Weekplan: Introduction to Data Structures

The 02105+02326 DTU Algorithms Team

Reading

Introduction to Algorithms, Cormen, Rivest, Leisersons and Stein (CLRS): Introduction to Part III + Chapter 10.

Exercises

1 Stacks and Queues

1.1 CLRS [w] 10.1-1.

1.2 Exercise 5.1 in the exam set from 2011.

1.3 CLRS 10.1-2.

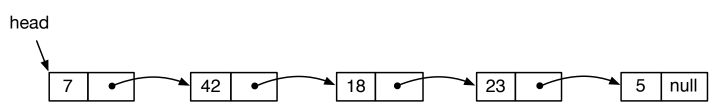
1.4 CLRS [w] 10.1-3.

1.5 CLRS 10.1-6.

2 Algorithms on Linked Lists Look at the algorithms Foo and Bar and the linked list below. Solve the following exercises.

```
FOO(head)
x = head
c = 0
while x ≠ null do
   x = x.next
   c = c + 1
end while
return c
```

```
BAR(x, s)
if x == null then
   return s
else
   return BAR(x.next, s + x.key)
end if
```



2.1 [w] Run Foo(*head*) by hand.

2.2 [w] Explain what Foo computes.

2.3 Run Bar(*head*, 0) by hand.

2.4 Explain what Bar does.

3 Implementation of Linked Lists Assume x is an element in a singly linked list as described in the lecture. Solve the following exercises.

3.1 [w] Assume x is not the last element in the list. What is the result of the following code snippet?

```
x.next = x.next.next;
```

3.2 [w] Let t be a new element that is not already in the list. What is the result of the following code snippet?

```
t.next = x.next;
x.next = t;
```

3.3 [w] Suppose we now swap the order of the statements:

```
x.next = t;
t.next = x.next;
```

What happens now? The same as above?

4 Implementation of Stacks and Queues Solve the following exercises.

4.1 [†] Implement a stack that can contain integers using a singly linked list.

4.2 [†] Implement a queue that can contain integers using a singly linked list.

5 Sorted Linked Lists Let L be a singly linked list consisting of n integers in sorted order. Solve the following exercises.

5.1 Give an algorithm to insert a new integer in L such that the list is still sorted afterwards.

5.2 Professor Gørtz suggests one can improve the insertion algorithm by using binary search. Is she right?

6 List Reversal Give an algorithm to reverse a singly linked list, ie. produces a singly linked list with the elements in the reversed order. Your algorithm should run in $\Theta(n)$ time and not use more than constant extra space (in addition to the list).

7 Dynamic Arrays and Stacks We are interested in implementing a stack using a dynamic array without a maximum size for the array in the beginning. Solve the following exercises.

7.1 [$*$] Generalize dynamic arrays to also support stacks that shrinks (ie. supports both PUSH and POP operations). The running time of any sequence of n operations must be $\Theta(n)$ and at any point in time your solution should use linear time in the number of elements currently in the stack.

7.2 [$**$] Show how one can obtain $O(1)$ time per stack operation using dynamic arrays and linear space in the number of elements currently in the stack. Only consider growing stacks and thus ignore POP. *Hint:* Consider how the work can be evenly distributed over all operations.