University of Zurich

Department of Informatics
Database Technology Group
Prof. Dr. Anton Dignös

Informatics II
Exercise 7

April 5, 2021

Goals:

- Practise operations of stacks and queues
- Implement stacks and queues using arrays
- Implement stacks and queues using linked lists
- Study and practise Deque

Abstract Data Types: Stacks, Queues

Task 1. Abstract structures of stacks and queues

a) Illustrate the result of each operation in the sequence PUSH(4), PUSH(1), PUSH(3), POP(), PUSH(8), and POP(S) on an initially empty stack S.

b) Illustrate the result of each operation in the sequence ENQUEUE(4), ENQUEUE(1), ENQUEUE(3), DEQUEUE(), ENQUEUE(8), and DEQUEUE() on an initially empty queue Q.

c) Explain how to implement two stacks in one array $A[]$ in such a way that neither stack overflows unless the total number of elements in both stacks together is n. The PUSH and POP operations should run in $O(1)$ time.

d) Explain how to implement a queue using two stacks. Analyze the running time of the queue operations.

e) Explain how to implement a stack using two queues. Analyze the running time of the stack operations.

Task 2. Implementation of stacks and queues in C

a) Write a C program that implements a stack using an array. Your C program should contain push and pop functions, and examples to call implemented functions.

b) Write a C program that implements a queue using an array. Your C program should contain enqueue and dequeue functions, and examples to call implemented functions.

c) Write a C program that implements a stack using a singly linked list. The operations PUSH and POP should still take $O(1)$ time.

d) Write a C program that implements a queue using a singly linked list. The operations ENQUEUE and DEQUEUE should still take $O(1)$ time.

e) Comparing stacks and queues using linked lists and stacks and queues using arrays.

Task 3. A double-ended queue, abbreviated to deque, allows elements added to the front and removed from the rear. We use an array of integers as the data structure for a deque of integers. Write a C program that contains the following functions:

1. addFront(), add an integer to the front
2. addRear(), add an integer to the rear
3. delFront(), remove an integer from the front
4. delRear(), remove an integer from the rear.

All four functions should have time complexity of $O(1)$. Consider how to implement these four functions. Your C program should contain examples to call these four implemented functions.