Worksheet 22 - Group 1

Worksheet Group 1 Members

Marc Clinedinst: clinedim@onid.oregonstate.edu
Kelby Faessler: faesslek@onid.oregonstate.edu
James Fitzwater: fitzwatj@onid.oregonstate.edu
Tom Gariepy: gariepyt@onid.oregonstate.edu
Sean Reilly: reillys@onid.oregonstate.edu
Joseph Struth: struthj@onid.oregonstate.edu

Worksheet 22: Constructing a Bag Using a Linked List

In this assignment, we implement two different functions which control the behavior of a bag data structure built on top of a linked list. More specifically, we implement the `linkedListContains` and `linkedListRemove` functions. These are described in detail below, with comments where necessary.

```
/*
     This function checks to see if a bag contains a particular
     value. Given that this operation should not occur on an empty
     bag, the function first checks to make sure that the bag is not
     empty.  If it is empty, an assertion error occurs.  Otherwise,
     the function loops iterates through the bag.  If the target
     value is encountered during this iteration, the function
returns
     1 to indicate that the target value was found.  Otherwise, the
     function returns 0 to indicate that the target value was not
     found.
*/
int linkedListContains(struct linkedList *q, TYPE e) {
     assert(!LinkedListIsEmpty(q));

     struct dlink *current = q->frontSentinel->next;

     while (current->next) {
          if (current->value == e) {
               return 1;
          }
          current = current->next;
     }

     return 0;
```

```
}
/*
      This function removes a value from the bag.  Given that this
      operation should not occur on an empty bag, the function should
      not occur on an empty bag, the function first checks to make
      sure that the bag i snot empty. If it is empty, an assertion
      error occurs.  Otherwise, the function iterates through the
bag.
      If the target value is encountered during this iteration, the
      the function removes the value from the bag.  If the target
      value is not found, the bag is left unchanged.
*/
void linkedListRemove(struct linkedList *q, TYPE e) {
      assert(!LinkedListIsEmpty(q));

      struct dlink *current = q->frontSentinel->next;

      while (current->next) {
            if (current->value == e) {
                  _removeLink(q, current);
            }
            current = current->next;
      }
}
```

In addition to the function definitions above, the worksheet also requires that we respond to the two questions below.

1. What were the algorithmic complexities of the methods `addLink` and `removeLink` that you wrote back in linked list for Deque?

Both the `addLink` and `removeLink` functions have O(1) complexity.

2. Given your answer to the previous question, what are the algorithmic complexities of the three principle Bag operations?

The `linkedListContains` and `linkedListRemove` functions have O(n) complexity, because each function has to perform a linear search before the target value can be found or removed.

Piazza Discussion

https://piazza.com/class/ib2kus4hsie528?cid=118