Lists, Stacks, and Queues

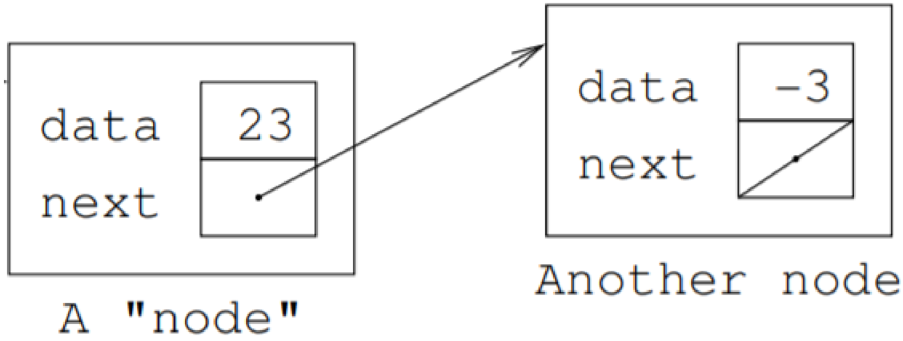
Abstract data types (ADTs)

- A mathematical model of an object that stores data
  - Defined by its behaviour from the point of view of the user
  - *Not* defined by how it is actually implemented
- Example: lists
  - Can insert, append, delete, and access elements
  - The actual implementation can be, for example, a linked list or an array

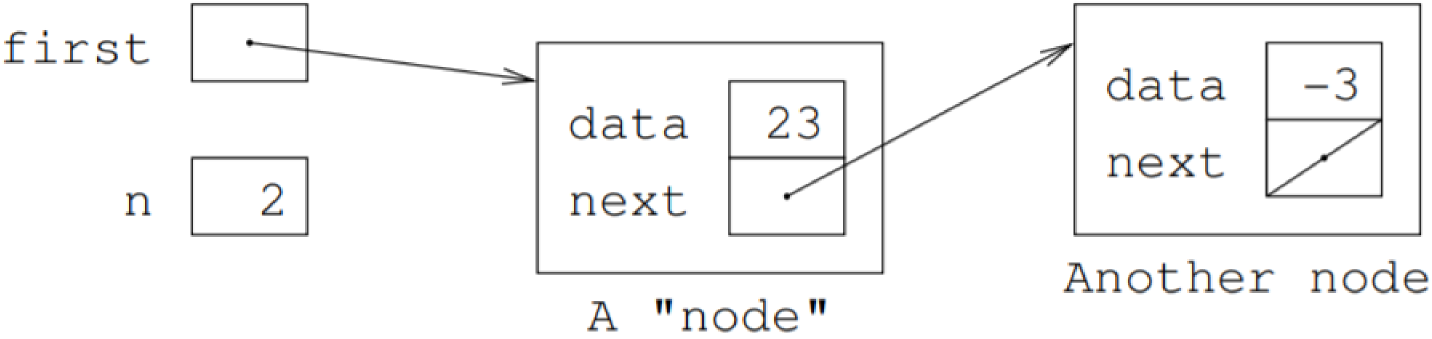
Data structures

- A set ways of storing data that implement a data type
  - Example: a linked list
  - Example: array
  - Example: structs

(Singly) linked lists

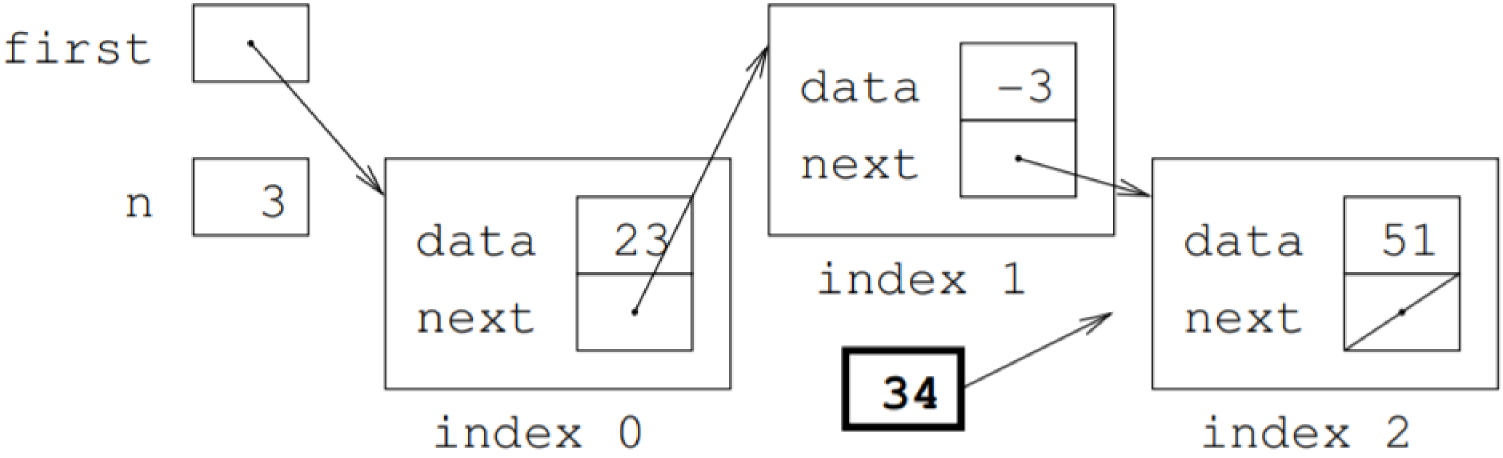


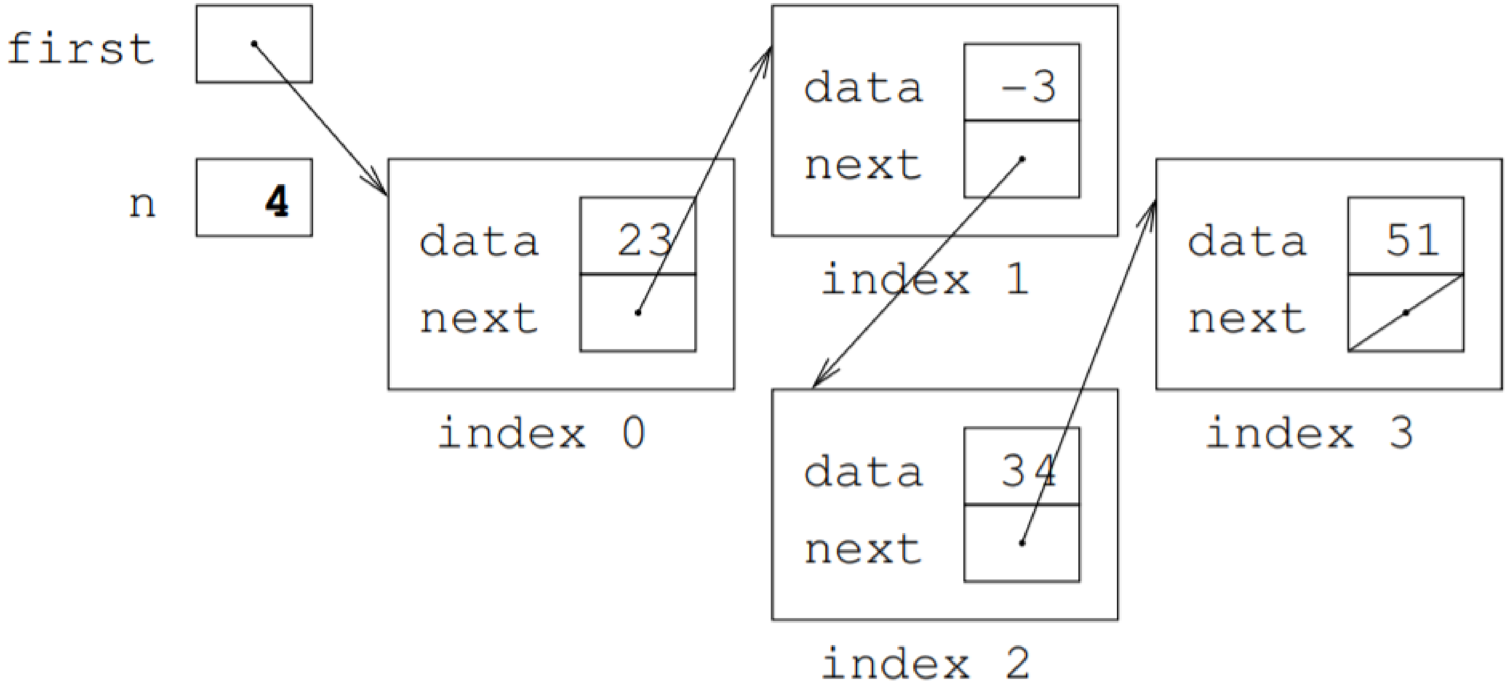
(Singly) linked lists with pointer to head



Insert

- Want to insert the value 34 at index 2



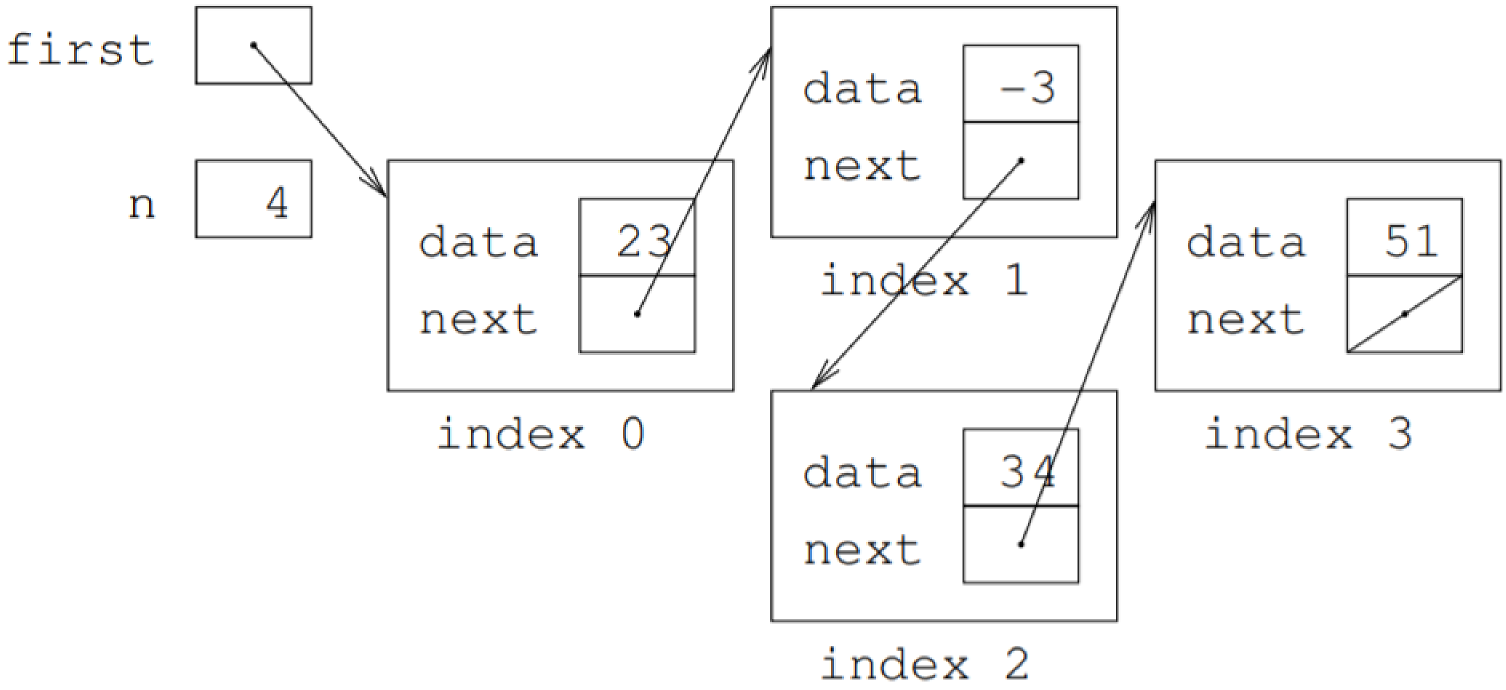

first
n
4
data
23
next
index 0
data
-3
next
index 1
data
34
next
index 2
data
51
next
index 3

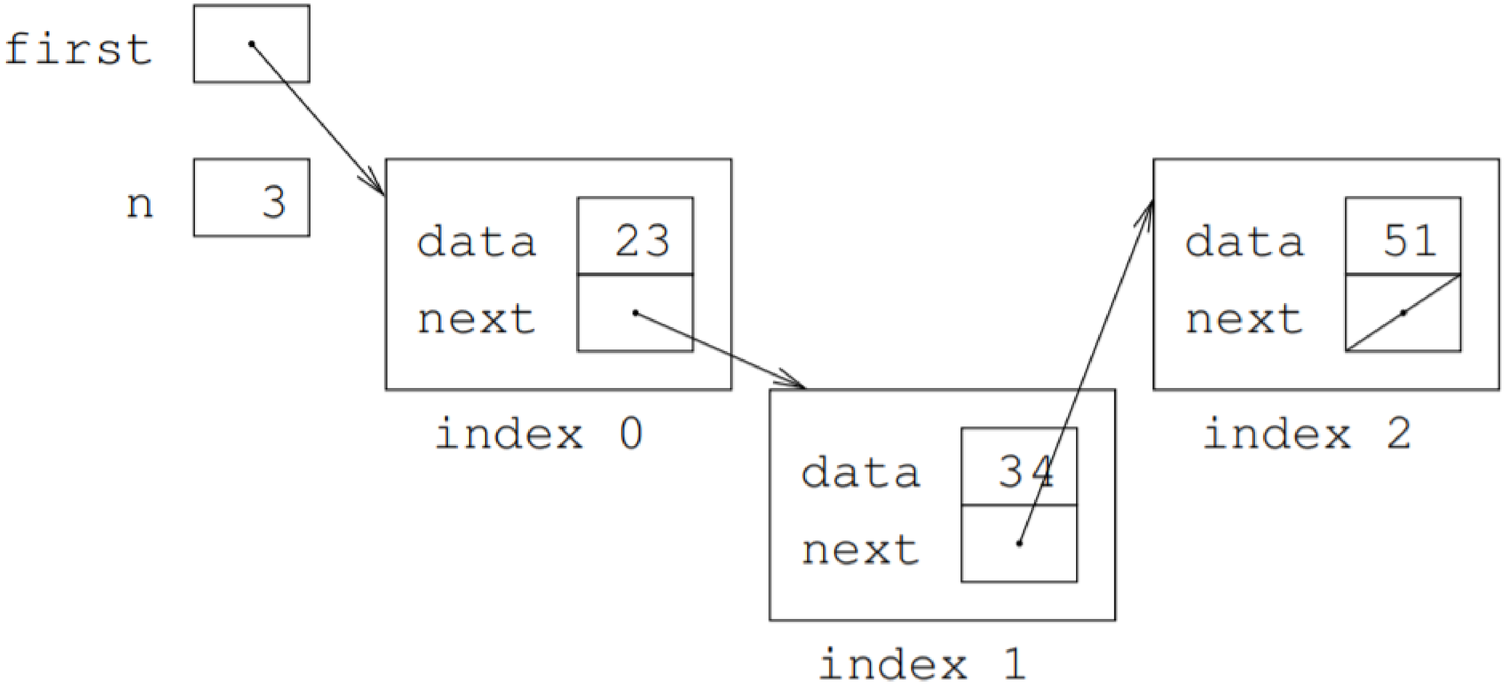
Complexity of insert

- Create and connect a new node: O(1)
    - (Assuming we have the pointer to the previous node)

Remove

- Suppose we want to remove the value at index 1



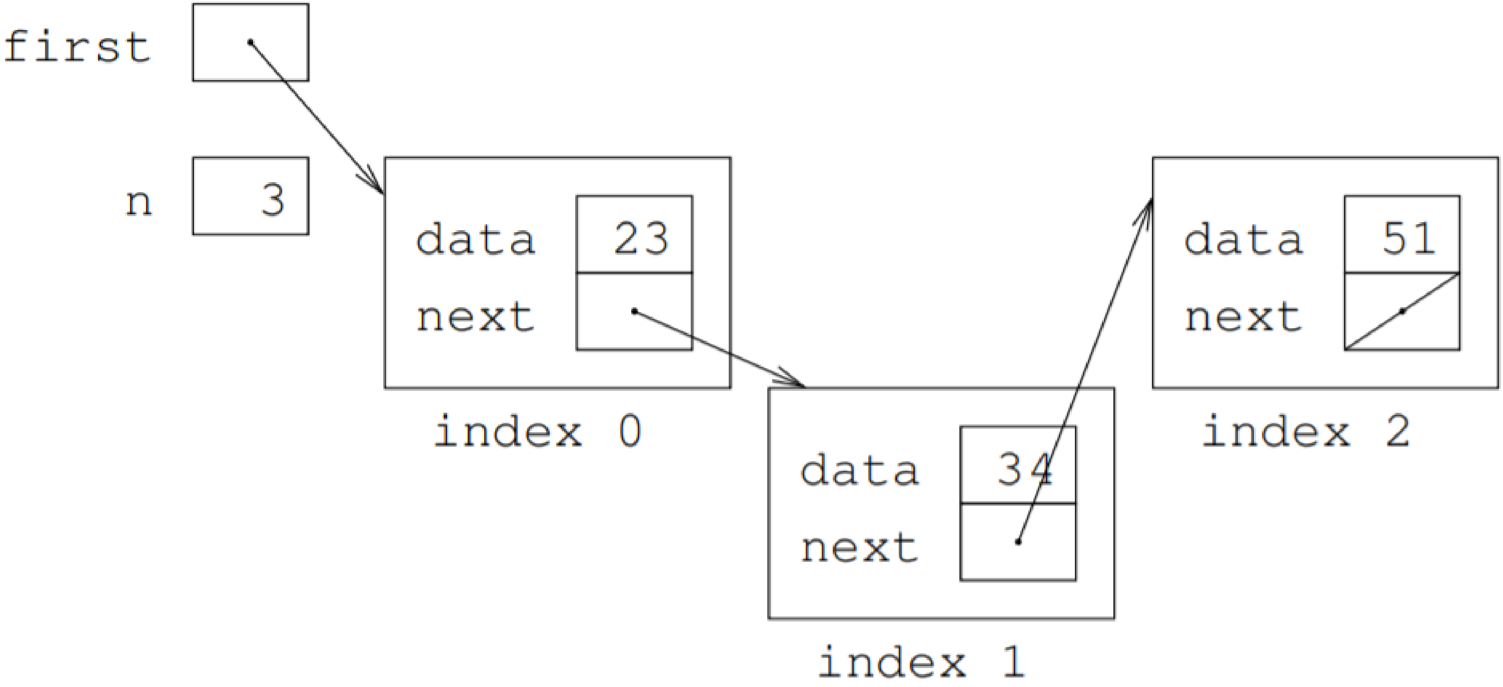

first
n
3
data
23
next
index 0
data
34
next
index 1
data
51
next
index 2

Complexity of remove

- O(1)
    - (Assuming we have the pointer to the previous node)

get

- Want to get the value at index 2
  - O(n) assuming we don't have the pointer to the node



Arrays

- Get: O(1)
- Insert at index i:
    - Copy arr[(i+1):n] to arr+i+2
        - O(n)
    - Copy value to arr[i]
    - May need to copy entire array to another block
    - O(n) time
- Remove index i:
    - Copy arr[(i+1):n] to arr+I
    - O(n) time

	Array	Linked list
Insert	O(n)	O(1)*
Remove	O(n)	O(1)*
Get	O(1)	O(n)

Stack

- An ADT
- A list with the operations
  - push: append an element to the end of the list
  - pop: remove an element from the end of the list and get it value
- "LIFO": last in, first out
- Array implementation: push is O(n), pop is O(1)
  - In practice, average time to push is not O(n)
- Linked list implementation: push is O(1), pop is O(1)
  - Need to store the last node of the linked list

Queue

- A list with the operations:
  - Enqueue: append an element to the end of the list
  - Dequeue: remove the element from the start of the list, get its value
- “FIFO”: first in, first out
- Linked list implementation: enqueue and dequeue are both O(1)
  - Need to store the head and the “tail” of the linked list

Queue: circular array implementation

- Store elements in an array, keep the index of both the first and the last element
  - For an array of size N, store element (N-1+m) in element (N-1+m) mod N
  - Enlarge the array when necessary
- Enqueue: O(n) time (but less in practice)
- Dequeue: O(1) time