Einsendeaufgabe 5

Stefan Berger

1.

Der abstrakte Datentyp Stapel wird mit Hilfe einer einfachen verketteten Liste implementiert. Ein Element besteht aus zwei Daten: key, das eigentliche Datum (Singular von Data, kein Datum wie im Kalender) vom Element, und next, eine Referenz auf das nächste Element. Erstellen Sie Algorithmen für die 3 Operationen der Seite 67 im Skript: top, pop und push. Orientieren Sie sich an den Algorithmen der verketteten Liste im Abschnitt 5.2.

Algorithmus 1: top

Eingabe: Verkettete Liste L
Ausgabe: Datum des Elements am Ende der Liste L, falls dieses existiert, sonst NIL

```
TOP(L)
 1 if head[L] = NIL then
 2     return NIL
 3 e ← head[L]
 4 while next[e] ≠ NIL do
 5     e ← next[e]
 6 return key[e]
```

Algorithmus 2: pop

Eingabe: Verkettete Liste L
Ausgabe: Datum des Elements am Ende der Liste L, falls dieses existiert, sonst NIL

```
POP(L)
 1 if head[L] = NIL then
 2     return NIL
 3 e ← head[L]
 4 if next[head[L]] = NIL then
 5     head[L] ← NIL
 6     return e
 7 while next[e] ≠ NIL do
 8     e_top ← e
 9     e ← next[e]
10 next_top ← NIL
11 return key[e]
```

Algorithmus 3: push

Eingabe: Verkettete Liste L, abzulegendes Datum x
Ausgabe: keine

PUSH(L, x)

```
1 e ← neues Element
2 key[e] ← x
3 if head[L] = NIL then
4     head[L] ← e
5     return
6 e_top ← head[L]
7 while next[e_top] ≠ NIL do
8     e_top ← next[e_top]
9 next[e_top] ← e
```

3.

Geben Sie Zeitkomplexität der Methoden top, pop, push, toString und equals in asymptotischer Notation an, und begründen Sie Ihre Antwort.

Die grundlegende Anweisung ist in allen Fällen der Vergleich next = NIL. Er wird für jedes Element im Stapel einmal ausgeführt. Die Zeitkomplexität aller Methoden ist deshalb linear:

TOP, POP, PUSH, TOSTRING, EQUALS $\in \Theta(n)$

Die Methoden TOP, POP und PUSH könnten optimiert werden, indem zusätzlich ein Zeiger auf das oberste Element im Stack gespeichert und (für POP) eine doppelt verkettete Liste verwendet wird. Die Operationen hätten dann eine konstante Zeitkomplexität $\Theta(1)$.