Assignment 3

Double-ended Queues

Implement the deque ADT (textbook, section 6.3) in Java using the following data structures:

1. circular arrays
2. doubly linked lists with header and trailer dummy nodes
3. doubly linked lists with **NO** header or trailer dummy nodes

Important Notes

- **Do not modify the given class and method definitions.**
- **Do not add packages to the submitted Java files.** Your programs may not compile if packages exist in the code.
- **Do not add I/O statements (e.g., scanner, print) to the submitted Java files.** I/O statements will mess up our automatic grading programs and produce incorrect outputs. Your programs will get zero in such cases.
- Class *EmptyDequeException* is implemented in file EmptyDequeException.java.
- The input data given in the above main programs are only examples for your testing. We will use different data sets to mark your programs.
- Assume that all inputs are valids.
- To compile and run the programs, use the following commands and the appropriate files:
  ```
  javac ArrayDeque.java
  javac ArrayMain.java
  java ArrayMain
  ```
- Submit only 3 files: ArrayDeque.java, ListDeque.java and AnotherListDeque.java.

1. Circular arrays

Follow the design discussed in the lecture.

When a new element is to be inserted into a deque and the array is full, do not throw en exception. Extend the array by doubling its capacity and insert the element into the new array. Use an initial array size of INIT_CAPACITY = 8.

Shrink the array by half the current capacity N when the number of elements in the deque falls below N/4 (**but the minimum capacity, INIT_CAPACITY, should always be 8**).

Complete the "blank" methods in file ArrayDeque.java. Use the main program ArrayMain.java to test your code.

Notes: In this implementation,

1. *front* is the index of the cell storing the first element of the queue (which is the next candidate to be removed by a *removeFirst* operation), unless the queue is empty (in which case *front* = *rear*).
2. *rear* is the index of next available (unused) array cell. For example, if *front* is 2, and the queue currently contains 3 elements (for example, 154, 28 and 79), then *rear* is 5.
3. the queue is empty when *front* = *rear*.
4. the queue is considered full when there are *N* - 1 elements in the queue (not *N* elements), where *N* = *capacity*, because *rear* should always point to an unused cell.
5. using the above rules, the size of the queue is given by (*N* - *front* + *rear*) mod *N*

2. Doubly linked lists with header and trailer dummy nodes

Follow the design discussed in the lecture and textbook.

Complete the "blank" methods in file ListDeque.java. Class *DNode* is implemented in file DNode.java. Use the main program ListMain.java to test your code.

3. Doubly linked lists with no header or trailer dummy nodes

Use variables *head* and *tail* to keep track of the front and rear of the deque. Variables *head* and *tail* are not dummy nodes: they contain actual data elements of the deque.

Complete the "blank" methods in file AnotherListDeque.java. Class *DNode* is implemented in file DNode.java. Use the main program AnotherListMain.java to test your code.

Posted 12 Feb. 2020
Last updated 12 Feb. 2020