
From CS261

Main: ProgrammingAssignment3

Due Friday, February 3, 2012

Linked List Variations

Problem 1: Bag implementation

Consider the doubly-linked list implementation of the bag ADT in Worksheet 19 (with front and back sentinels). In this implementation, we pass a linked-list pointer variable to the bag interface functions. This method lacks flexibility in the sense that if you change the implementation of the bag to a dynamic array implementation, we would have to rewrite all the function calls where the bag interface is used. To avoid this problem, we use a "wrapper". For example, the wrapper for a bag would be a `struct bag` which contains the linked list implementing the bag. All the bag interface functions, such as `contains()`, `remove()` etc. now take as a parameter, a pointer to an instance of `struct bag`. Using the signatures provided in `LinkedList.h` (which also contains the wrapper for bag), complete the implementation of the bag ADT. The bag interface should implement the `add (addToBag())`, `contains (containsBag())` and `remove (removeFromBag())` functions. These functions, in turn, should call the appropriate functions of the underlying implementation of the bag - which will also have to be implemented. For this part of the assignment, complete the implementation of the linked list with the bag wrapper.

Files needed:

- `LinkedList.c`
- `LinkedList.h`
- `LinkedListMain.c`
- `makefileLinkedList.txt` (after you download, rename it to `makefile`)

Problem 2 - Comparison

The file `main.c` contains code which does the following

1. Takes an integer argument (say `n`) from the command line and adds that many elements to the bag
2. Calls `contains()` with each integer 0 to `n` in a loop.
3. Prints the time taken by these `contains()` calls in milliseconds

Your job is to compare, for various values of `n`,

1. Running time for bag `contains` - from (Problem 1)
2. Running time for bag `contains` - implemented using a dynamic array. The dynamic array must have a capacity of 1000 to start with. The policy is to double the size when the array is full. We're providing the dynamic array implementation below.

Plot the running times of the two programs for various values of `n` starting from `n=1000` to `n=200,000`; doubling the number of elements on each run.

- Which of the implementations is the fastest?
- Would you expect anything to change if the loop performed `removeBag` instead of `contains`? If so, what?

Files needed:

- files from the previous problem
- **dynamicArray.h**
- **dynamicArray.c**
- **dynamicArrayMain.c**(note, to run this on other data structures, you'll need to change the include at the top)
- **makefiledynamicArray.txt** (after you download, rename it to `makefile`)

Problem 3: Implementation of the Deque ADT using a Circularly linked list

For this problem, you will implement the Deque ADT with a Circularly-Doubly-Linked List with a Sentinel. As you know, the sentinel is a special link, does not contain a value, and should not be removed. Using a sentinel makes some linked list operations easier and cleaner in implementation. This list is circular, meaning the end points back to the beginning, thus one sentinel suffices. The header file and the implementation file for this approach are `cirListDeque.h` and `cirListDeque.c`, respectively. Complete the functions in `cirListDeque.c`.

Files needed

- **cirListDeque.h**
- **cirListDeque.c**
- **makefilecirListDeque.txt** (rename to `makefile` after downloading)
- **listDequeTest.c**

Grading

Problem 1 (40 pts)

- `init` 2
- `_addLink` 4
- `addBack` 2
- `addFront` 2
- `front` 2
- `back` 2
- `_removeLink` 4
- `removeFront` 2
- `removeBack` 2
- `empty` 2
- `print` 4
- `contains` 2
- `remove` 2

- addToBag 2
- removeFromBag 2
- bagContains 2
- isEmptyBag 2

Problem 2 (10 pts)

- plot 8
- Question 2

Problem 3 (50 pts)

- init 4
- _createLink 4
- _addLinkAfter 4
- addBack 3
- addFront 3
- front 3
- back 3
- _removeLink 4
- removeFront 3
- removeBack 3
- _free 4
- isEmpty 2
- print 4
- reverse 6

What to submit

Submit linkedList.c and cirListDeque.c. For problem 2, turn in a PDF file with the graph and answers to the two questions. Do not make modification to the headers and the signatures therein.

Retrieved from

<https://secure.engr.oregonstate.edu/classes/eecs/winter2012/cs261/index.php/Main/ProgrammingAssignment3>

Page last modified on January 28, 2012, at 12:06 AM