Lecture 4
Linked List

Bo Tang @ SUSTech, Fall 2022

Our Roadmap

- Linked List Definition
- Linked List Operators
- Illustration Example

Representing a Sequence of Data

- An ordered collection of items (position matters)
  - Array, lists, stacks, and queues
- What did you study before? Array!
- Advantages of using an array
  - Easy and efficient access to any item in the sequence
  - item[i]: return the i-th element in array item
  - Every item can be accessed in constant time
  - This feature of arrays is known as “random access”
  - Very compact (in terms of memory)
- Disadvantages of using an array ?

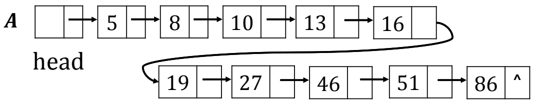
Disadvantages of an Array

- Have to specify an initial array size
- Resize an array is possible, but not so easy
- Difficult to insert/delete elements at arbitrary positions
  - Delete 10 in array A, time complexity?

A	5	8	10	13	16	19	27	46	51	86
A	5	8		13	16	19	27	46	51	86
A	5	8	13	16	19	27	46	51	86	

A Linked List

- Alternative Representation of a sequence. Example:

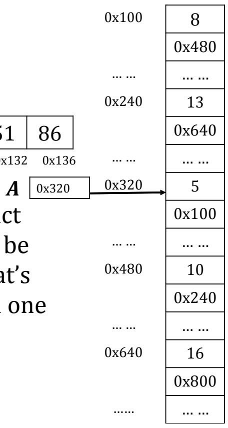


- A linked list stores a sequence of elements in separate nodes
- Each node contains: a single item, a “link” to the node containing the next item: 13 →
- The last node in the linked list has a link value of “NULL”: 86 ^
- The linked list as a whole is represented by a variable that hold a reference to the first node (e.g., ***A***)

Array vs. Linked List in Memory

- In an array, the elements occupy consecutive memory locations:

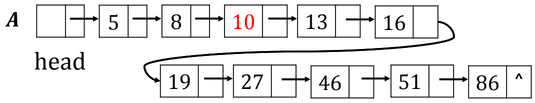
5	8	10	13	16	19	27	46	51	86
0x100	0x104	0x108	0x112	0x116	0x120	0x124	0x128	0x132	0x136



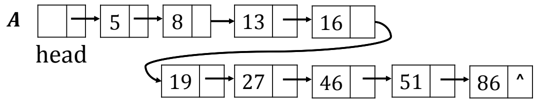
- In linked list, each node is a distinct object. The nodes do NOT have to be next to each other in memory. That’s why we need the links to get from one node to the next.

Features of Linked List

- It can grow without limit (not fixed length)
- Easy to insert/delete an element
- Delete 10 in Linked List A, before:

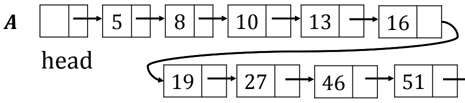


- After:

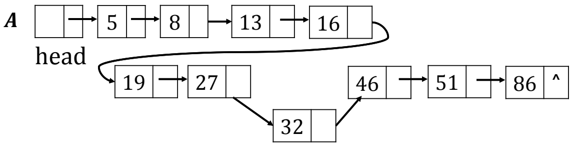


Features of Linked List

- Insert 32 in Linked List A, before:



- After:



- Time Complexity?

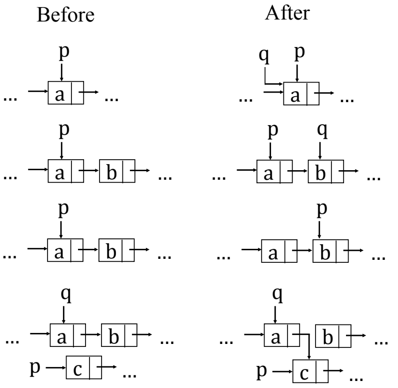
Features of Linked List

- Disadvantages of Linked List
  - They do not provide random access
  - Need to “walk down” the list to access an item
  - The links take up additional memory
  - Not compact (in terms of Memory)
- Linked List vs. Array
  - Space complexity
  - Time Complexity: Insert, Delete, Find

Our Roadmap

- Linked List Definition ✓
- Linked List Operators
- Illustration Example

Basic Operators of Linked List



① q ← p

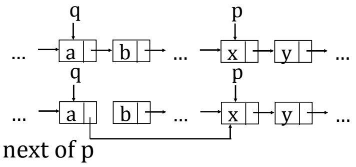
② q ← next of p

③ p ← next of p

④ next of q ← p
(a)

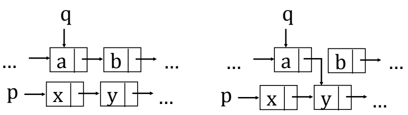
Basic Operators of Linked List

(b)

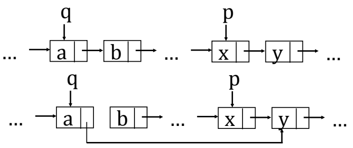


⑤ next of q ← next of p

(a)



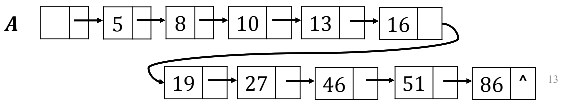
(b)



Traverse a Linked List

- Many tasks require us to traverse or “walk down” a linked list
- Recursion Pseudocode
- **Algorithm:** traverse(A):

```
if (A=NULL)
      return
else
    print A.value
    traverse(A.next)
```



Traverse a Linked List

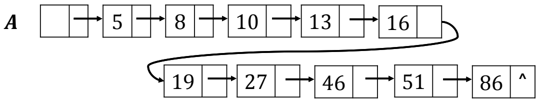
- It can also be done using iteration (for loops, while loops, etc.)
- Iteration Pseudocode
- **Algorithm:** traverseIteration(A):

```
node trav ← A
While (trav != NULL)
      print trav.value
      trav ← trav.next
```

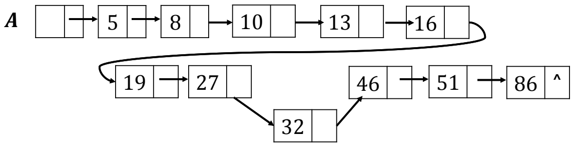
- We use iteration in the following operators, but you can try to use recursion to implement these operators.

Inserting an Item at Position i

- Insert 32 in Linked List A at position 8, before:



- After:



- How to do that?

Inserting an Item at Position i

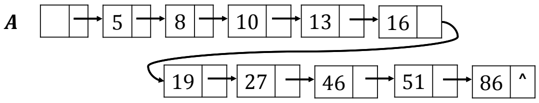
- **Problem:** insert node q in Linked List A at Position i
- **Algorithm:** insertNode(A, node q, i):

```
a ← 0, node p ← A,
while (i-1 > a)
      p ← p.next
      a ← a + 1
tmp ← p.next
p.next ← q
q.next ← tmp
return A
```

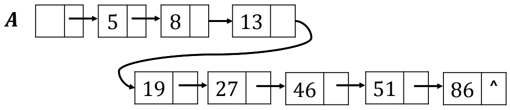
- Time Complexity: **O(n)**
- Space Complexity: **O(1)**

Deleting an Item at Position i

- Delete position 5 in Linked List A, before:



- After:



- How to do that?

Deleting an Item at Position i

- **Problem:** delete node in Linked List A at Position i
- **Algorithm:** deleteNode(A, i):

```
a ← 0, node p ← A,
while (i-1 > a)
      p ← p.next
      a ← a + 1
p.next ← p.next.next
return A
```

- Time Complexity: **O(n)**
- Space Complexity: **O(1)**

Finding an Item at Position i

- **Problem:** Find value x in Linked List A
- **Algorithm:** findValue(`A, x`):

```
a ← 0, node p ← A,
while (p!=NULL)
        if (x = p.value)
                return p
        p ← p.next
return -1
```

- Time Complexity: **O(n)**
- Space Complexity: **O(1)**

Updating an Item at Position i

- **Problem:** Update nodes with value x to y in Linked List A
- **Algorithm:** updateNodes(`A, x`):

```
a ← 0, node p ← A,
while (p!=NULL)
        if (x = p.value)
                p.value ← y
        p ← p.next
return A
```

- Time Complexity: **O(n)**
- Space Complexity: **O(1)**

Our Roadmap

- Linked List Definition ✓
- Linked List Operators ✓
- Illustration Example

Operators on **polynomials**

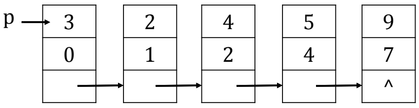
- **Polynomials**: $p(x) = p_0 + p_1x + p_2x^2 + \dots + p_nx^n$
- a set of ordered pairs of $<p_i, i>$ where p_i is the coefficient and i is the exponent.
- We use linked list store the $< p_i, i >$ pairs of p(x)
- Without loss of generality, we skip all nodes w/ $p_i = 0$
- Node representation:

```
node polyItem{
    float coef    // record p_i
    int expo // record exponent
    node next // reference to next polyItem
}
```

- **Question**: how about use array?

Finding degree of a Polynomials

- **Polynomials**: $p(x) = 3 + 2x + 4x^2 + 5x^4 + 9x^7$

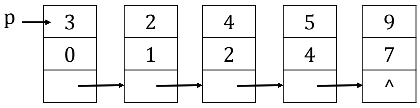


- Degree of p(x): 7
- **Algorithm:** findDegree(p):

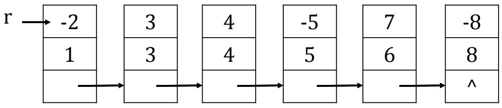
```
node tmp ← p
While (tmp.next != NULL)
        tmp ← tmp.next
return tmp.expo
```

Adding two polynomials

- $p(x) = 3 + 2x + 4x^2 + 5x^4 + 9x^7$

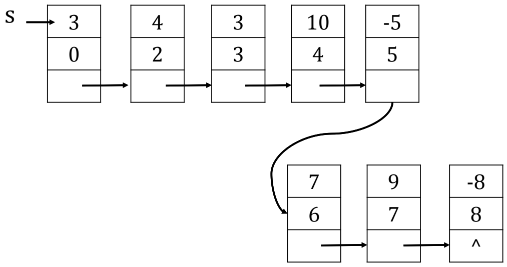


- $r(x) = -2x + 3x^3 + 5x^4 - 5x^5 + 7x^6 - 8x^8$



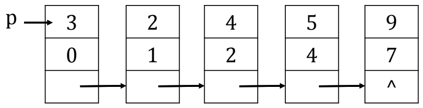
Adding two polynomials

- $s(x) = p(x) + r(x)$
  $= 3 + 4x^2 + 3x^3 + 10x^4 - 5x^5 + 7x^6 + 9x^7 - 8x^8$

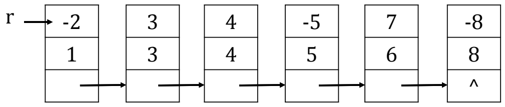


Subtracting two polynomials

- $p(x) = 3 + 2x + 4x^2 + 5x^4 + 9x^7$

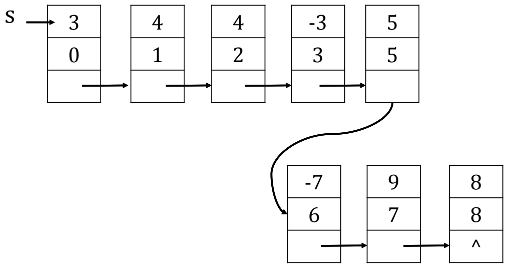


- $r(x) = -2x + 3x^3 + 5x^4 - 5x^5 + 7x^6 - 8x^8$



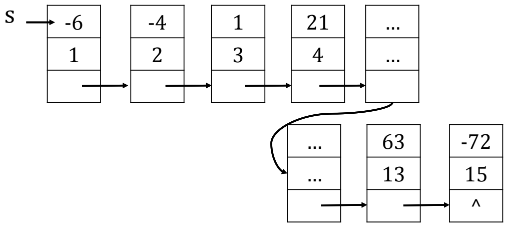
Subtracting two polynomials

- $s(x) = p(x) - r(x)$
  $= 3 + 4x + 4x^2 - 3x^3 + 5x^5 - 7x^6 + 9x^7 + 8x^8$



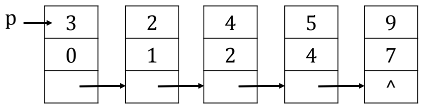
Multiplying two polynomials

- $p(x) = 3 + 2x + 4x^2 + 5x^4 + 9x^7$
- $r(x) = -2x + 3x^3 + 5x^4 - 5x^5 + 7x^6 - 8x^8$
- $s(x) = p(x)*r(x)$
  $= -6x - 4x^2 + x^3 + 21x^4 - 3x^5 + 31x^6 + 9x^7 + 11x^8 - 41x^9 + 30x^{10} + 45x^{11} - 85x^{12} + 63x^{13} - 72x^{15}$

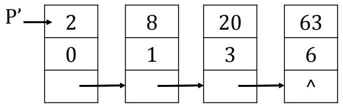


Differentiating of a polynomial

- $p(x) = 3 + 2x + 4x^2 + 5x^4 + 9x^7$

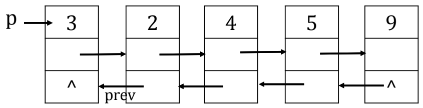


- $p'(x) = 2 + 8x + 20x^3 + 63x^6$



Other variants of Lined List

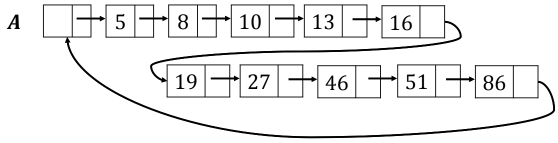
- Double linked list



  - add a prev reference to each node: refers to the previous node
  - allow us to "back up" from a given node

Other variants of Lined List

- Circular linked list



- Is it a empty list? head.next = head ?
- Is it the end of list? tmp.next = head?

Our Roadmap

- Linked List Definition ✓
- Linked List Operators ✓
- Illustration Example ✓

Thank You!