Arrays and Structures:

The Polynomial Abstract Data Type

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2024

Outline

Outline

Ordered or Linear Lists

- Months in a year.
  - January, February, March, April, May, June, July, August, September, October, November, December.
- Days of the week.
  - Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.
- Values in a deck of card.
  - Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.
  - The items numbered i, $i+1$, ...become numbered $i+1$, $i+2$, ...

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.
  - The items numbered i, $i+1$, ...become numbered $i+1$, $i+2$, ...
- **Deleting** the ith element.

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.
  - The items numbered i, $i+1$, ...become numbered $i+1$, $i+2$, ...
- **Deleting** the ith element.
  - The items numbered $i+1$, $i+2$, ...become numbered i, $i+1$, ...

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.
  - The items numbered i, $i+1$, ...become numbered $i+1$, $i+2$, ...
- **Deleting** the ith element.
  - The items numbered $i+1$, $i+2$, ...become numbered i, $i+1$, ...

▷ What's the problem if we implement the ordered list using an "array"?

Operations on an Ordered List

- **Finding** the length, n, of the list.
- **Reading** the items from left to right (or right to left).
- **Retrieving** the ith element.
- **Storing** a new value into the ith position.
- **Inserting** a new element at the ith position.
  - The items numbered i, $i+1$, ...become numbered $i+1$, $i+2$, ...
- **Deleting** the ith element.
  - The items numbered $i+1$, $i+2$, ...become numbered i, $i+1$, ...

▷ What's the problem if we implement the ordered list using an "array"?

Goal

To build a set of functions that allow for the manipulation of polynomials.

Goal

To build a set of functions that allow for the manipulation of polynomials.

Given $A(x) = 3x^{20} + 2x^5 + 4$ and $B(x) = x^4 + 10x^3 + 3x^2 + 1$.

Goal

To build a set of functions that allow for the manipulation of polynomials.

Given $A(x) = 3x^{20} + 2x^5 + 4$ and $B(x) = x^4 + 10x^3 + 3x^2 + 1$.

- $A(x) + B(x) = ?$
- $A(x) * B(x) = ?$

Array Representation (Approach #1)

```c
#define MAX_DEGREE 101
typedef struct {
    int degree;
    float coef[MAX_DEGREE];
} poly;
```

Usage:

```c
poly a;
a.degree = n;
for (i=0; i<n; i++) {
    scanf("%f", &a.coef[i]);
}
```

Array Representation (Approach #2)

```c
#define MAX_TERMS 100
typedef struct {
    float coef;
    int expon;
} poly;
poly terms[MAX_TERMS];
int avail = 0; // available spaces
```

Example:

$$A(x) = 2x^{1000} + 1$$
$$B(x) = x^4 + 10x^3 + 3x^2 + 1$$

	start_A	finish_A	start_B			finish_B	avail
coef	2	1	1	10	3	1	
expon	1000	0	4	3	2	0	

- Storage: $\leq 2 \times$ Approach #1 when all the items are nonzero.
- n, m: # nonzeros in A and B, resp.

Function to Add Two Polynomials ($O(n+m)$ time)

```
void padd(int starta, int finisha, int startb, int finishb,
         int *startd, int *finishd) { /*add A(x) and B(x) to obtain D(x) */
    float coefficient;
    *startd = avail;
    while (starta <= finisha && startb <= finishb)
        switch(COMPARE(terms[starta].expon, terms[startb].expon)) {
            case -1: /* a expon < b expon */
                attach(terms[startb].coef, terms[startb].expon); startb++; break;
            case 0: /* equal expontents */
                coefficient = terms[starta].coef + terms[startb].coef;
                if (coefficient) attach(coefficient, terms[starta].expon);
                starta++; startb++; break;
            case 1: /* a expon > b expon */
                attach(terms[starta].coef, terms[starta].expon); starta++;
        }
    for (; starta <= finisha; starta++) /* add in remaining terms of A(x) */
        attach(terms[starta].coef, terms[starta].expon);
    for (; startb <= finishb; startb++) /* add in remaining terms of B(x) */
        attach(terms[startb].coef, terms[startb].expon);
    *finishd = avail-1;
}
```

Function to Add a New Term

```
void attach(float coefficient, int exponent) {
/* add a new term to the polynomial */
    if (avail > MAX_TERMS) {
        fprintf(stderr, "Too many terms in the polynomial\n");
        exit(1); // exit(EXIT_FAILURE);
    }
    terms[avail].coef = coefficient;
    terms[avail++].expon = exponent;
}
```

- **Issue:** Compaction is required when polynomials are no longer needed.
  - Additional time for making data movement.

Exercise: Implement the Multiplication of Two Polynomials

```
void pmult(poly a[] , poly b[], poly c[], int na, int nb, int *nc)
{
    int i, j;
    *nc = 0;
    for (i = 0; i < na; i++)
        for (j = 0; j < nb; j++) {
            // simplify the following two lines
            c[*nc].coef = a[i].coef * b[j].coef;
            c[(*nc)++].expon = a[i].expon + b[j].expon;
        }
}
```

Example (Hint)

$$\begin{aligned} A(x) &= 3x^2 + 2x + 1 \\ B(x) &= 5x^3 + 4x - 1. \end{aligned}$$

coef	3	2	1	5	4	-1	15	10	5	12	8	4	15	10	17	8	4	-3	-2	-1	15	10	17	5	2	-1
expon	2	1	0	3	1	0	5	4	3	3	2	1	5	4	3	2	1	2	1	0	5	4	3	2	1	0

Discussions