Lists, Stacks, & Queues: A Comparison & Big-O

Instructor - Andrew S. O’Fallon

CptS 122 Washington State University

Lists, Stacks, & Queues (I)

- Lists
  - Insertions and deletions may be made anywhere in the array or linked list; insertions at end (O(n)), insertions at front (O(1))
  - Support searching for data; Best case: 1st item is the target (O(1)); Worst case: target is not in the list (O(n))
  - Valuable for storing data that may be reordered and/or searched
- Stacks – last in, first out (LIFO)
  - Insertions and deletions at one end of the array or linked list only (i.e. generally the front)
- Queues – first in, first out or first come, first serve (FIFO)
  - Insertions at the end of the array or linked list, and deletions at the front

Stacks: Arrays Vs. Linked Lists (I)

- Array Implementation
  - Advantages:
    - Indexing is available (random access) – can keep track of the top of the stack with an index, which is also the size of the stack
    - Insert and delete data from one end – thus could insert (push) and delete (pop) an item in constant time (O(1)) using the index
  - Disadvantages:
    - Generally, the maximum number of items that the stack could contain must be predetermined
      - The use of a vector solves this problem

Stacks: Arrays Vs. Linked Lists (II)

- Linked List Implementation
  - Advantages:
    - Insert and delete data from one end – thus could insert (push) and delete (pop) an item in constant time (O(1)), by inserting at the front or deleting the front, respectively
    - Providing the maximum number of items for the stack does *not* need to be predetermined – can continue to allocate memory for items as needed
  - Disadvantages:
    - Must explicitly manage the memory on the heap – malloc () and free ()

Queues: Arrays Vs. Linked Lists (I)

- Array Implementation
  - Advantages:
    - Indexing is available (random access) – can keep track of the front of the queue with an index, and also keep track of the end of the queue with another index
    - Insert (enqueue) at the end of the queue in constant time (O(1))
  - Disadvantages:
    - Delete (dequeue) requires that all items in the array be shifted over, which requires linear time (O(n))
    - Generally, the maximum number of items that the queue could contain must be predetermined
      - The use of a vector (introduced later in the course) solves this problem

Queues: Arrays Vs. Linked Lists (II)

- Linked List Implementation
  - Advantages:
    - Insert at one end and delete data from other end – thus could insert (enqueue) and delete (dequeue) an item in constant time (O(1)), by deleting at the front (one pointer) or inserting at the end (another pointer), respectively
    - Providing the maximum number of items for the queue does *not* need to be predetermined – can continue to allocate memory for items as needed
  - Disadvantages:
    - Must explicitly manage the memory on the heap – malloc () and free ()

References

- P.J. Deitel & H.M. Deitel, *C: How to Program* (8th ed.), Prentice Hall, 2016
- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (7th Ed.)*, Addison-Wesley, 2013

Collaborators

- Jack Hagemeister