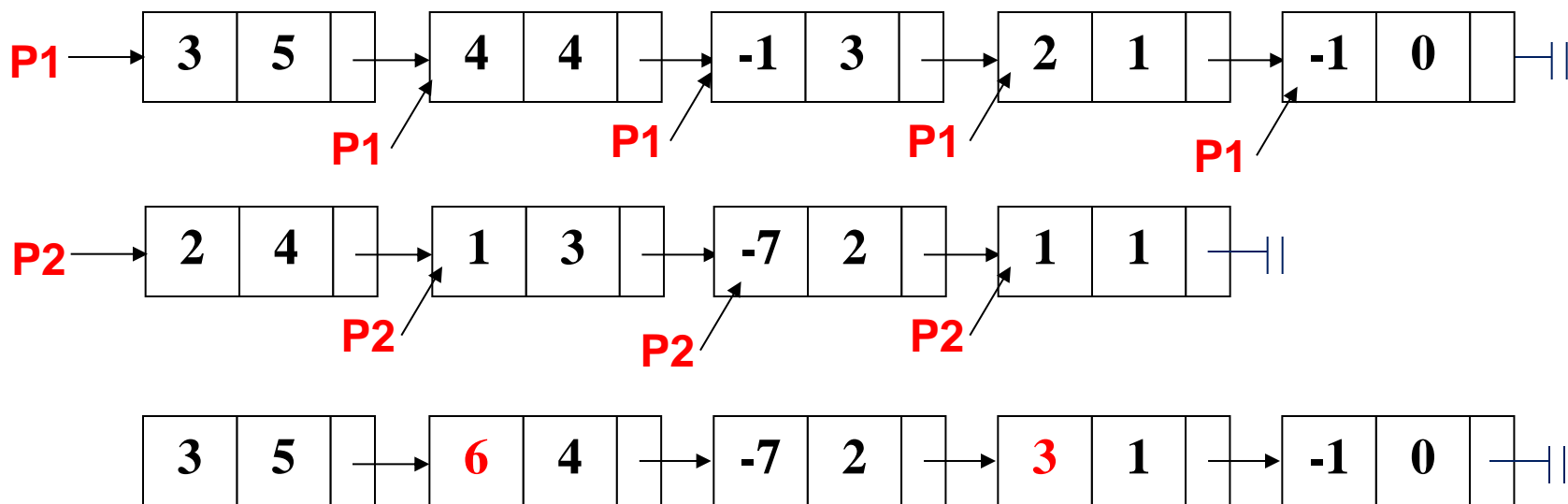
```
struct PolyNode {  
    int coef;    // 系数  
    int expon;   // 指数  
    struct PolyNode *link; // 指向下一个节点的指针  
};  
typedef struct PolyNode *Polynomial;  
Polynomial P1, P2;
```

多项式加法运算

算法思路：两个指针P1和P2分别指向这两个多项式第一个结点，不断循环：

- **P1->expon==P2->expon:** 系数相加，若结果不为0，则作为结果多项式对应项的系数。同时，P1和P2都分别指向下一项；
- **P1->expon>P2->expon:** 将P1的当前项存入结果多项式，并使P1指向下一项；
- **P1->expon<P2->expon:** 将P2的当前项存入结果多项式，并使P2指向下一项；

当某一多项式处理完时，将另一个多项式的所有结点依次复制到结果多项式中去。



Polynomial PolyAdd (Polynomial P1, Polynomial P2)

```
{  
    Polynomial front, rear, temp;  
    int sum;  
    rear = (Polynomial) malloc(sizeof(struct PolyNode));  
    front = rear; /* 由front 记录结果多项式链表头结点 */
```

为方便表头插入，先产生一个临时空结点作为结果多项式链表头

```
while ( P1 && P2 ) /* 当两个多项式都有非零项待处理时 */
```

```
switch ( Compare(P1->expon, P2->expon) ) {
```

```
case 1:
```

```
    Attach( P1->coef, P1->expon, &rear);
```

```
    P1 = P1->link;
```

```
    break;
```

```
case -1:
```

```
    Attach(P2->coef, P2->expon, &rear);
```

```
    P2 = P2->link;
```

```
    break;
```

```
case 0:
```

```
    sum = P1->coef + P2->coef;
```

```
    if ( sum ) Attach(sum, P1->expon, &rear);
```

```
    P1 = P1->link;
```

```
    P2 = P2->link;
```

```
    break;
```

```
}
```

P1中的数据项指数较大

P2中的数据项指数较大

两数据项指数相等

注意判断系数和是否为0

```
/* 将未处理完的另一个多项式的所有节点依次复制到结果多项式中去 */
```

```
for ( ; P1; P1 = P1->link ) Attach(P1->coef, P1->expon, &rear);
```

```
for ( ; P2; P2 = P2->link ) Attach(P2->coef, P2->expon, &rear);
```

```
rear->link = NULL;
```

```
temp = front;
```

```
front = front->link; /* 令front指向结果多项式第一个非零项 */
```

```
free(temp); /* 释放临时空表头结点 */
```

```
return front;
```

```
void Attach( int c, int e, Polynomial *pRear )
```

```
{    /* 由于在本函数中需要改变当前结果表达式尾项指针的值, */  
    /* 所以函数传递进来的是结点指针的地址, *pRear指向尾项*/  
    Polynomial P;
```

```
    P =(Polynomial)malloc(sizeof(struct PolyNode)); /* 申请新结点 */
```

```
    P->coef = c;                                /* 对新结点赋值 */
```

```
    P->expon = e;
```

```
    P->link=NULL;
```

```
    /* 将P指向的新结点插入到当前结果表达式尾项的后面 */
```

```
    (*pRear)->link = P;
```

```
    *pRear = P;          /* 修改pRear值 */
```

```
}
```

