CSE 674 Advanced Data Structures

More on Lists

Andrew C. Lee

EECS, Syracuse

Contents

- Internal of a linked list
- Manipulation of linked lists: doubly linked list example
- Implementations of Stacks and Queues
- The *vector* data structure

Internals of a linked list

- Revisit the `pointers.cpp` example
  - reference operator: `&` and the dereference operator: `*`
  - requires addtional storage
  - data sharing; ease of insertion/deletion of elements
- External lists
  - separate control information from data information
  - We may not know a priori all future uses of the lists

Implementation of a doubly linked list

- Node Structure
- Basic operations:
  - `LIST-SEARCH(L,k)`
  - `LIST-INSERT(L,x)`
  - `LIST-DELETE(L,x)`
  - The use of a *sentinel*
- How to check your codes (which use pointers)

Reference: Cormen Chapter 10, Section 2

Stacks

1. Array implementations

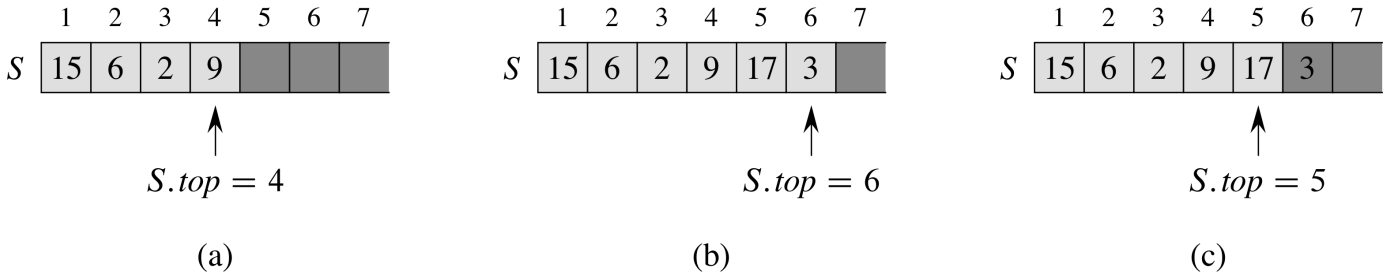

Figure : Sketches of a Stack (array based implementation from Cormen's text

2. The *push* and *pop* operations

Stacks: Examples and Discussions

Coding Example See Demo code 1 (Brass) for an example of implementing a stack via an array.

Discussions: How will you implement a stack via a linked list ?

Stacks

Some remarks:

1. Insertion and deletion are performed at the same end
2. LIFO
3. Typically use in
   - processing goals such that goal is completed only when all the subgoals are completed
   - backtracking
   - the evaluation of arithemetic expressions via reverse polish notation
4. Avoid/handle overflows and underflows

Queues

1. Array implementations

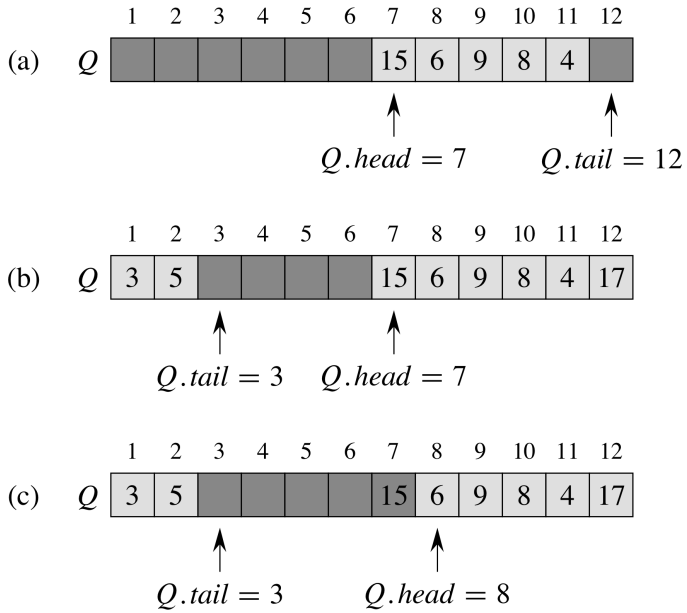

Figure : Sketches of a Queue (An array based implementation from Cormen's text

2. The *enqueue* and *dequeue* operations

Queue: Implementation

Discussions: How will you implement a queue via a singly linked list ?

Discussions: How will you implement a queue via a doubly linked list ?

Queues

Some remarks:

1. Note the "circular nature" of the queue in our array implementations
2. Note the criteria one use to distinguish an empty queue and a full queue
3. A *deque* denotes a double-ended queue (generalized stacks and queues).

Discussions: Will you implement a deque via a *singly-linked* list ?