자료구조 Data Structure | 조행래

연결 리스트

연결 리스트를 이용한 다항식의 구현

학습 목표

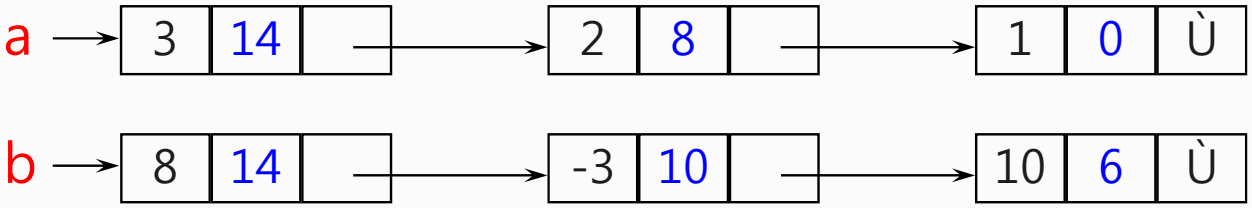
- 연결 리스트를 이용하여 다항식을 표현할 수 있다.
- 연결 리스트로 표현한 다항식에 대해 덧셈 연산을 구현할 수 있다.

1. 다항식의 연결 리스트 표현

- **다항식 구조체**

```
struct poly {
        int coef;           // 계수
        int expon;          // 지수
        struct poly *link;  // 링크
} *a, *b, *d;
```

$a = 3x^{14} + 2x^{8} + 1$ $\qquad$ $b = 8x^{14} - 3x^{10} + 10x^{6}$

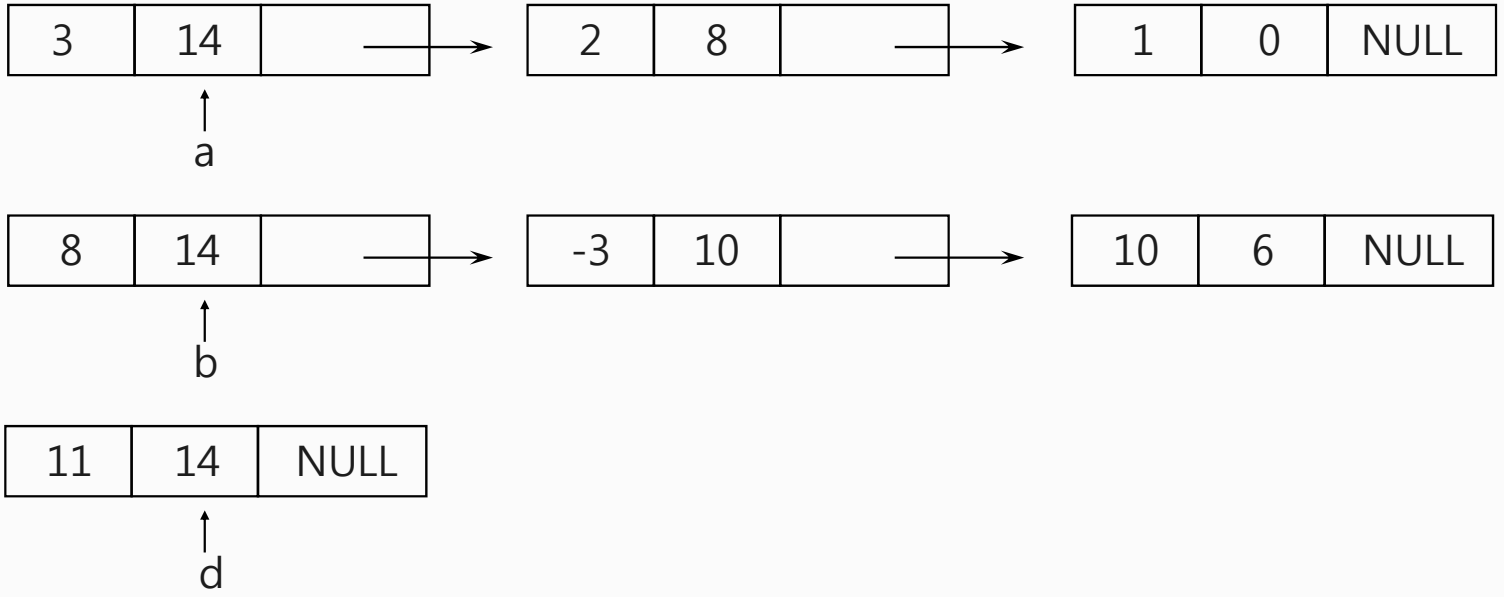


지수의 내림차순으로 연결

2. 다항식의 덧셈 알고리즘

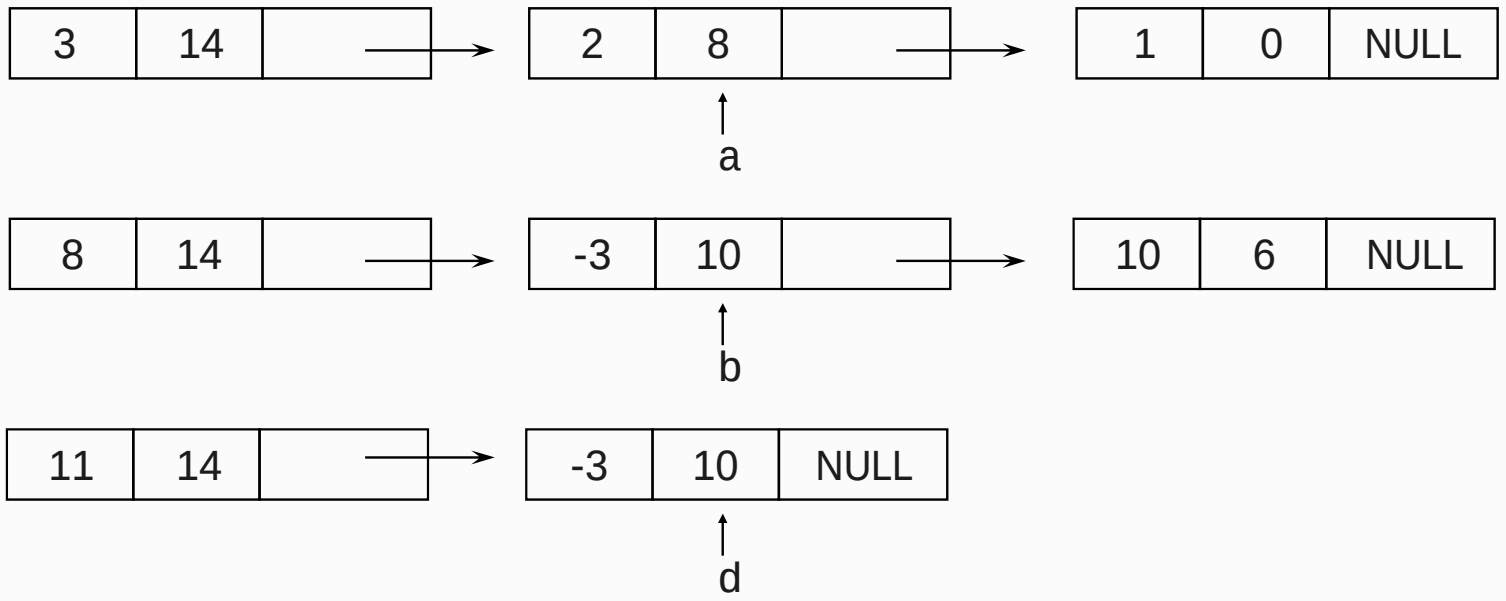
- **d = a + b**
  - 각 다항식의 최고차 항부터 차례대로 비교

- **최고차 항의 지수가 동일할 경우**



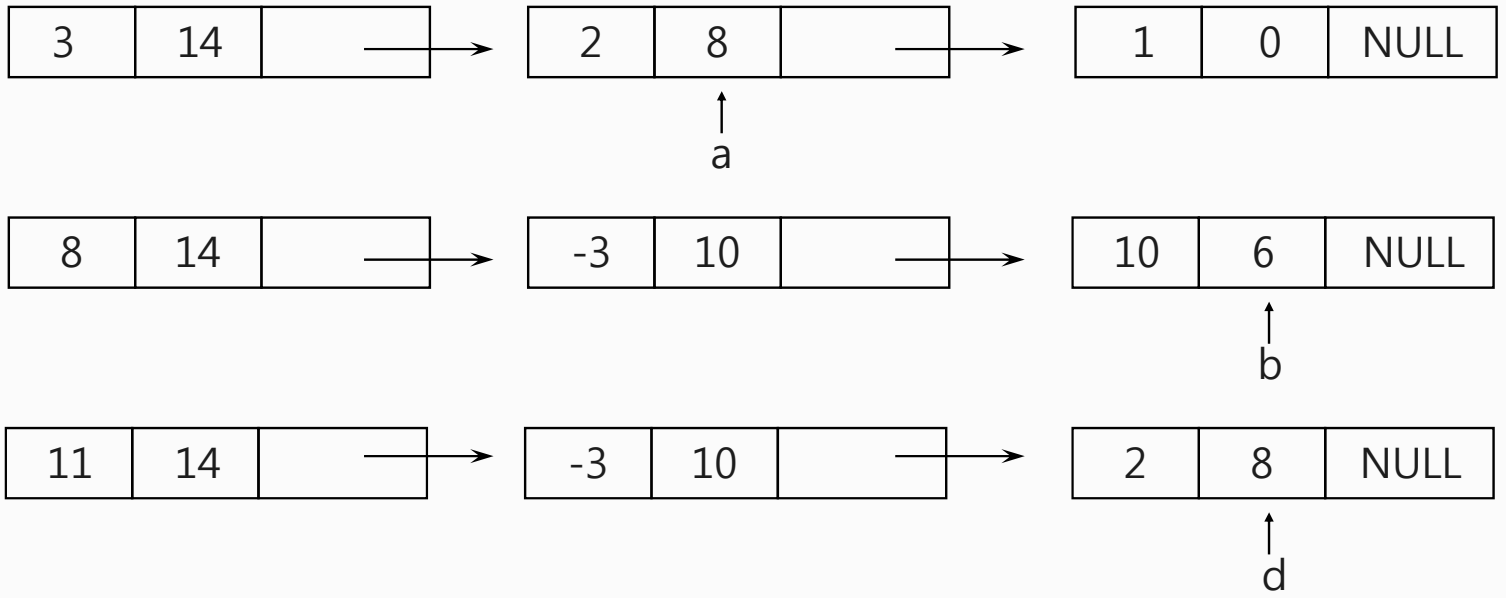
다항식의 덧셈 알고리즘 (계속)

- a→expon < b→expon 인 경우



다항식의 덧셈 알고리즘 (계속)

- a→expon > b→expon 인 경우



다항식 d의 항 추가는 리스트의 끝에서 발생 → 큐 방식

3. 덧셈 알고리즘의 구현

```
struct poly *padd(struct poly *a, struct poly *b)
{
    // d = a + b인 다항식 d를 return
    struct poly *front, *rear, *tmp;
    int sum;
    // 사용하지 않는 처음 노드를 생성. 이유는?
    rear = (struct poly *)malloc(sizeof(struct poly));
    if (rear == NULL) {
        fprintf(stderr, "The memory is full₩n");
        exit(1);
    }
    front = rear;
    while (a && b)
        switch (COMPARE(a→expon, b→expon)) {
          case -1: // a→expon < b→expon
                attach(b→coef, b→expon, &rear);
                b = b→link;      break;
```

덧셈 알고리즘의 구현 (계속)

```
            case 0: // a→expon == b→expon
                    sum = a→coef + b→coef;
                    if (sum) attach(sum, a→expon, &rear);
                    a = a→link;  b = b→link;  break;
            case 1: // a→expon > b→expon
                    attach(a→coef, a→expon, &rear);
                    a = a→link;
        }
    // 리스트 a와 b의 나머지 부분을 복사
  for (; a; a = a→link) attach(a→coef, a→expon, &rear);
  for (; b; b = b→link) attach(b→coef, b→expon, &rear);
  rear→link = NULL;
  // 처음 노드를 삭제
  tmp = front;  front = front→link;  free(tmp);
  return front;
}
```

덧셈 알고리즘의 구현 (계속)

```
void attach ( float coefficient,  int exponent,  struct poly **rear)
{
/*  coef = coefficient이고 expon = exponent인 새로운 노드를 생성한
    후, rear 다음에 연결하고 rear 변경 */

    struct poly *tmp;
    tmp = (struct poly *) malloc(sizeof(struct poly));
    if (tmp == NULL)  {
          fprintf ( stderr, " The memory is full ₩n " );
          exit(1);
    }
    tmp→coef  =  coefficient;
    tmp→expon  =  exponent;
    (*rear)→link  =  tmp;     // rear가 NULL인 경우에 대한 고려 없음
    *rear =  tmp;
}
```

요약 정리

- 연결 리스트를 이용한 다항식의 표현 방법을 설명
- 연결 리스트로 표현한 다항식의 덧셈 알고리즘을 구현