Weekplan: Introduction to Data Structures

Philip Bille Inge Li Gørtz

Reading

Introduction to Algorithms, 4th edition, Cormen, Rivest, Leisersons, and Stein (CLRS): Introduction to Part III + Chapter 10 + *Introduction to Programming in Python: An Interdisciplinary Approach*, Sedgewick, Wayne, and Dondero (IPP): Chapter 4.3.

Exercises

1 Stacks and Queues

1.1 [w] Show the execution of the sequence PUSH(4), PUSH(1), PUSH(3), POP, PUSH(8), POP on an initially empty stack stored in an array with capacity 6.

1.2 Show how to implement two stacks in a single array $A[0, N-1]$ such that neither stack overflows unless the total number of elements in both stacks exceeds N. The PUSH and POP operations should take $O(1)$ time.

1.3 [w] Apply the sequence ENQUEUE(4), ENQUEUE(1), ENQUEUE(3), DEQUEUE, ENQUEUE(8), DEQUEUE on an initially empty queue stored in an array with capacity 6.

1.4 [$*$] Show how to implement a queue using two stacks (and no other data structures). Analyze the running time of the queue operations.

2 Exercise 5.1 in the exam set from 2011 Let S be a stack. Perform the following operations from left to right: a letter i means *Push(S,i)* and $*$ means *Pop(S)*.

D * T U * * I N * F O R * M * A T I K

Which of the following sequences of letters corresponds to the sequence of letters popped (returned by *Pop(S)*) while performing the above operations:

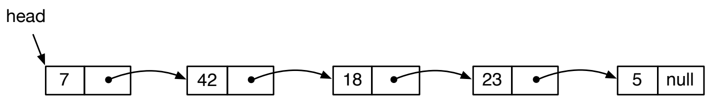
$\boxed{A}$ D U T I R M $\boxed{B}$ D U T N R M K I T A O F $\boxed{C}$ D U T N R M

$\boxed{D}$ D T U I N F $\boxed{E}$ D U T N R M I T A O F $\boxed{F}$ D U T N O M

3 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 $x \neq null$ 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
```



3.1 [w] Run FOO(*head*) by hand.

3.2 [w] Explain what FOO computes.

3.3 Run BAR($head, 0$) by hand.

3.4 Explain what BAR does.

4 Manipulating Linked Lists Assume `x` is an item in a linked list described in the lecture. Solve the following exercises.

4.1 [w] Assume `x` is not the last item in the list. What is the result of the following code snippet?

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

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

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

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

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

What happens now? Is it the same as above?

5 Doubly Linked Lists A *doubly linked list* extends the standard linked list (often also called *singly linked list*) by adding a reference `prev` to the previous item in the list.

5.1 Assume x is an item in a doubly linked list. Show how to delete x from the list in constant time.

5.2 Assume x and y are items in a doubly linked list such that x precedes y in the list. Show how to delete the entire contiguous sublist of items from x to y in constant time.

6 Implementation of Stacks and Queues Solve the following exercises.

6.1 [†] Implement a stack containing integers using a linked list.

6.2 [†] Implement a queue containing integers using a linked list.

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

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

7.2 A friend suggests using binary search to speed up searching in a sorted linked list. Will this work?

8 List Reversal Give an algorithm to reverse a linked list, i.e., produce a linked list with the elements in the reversed order. Your algorithm should run in $O(n)$ time and not use more than constant extra space (in addition to the list).

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

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

9.2 [$**$] 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.

10 Death by Light Switches 32 prisoners are sentenced to life in prison in solitary confinement. The evil warden proposes a deal to keep them (or him?) entertained. The warden has a bowl containing the cell numbers of all the prisoners. Each day, he randomly chooses one cell from the bowl, the corresponding prisoner is taken to the interrogation room, and the cell number is returned to the bowl. The interrogation room is empty except for k light switches (k is defined later), which the prisoner can choose to turn on or off. The prisoner may make the assertion that all 32 prisoners have been in the room. If the prisoner is correct, all prisoners will be released. If not, the prisoners are brutally executed! The prisoners are given one meeting to discuss a strategy before their communication is completely severed. Initially, all light switches are off. Solve the following exercises:

10.1 Is it possible to devise a strategy that ensures, with 100% certainty, that one of them will guess correctly and ensure their freedom if there are $k = 32$ light switches?

10.2 If $k = 5$?

10.3 $[**]$ If $k = 1$?