Linked List

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 17, 2020

Singly Linked List

Application: Polynomial Computation

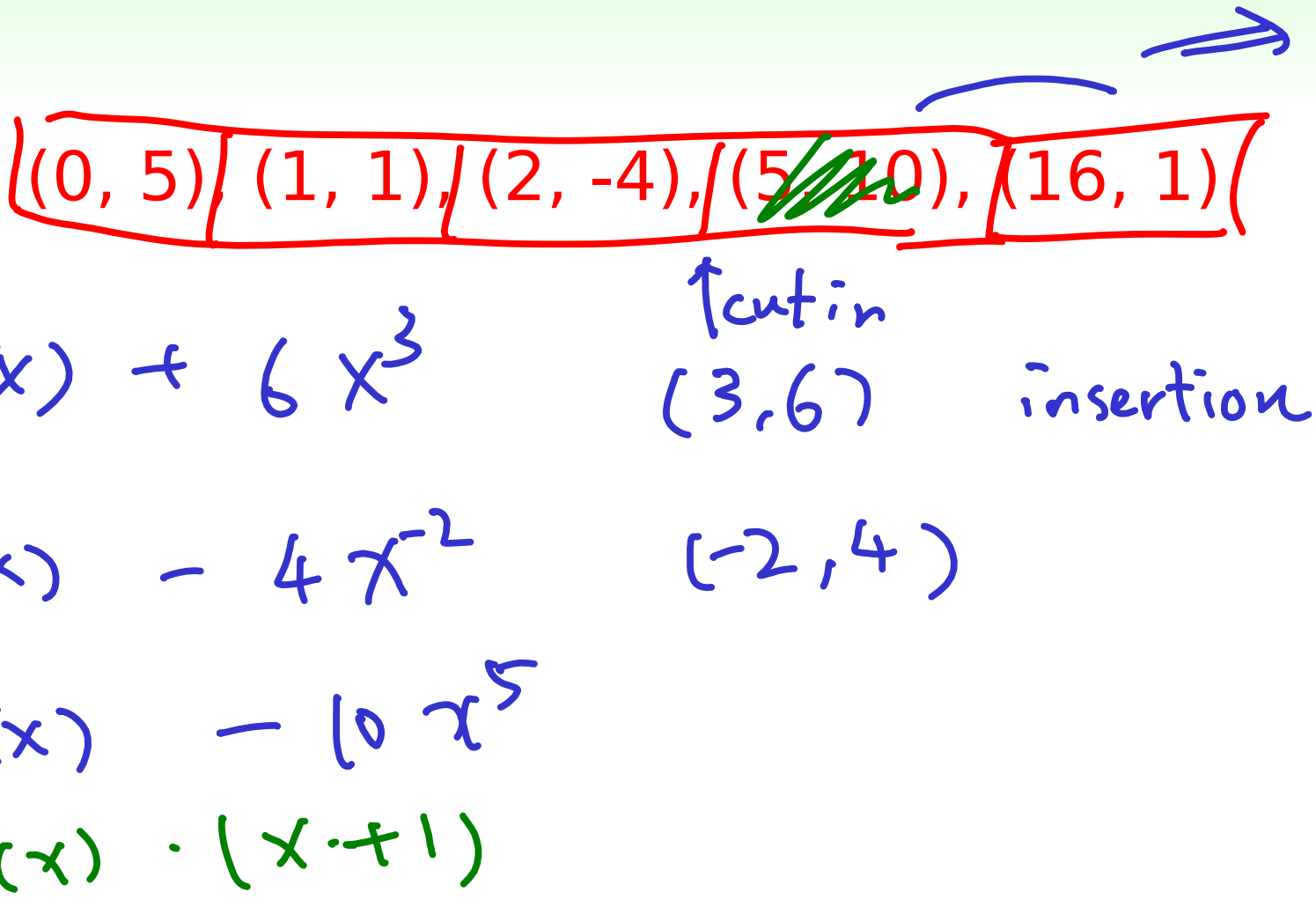
f(x) = 5 + x - 4 x^2 + 10 x^5 + x^16
g(x) = 3 - 6 x^5

5 x^0
1 x^1
-4 x^2
10 x^5
1 x^16

f: (0, 5), (1, 1), (2, -4), (5, 10), (16, 1)
g: (0, 3), (5, -6)

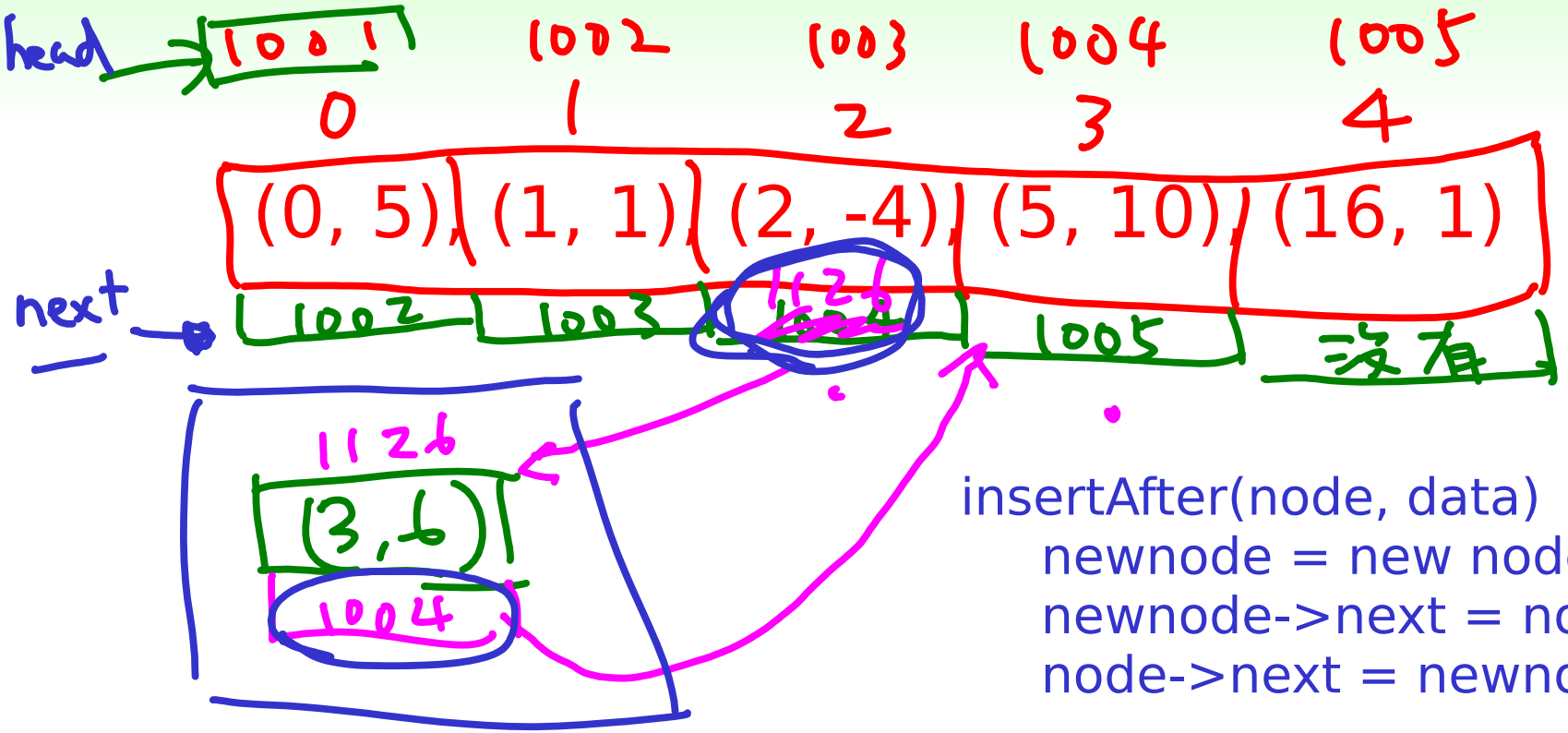
solution 0: use ordered array on (exponent, coefficient)

Issues of (Ordered) Array for Polynomial Computation

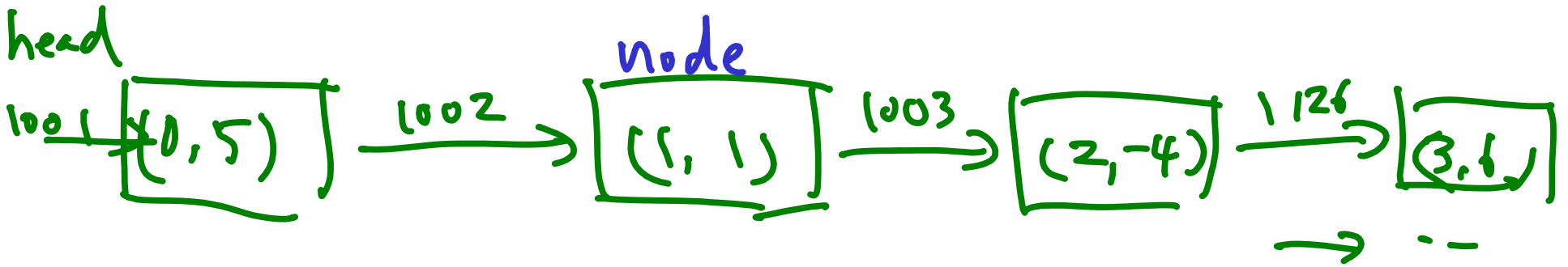


ordered (consecutive) array:
not flexible for resizing/insertion/removal

Solution 1: Singly Linked List for Flexible Insertion




```
insertAfter(node, data)
    newnode = new node(data)
    newnode->next = node->next
    node->next = newnode
```

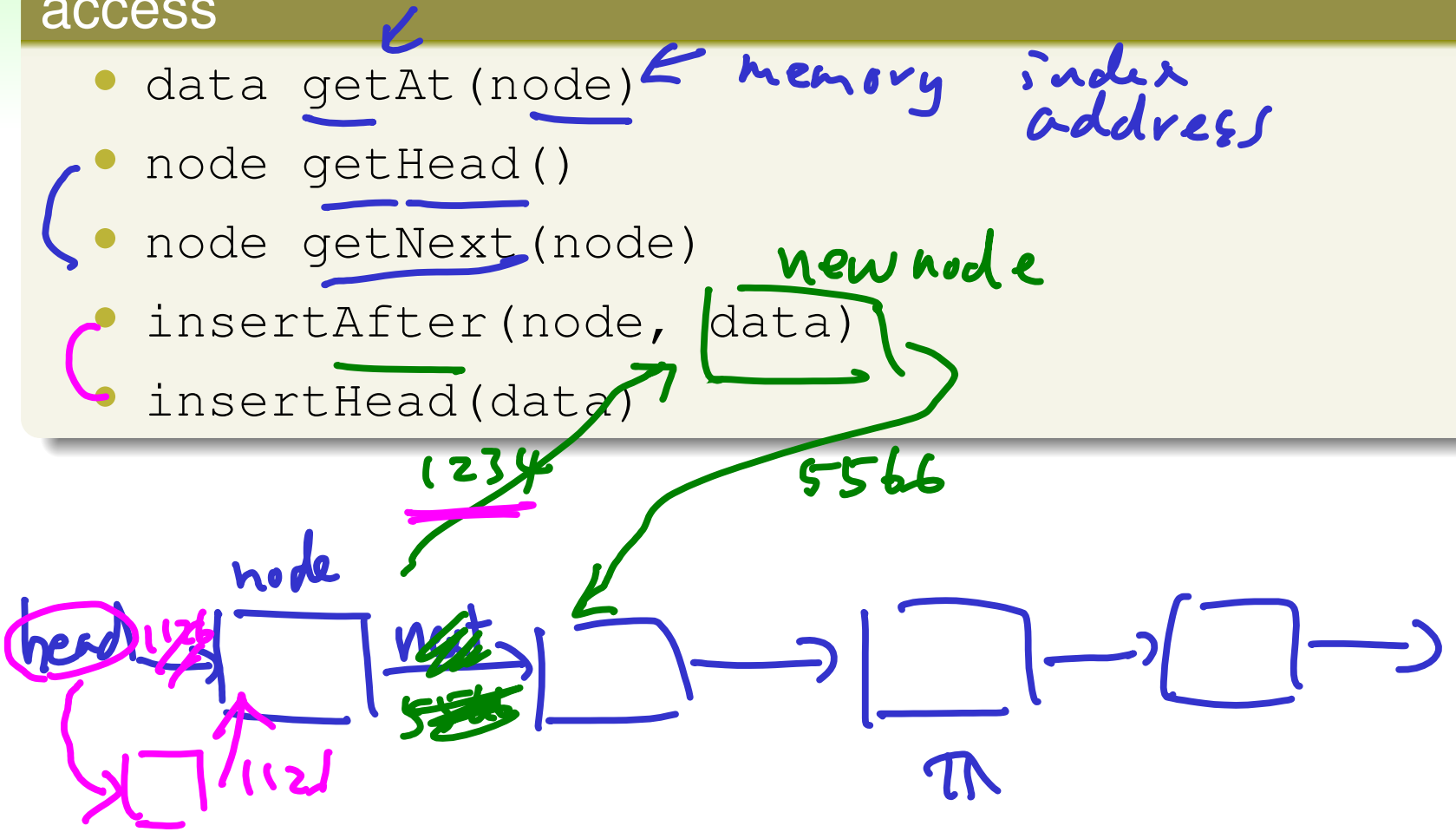


overhead of ~~next~~ $\Leftrightarrow$ flexible `insertAfter`

Singly Linked List as Abstract Data Structure: Access

access

- `data getAt(node)`
- `node getHead()`
- `node getNext(node)`
- `insertAfter(node, data)`
- `insertHead(data)`



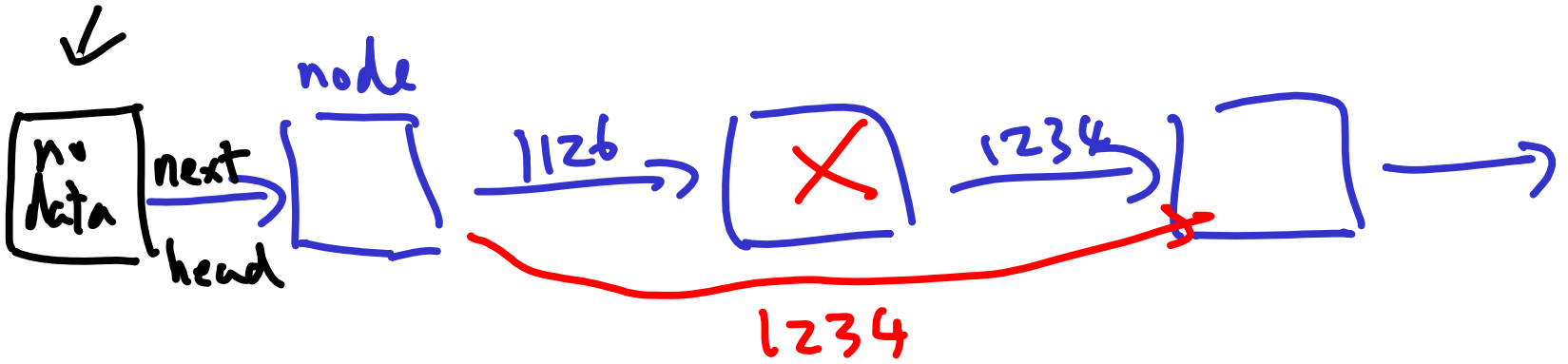
linked list: sequential access; array: random access

Singly Linked List as ADT: Maintenance

maintenance

- `construct(length)`: trivial
- `updateHere(node, data)`: trivial
- `removeAfter(node)`: simple
- `removeHead`: simple

insertAfter

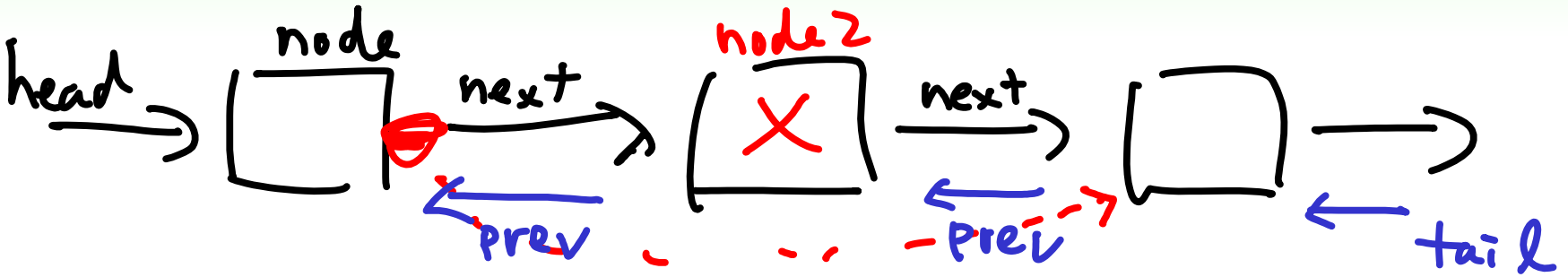



```
tofree = node.next
node.next = node.next.next
free(tofree)
```

think: dummy head node or not?

Doubly Linked List

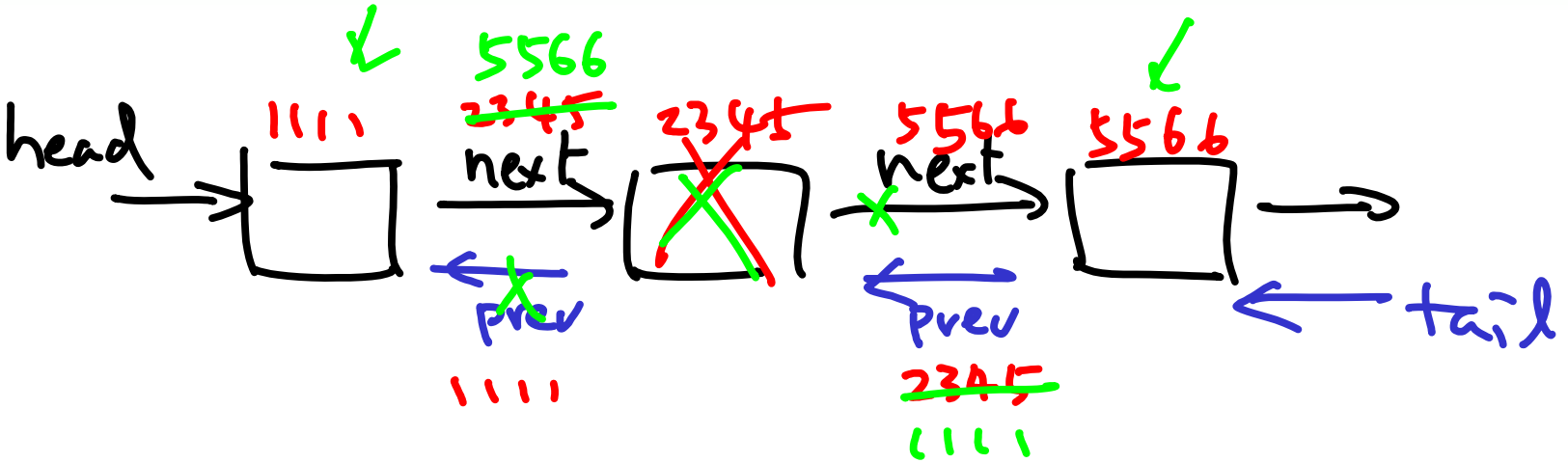
`removeHere` for Singly Linked List



removeAfter(node)
removeHere(node2)

`removeHere` (and `insertHere`): hard for singly linked list

Doubly Linked List: More Flexible `removeHere`



overhead of `prev` $\Leftrightarrow$ flexible `removeHere`

Iterator for Sequential Access

singly linked list:

```
for(node = head; node != end; node = node->next){
...
}
```

reverse doubly linked list

```
for(node = tail; node != end; node = node->prev){
...
}
```

array

```
for(index = 0; index <= tail; index++){
...
}
```

iterator: abstraction of array index, linked list node and more!

C++ STL List: a Doubly Linked List

access

- `node getHead(): list.begin()`
- `node getNext(node): iterator++`
- `data getAt(node): (*iterator)`
- `insertHere(node, data): list.insert(iterator, data))`

and more!

maintenance

- `updateHere(node, data): (*iterator = data)`
- `removeHere(node): list.erase(iterator)`

and more!

STL list and its iterator:
a more "structured" way of using doubly linked list

Linked List for Sparse Vectors

Application: Sparse Vector in Scientific Computing

"vector": [0, 3.5, 0, 0, 7, 4.2, 9]
number of dimension * size(double)

"sparse vector": number of non-zero * size(pairs)
(2, 3.5), (5, 7), (6, 4.2), (7, 9)

$1 + 2 * x^5 + 10 * x^{1000}$

polynomial: can be viewed as special case of sparse vector