# TECHNISCHE UNIVERSITÄT DARMSTADT FACHGEBIET THEORETISCHE INFORMATIK

PROF. JOHANNES BUCHMANN NABIL ALKEILANI ALKADRI NINA BINDEL PATRICK STRUCK

# Algorithmen und Datenstrukturen



SoSe 2018

**4. Lösungsblatt** — 07.05.2018 v1.0

#### P1 Stacks

a) Illustrieren Sie jeweils das Ergebnis nach jeder Ausführung der folgenden Operationen in angegebener Reihenfolge auf demselben Stack S, das in einem Array S[1...8] gespeichert wird und geben Sie den jeweiligen Index des Elements an auf das *S.top* nach Ausführung des Befehls zeigt. Zu Anfang ist der Stack S leer und *S.top* enthält den Wert 0.

	1	2	3	4	5	6	7	8
S								

- PUSH(S,9)
- PUSH(S,4)
- PUSH(S,7)
- POP(S)
- PUSH(S,2)
- POP(S)
- POP(S)
- b) Was passiert jeweils wenn der Befehl POP(S) auf einem leeren Stack oder PUSH(S,x) auf einem bereits vollen Stack ausgeführt wird?
- c) Geben Sie die Laufzeiten für die Befehle STACK-EMPTY(S), PUSH(S,x) und POP(S) in O-Notation an.

# Lösung.

a) Siehe Tabelle

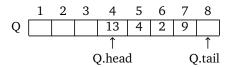
1	2	3	4	5	6	7	8	
9								S.top = 1
9	4							S.top = 2
9	4	7						S.top = 3
9	4							S.top = 2
9	4	2						S.top = 3
9	4							S.top = 2
9								S.top = 1

- b) Es wird ein (Stack-)underflow bzw. ein (Stack-)overflow Fehler zurückgeworfen.
- c) Die einzelnen Operationen haben konstante Laufzeit.

STACK-EMPTY(S)	O(1)
PUSH(S,x)	O(1)
POP(S)	O(1)

#### P2 Queues

Illustrieren Sie analog zu Aufgabe P1 jeweils das Ergebnis nach jeder Ausführung der folgenden Operationen auf einer Queue, die in einem Array Q[1...8] gespeichert ist. Geben Sie diesmal jeweils die Indizes an auf die *Q.head* und *Q.tail* nach jeder Ausführung zeigen. Die Queue enthält anfangs 4 Elemente wie unten abgebildet. *Q.head* enthält somit vor der ersten Ausführung den Wert 4 und *Q.tail* enthält den Wert 8.



- ENQUEUE(Q,6)
- ENQUEUE(Q,7)
- ENQUEUE(Q,3)
- DEQUEUE(Q)
- ENQUEUE(Q,11)
- DEQUEUE(Q)
- DEQUEUE(Q)

# Lösung.

1	2	3	4	5	6	7	8	
			13	4	2	9	6	Q.head = $4$ , Q.tail = $1$
7			13	4	2	9	6	Q.head = 4, Q.tail = 2
7	3		13	4	2	9	6	Q.head = 4, Q.tail = 3
7	3			4	2	9	6	Q.head = 5, Q.tail = 3
7	3	11		4	2	9	6	Q.head = 5, Q.tail = 4
7	3	11			2	9	6	Q.head = 6, Q.tail = 4
7	3	11				9	6	Q.head = 7, Q.tail = 4

#### **P3** Linked Lists

- a) Vergleichen Sie doppelt verkettete Liste (doubly linked lists) mit einfach verketteten Listen (singly linked lists) indem Sie Vorteile dieser Datenstrukturen beschreiben. In welchen Fällen ist eine Variante gegenüber der anderen zu bevorzugen?
- b) Beschreiben Sie wie man einen Stack-Speicher basierend auf einer einfach verketteten Liste L aufbauen kann. Wie würden Sie die Funktionen PUSH(L,x) und POP(L) umsetzen?
- c) Formulieren Sie einen nicht-rekursiven Algorithmus, der eine einfach verkettete Liste mit n Elementen umkehrt. Die Methode soll keinen zusätzlichen Speicher wie extra Listen benutzen. Es sind nur temporäre Hilfsvariablen, die unabhängig von der Größe von n sind, erlaubt. Die Laufzeit der Funktion soll O(n) nicht überschreiten.

#### Lösung.

- a) Vorteile doppelt verkettete Liste:
  - Die Liste kann vorwärts und rückwärts durchlaufen werden.
  - Löschen eines Elements effizienter wenn man den Zeiger auf das zu löschende Element hat (O(1)), da man über dieses Element auch Vorgänger und Nachfolger anpassen kann für die Löschung. Bei einfach verketeten Listen bräuchte man zusätzlich den Zeiger vom Vorgänger-Element und müsste dafür im schlimmsten Fall die gesamte Liste durchlaufen (O(n)).

• (Liste kann einfach umgekehrt werden.)

Vorteile einfach verkettete Liste:

- Weniger Speicherbedarf pro Element, da nur Zeiger auf Nachfolger gespeichert wird. Bei dopppelt verketteten Listen jeweils Zeiger auf Vorgänger und Nachfolger.
  - (Zusatz: Es ist auch möglich doppelt verkettete Listen mit nur einem Zeiger umzusetzen, siehe: XOR linked list)
- Weniger Arbeitsschritte, da bei allen Befehlen wie Einfügen und Löschen nur ein Zeiger pro Element angepasst werden muss. Bei doppelt verketteten Listen jeweils die Zeiger auf Vorgänger und Nachfolger, was zu erhöhtem Arbeitsaufwand führt.
- b) Die PUSH(L,x) ist genau diesselbe Funktion wie LIST-INSERT(L,x) bei Listen. Die POP(L) Funktion setzt y = L.head und ruft LIST-DELETE(L,L.head) auf. Zum Schluss wird der Wert y zurückgegeben.
- c) Siehe Pseudocode

## Algorithm 1 Reverse List

```
1: procedure REVERSE(L)
       a = L.head
                                                                               ▶ a zeigt auf das erste Element der Liste
                                                                             ▶ b zeigt auf das zweite Element der Liste
3:
       b = a.next
                                                                                         ▶ Lösche Pointer von a nach b
       a.next = null
 4:
 5:
       while b \neq null do
 6:
          tmp = b.next
          b.next = a
                                                                                    ▶ Füge Pointer von b nach a hinzu
 7:
          a = b
 8:
                                                                   ▶ setze a und b jeweils ein ein Listenelement weiter
 9:
          b = tmp
       end while
10:
11:
       L.head = a
                                                                                       ▶ Setze neuen Listen Head auf a
12: end procedure
```

# **H1 Stacks und Queues**

- a) Zeigen Sie wie man eine Queue mit Hilfe von zwei Stacks realisieren kann. Geben Sie außerdem die Laufzeit der Methoden ENQUEUE und DEQUEUE an.
- b) Zeigen Sie wie man einen Stack mit Hilfe von zwei Queues realisieren kann. Geben Sie außerdem die Laufzeit der Methoden PUSH und POP an.

#### Lösung.

- a) ENQUEUE wird umgesetzt indem man ein Element auf den ersten Stack S1 pusht. (ENQUEUE: O(1)) DEQUEUE funktioniert wie folgt: Wir rufen POP(S2) auf. Falls Stack S2 jedoch leer ist, wird für jedes Element in Stack S1 POP(S1) aufgerufen und in Stack S2 gepusht. Zum Schluss wird mit POP(S2) das oberste Element zurückgegeben. (DEQUEUE O(n) im Worst Case)
- b) Solange Elemente in den Stack gepusht werden sollen, werden diese per ENQUEUE in Q1 abgelegt. (PUSH: O(1)) Sobald ein POP Befehl auf dem Stack ausgeführt werden soll, werden alle Elemente von Q1 bis auf das letzte per DEQUEUE aus Q1 entnommen und per ENQUEUE in Q2 abgelegt. Das letzte Element aus Q1 wird dann zurückgegeben und Q1 ist nun leer. Dies kann dann für jeden POP Befehl in beide Richtungen ausgeführt werden (Q1  $\rightarrow$  Q2 und Q2  $\rightarrow$  Q1). (POP: O(n))

## **H2** Linked Lists

Gegeben ist die folgende Sequenz von Elementen:  $S = \langle 24, 5, 7, 12, 2, 13 \rangle$ .

Illustrieren Sie wie die Elemente in einer doppelt verketteten Liste gespeichert werden, das umgesetzt ist als:

- 1. ein multidimensionales Array
- 2. ein eindimensionales Array

# Lösung.

Mögliche Lösungen:

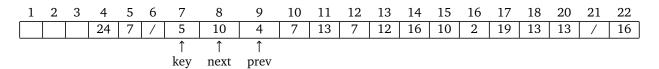
(Lösungen können variieren, da keine Sortierung in Linked List und in implementierten Arrays)

1. Variable L = 2 (Index auf das Head der Linked List zeigt)

Array index	1	2	3	4	5	6	7	
next	/	3	4	5	6	7	/	
key		24	5	7	12	2	13	_
prev		/	2	3	4	5	6	

In der Zeile key stehen die in der Liste gespeicherten Werte. Die Werte next bzw. prev speichern für jeden Wert die Pointer (die Position im Array) auf das folgende bzw. vorherige Element der Liste.

2. Variable L = 4 (Index auf das Head der Linked List zeigt)



Für jedes Element sind drei Zellen des Array nötig um neben dem Wert (key) die beiden Pointer (next und prev) zu speichern. Das zweite Element (5) steht an Position 7 im Array. An Position 8 steht die Array Position (10) des folgenden Elements (7) und an Position 9 die Position (4) der vorherigen Elements (24).