BENNETT UNIVERSITY
TIMES OF INDIA GROUP

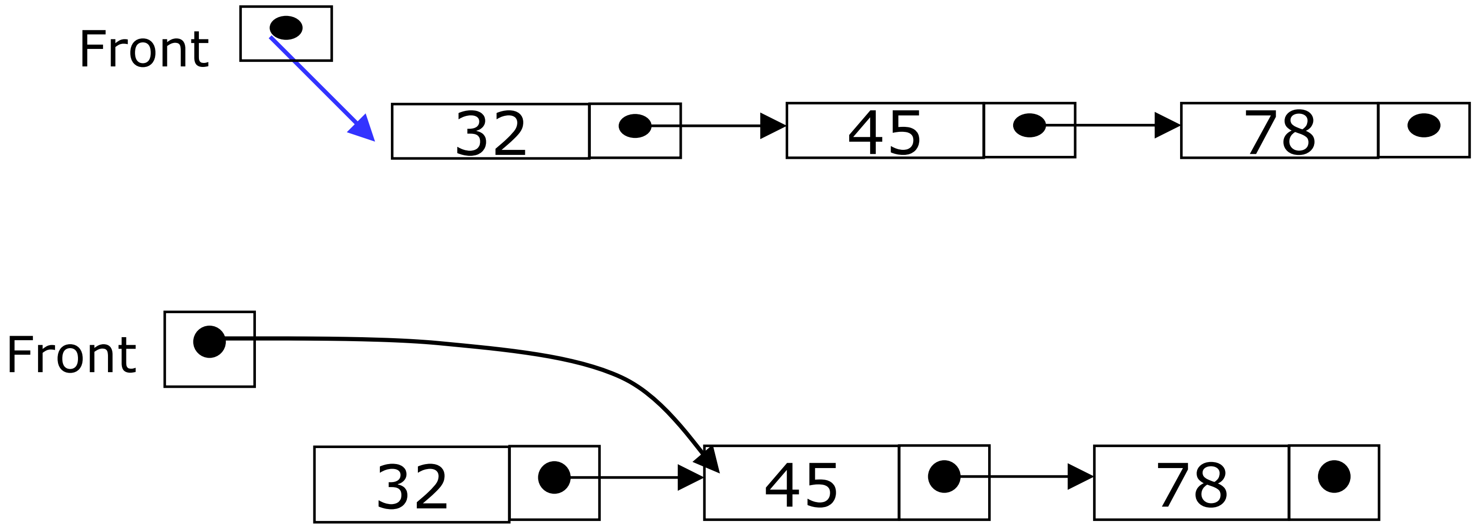
ECSE207L
DATA STRUCTURES

Dr. Tapas Badal
Dept. of CSE
Bennett University

Deletion in Linked Lists

Deleting the first node

- To delete the first element, change the link in the header



- Node ptr = Front.getLink();
  Front = ptr;

Deleting the last node

- When list is traversed it goes to the last node.
- To delete the last node we need to set the link of its previous node to null.
- But there is no way to go back from last node to its previous node.

- So we need to run two pointers.
- When one pointer reaches last node, the other pointer reaches its previous node.

Steps to Delete the last node

- Set pointer prev (previous node) to first node of the list.
- Set pointer cur (current node) to second node of the list.
- Traverse the list till cur reaches the last node
- set link for prev to null, so that last node is not reachable.

```
Node prev = Front;
Node cur = Front.getLink();
while ( cur.getLink() != null) {
      prev = cur;
     cur = cur.getLink();
  }
  prev.setLink(null);
```

(Add code to handle single node in the list.)

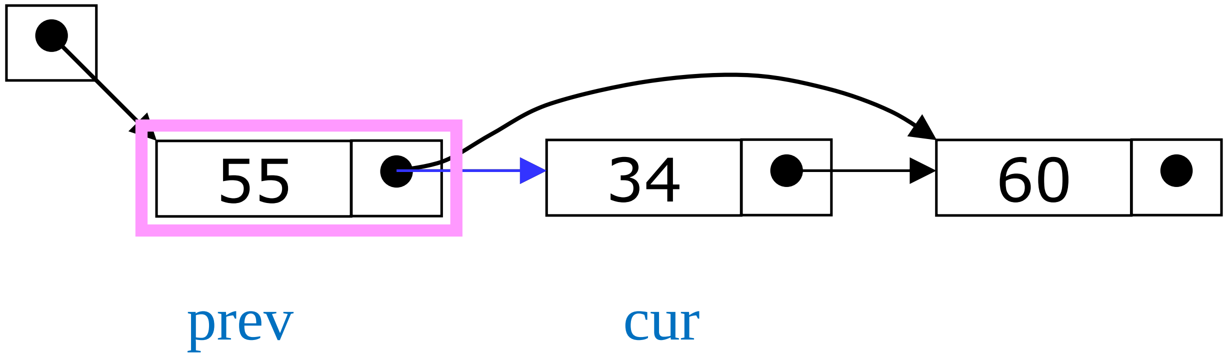
Deleting a node containing data value d

- Use prev and cur.
- Traverse till cur reaches d

Delete node containing 34

prev.setLink(cur.getLink());

- To delete a node, change the link in its previous node



Deleting a node containing data d

- Set pointer prev (previous node) to first node of the list.
- Set pointer cur (current node) to second node of the list.
- Traverse the list till cur reaches the node to be deleted.
- set link for prev to next node of cur so that current node is not reachable.

```
if (Front.getData == d) Front = Front.getLink();
else {
      Node prev = Front;
      Node cur = Front.getLink();
      while ( cur.getData() != d) {
                                        prev = cur;
            cur = cur.getLink();    }
        prev.setLink(cur.getLink());
     }
```

Time Complexity of Deletion

- Deletion means changing two pointer values. (constant time.)
- It does not matter whether Deletion is at beginning, at end or anywhere in between.

- Time complexity: O(1)

- Compare this with deletion in Arrays.

Advantages of Linked lists

- Dynamic in nature. Memory allocated at run time.
- Insertion and Deletions are constant time operations. No need to shift data as was necessary with arrays.
- Can handle polynomials.
- Other data structures like queues, stacks are easily implemented using linked lists.

Linked Lists – III
Applications and extensions

How do we interpret Polynomials

$3x^2 + 9x + 13$

- Important information:
  - Exponents involved and
  - coeff. for each exponent

Polynomial ADT

- Add two polynomials
- subtract one from another
- Multiply
- Differentiate a polynomial
- Given x find the value of the polynomial

Storing polynomials using arrays

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

- $5 + 2x + 3x^2$
- [5 2 3]
- $7x + 8$
- [8 7 0]
- Store only the coefficients in proper place

Polynomial ADT

- Array Implementation:
- $p1(x) = 8x^3 + 3x^2 + 2x + 6$
- $p2(x) = 23x^4 + 18x - 3$

p1(x)

6	2	3	8

0 2

p2(x)

-3	18	0	0	23

0 2 4

Index represents exponents

Adding two Polynomials

Add

- $5 + 2x + 3x^2$
- $7x + 8$

Issue with arrays

- $p3(x) = 16x^{21} - 3x^{5} + 2x + 6$

6	2	0	0	-3	0	…………	0	16

WASTE OF SPACE!

- This is why arrays aren't good to represent polynomials

Storing Polynomials using linked lists

- Let us now see how two polynomials can be added.
- Let P1 and P2 be two polynomials.
  - stored as linked lists
  - Each node contains exponent and co-eff values
  - in sorted (decreasing) order of exponents
- The addition operation is defined as follows
  - Add terms of like-exponents.

Representing a polynomial using a linked list

- Store the coefficient and exponent of each term in nodes
  - int [] item1 = {5, 12}
  - int [] item2 = {2, 9}
  - int [] item3 = {-1, 3}

$$5x^{12} + 2x^{9} - x^{3}$$

$$\boxed{5 \mid 12} \rightarrow \boxed{2 \mid 9} \rightarrow \boxed{-1 \mid 3}$$

Operations on Polynomials

- We have P1 and P2 arranged in a linked list in decreasing order of exponents.
- We can scan these and add like terms.
  - Need to store the resulting term only if it has non-zero coefficient.
- The number of terms in the result polynomial P1+P2 need not be known in advance.
- We'll use as much space as there are terms in P1+P2.

Adding two polynomials

- $4x^3 + 10x^2 + 5x + 3$
- $x^6 + 8x^3 \qquad\qquad + 2x + 1$
- ___________________________

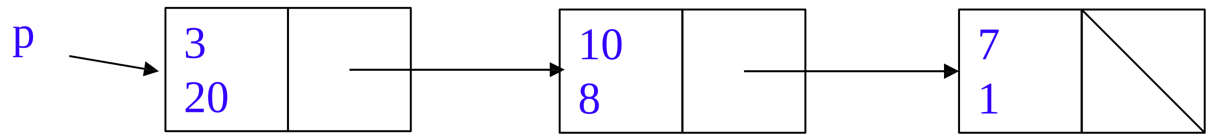
- $x^6 + 12x^3 + 10x^2 + 7x + 4$

- One extra node created

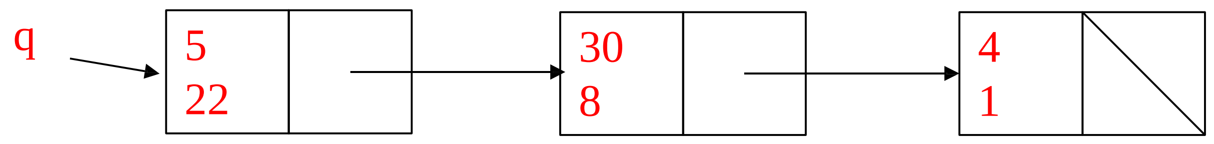
Addition of Two Polynomials using linked lists

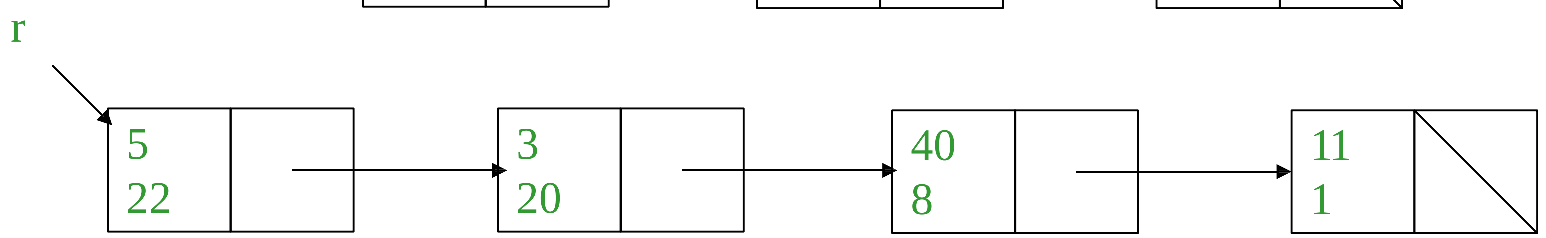
- One pass down each list: $\theta(n+m)$

$7x+10x^8+3x^{20}$



$4x+30x^8+5x^{22}$





Adding two polynomials

- $4x^3 + 10x^2 + 5x + 3$
- $x^6 + 8x^3 - 5x + 1$
- ______________________________

- $x^6 + 12x^3 + 10x^2 + 4$

- Only 4 nodes in the resultant polynomial

Multiplying two Polynomials

- Let us consider multiplication
- Can be done as repeated addition.
- So, multiply P1 with each term of P2.
- If there are M terms in P2, there will be M polynomials
- Add the resulting M polynomials.

THANKYOU

@csebennett   cse_bennett