5118006-03 Data Structures

Linked List

12 Apr 2024

Shin Hong

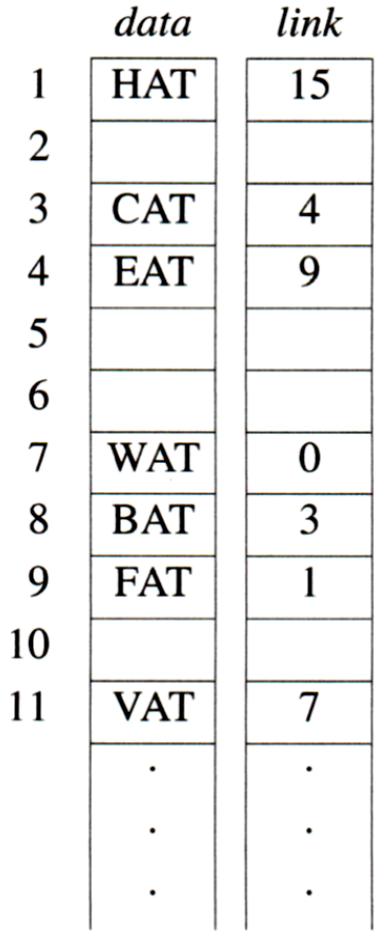
Array vs. Linked List

- An **array** stores data objects fixed distance apart in a continuous memory region
  - Pros: possible to randomly access a certain index
  - Cons: changing the index of an object (by insertion and deletion) can result in many operation

- A **linked list** stores an element with a pointer of the successor to form a chain of elements
  - Pros: insertion or deletion of a specific element can be done without moving other elements
  - Cons: an element at an index can be accessed only through its predecessor

Singly Linked List (1/2)

- Array-based implementation
  - use a data element array, a link array, and the index of first element
  - the next element of *data*[*k*] is found at *link*[*k*]
  - link[k] is zero when it does not link to any data element
- Example: `arrlist.c`

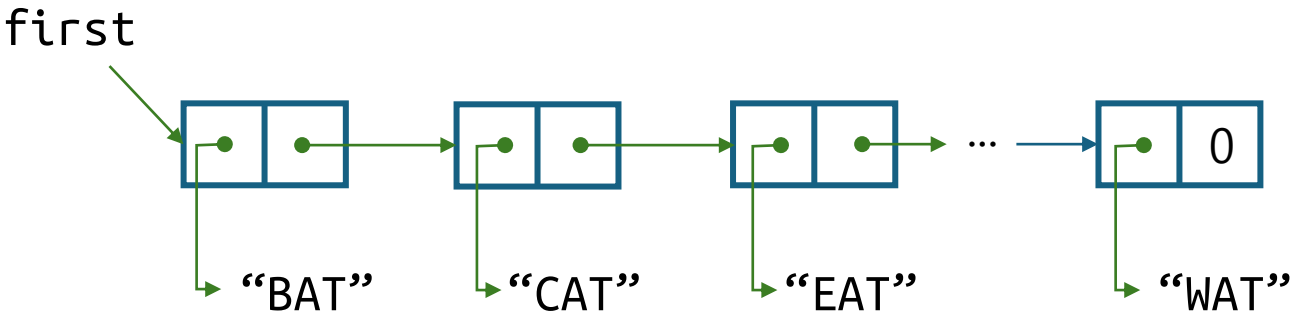


	data	*link*
1	HAT	15
2		
3	CAT	4
4	EAT	9
5		
6		
7	WAT	0
8	BAT	3
9	FAT	1
10		
11	VAT	7
	.	.
	.	.
	.	.

Singly Linked List (2/2)

- Pointer-based implementation
  - use a node with a data field and a pointer field which points to the next node
    - self-referential structure
  - use a pointer to indicate the starting node
    - or, place a header node

- Example: `llist.c`



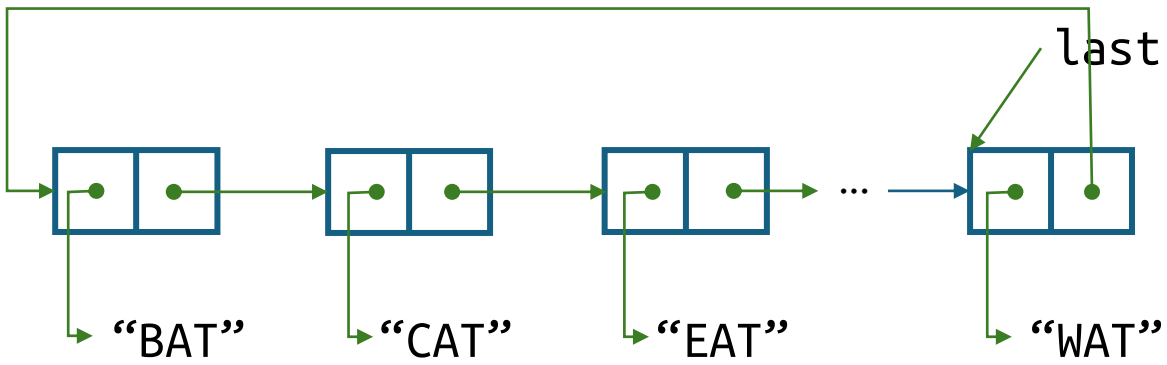
Linked List Operation

- size (length)
- insert
- retrieve
- delete

Circular LinkedList

- Let the last node points to the first node
  - instead of pointing NULL

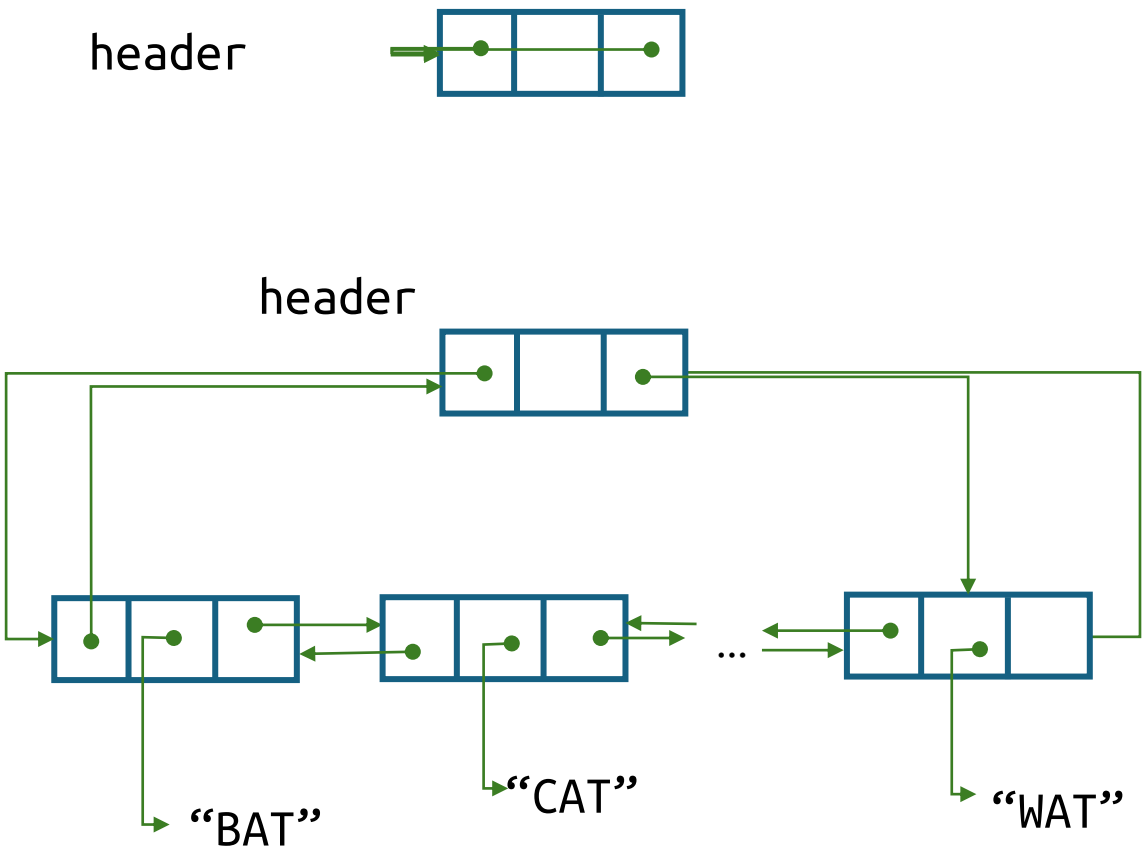
- A list maintains a pointer to the last node
  - rather than the first node
  - the next of the last node is the first node
  - beneficial to the insert operation



Doubly Linked List

- A node has two pointers, one to the next node and the other to the previous node.
  - efficient to iterate over the list forward and backward
  - efficient at modifying the list
- Doubly linked list is the most widely used form of list data structures in real-world
- Implementation
  - the list has a special “header node” which stores the beginning points of forward and backward traversals

Example



Equivalence Class

- Two elements, x and y, belongs to the same equivalent class if the following condition is satisfied
  (1) x is identical to y (x = y)
  (2) x and z belong the same equivalent class and, z and y belong the same equivalent class
- Write a program that receives the pairs of integers each of which belong to the same equivalent class, and finds out all equivalent classes
  - 0 ≡ 4
  - 3 ≡ 1
  - 6 ≡ 10
  - 8 ≡ 9
  - 7 ≡ 4
  - 6 ≡ 8
  - 3 ≡ 5
  - 2 ≡ 11
  - 11 ≡ 0