Meilenstein 03

Aufgabe 3.4: Laufzeit

Die Aufzeichnung haben wir Folgenderweise aufgebaut: Zuerst der Originalquellcode, danach die, von uns vereinfachte Version des Codes und dann die daraus folgenden Laufzeiten und Berechnungen. (verallgemeinert)
Danach dann spezifisch auf die main bezogen.

Teil 1

```
bool breakLoop(int c, int a) {
    int x = 0;
    int y = a;
    int z = c;

    while(c > x) {                          //c>0  x ist immer 0
        for(int i = a * a; i < 0; i++) {    // schleife läuft nicht i < 0
            x += i;                         //x+= a*a, a++ solange a<0
        }
        c = c - z;                          //c=c-c läuft genau einmal bis
c>0, dann ist c=0, nur in den klammern
    }

    return x == y;                          // 0==a //ist a 0, returns 0 ==
a konstante komplexität O von 1
}
```

```
bool istaNull(int a) {
    return 0 == a;                      // 0==a //ist a 0, returns 0 == a konstante ko
mplexität O von 1
}
```

X ist immer 0, z ist immer c.
Die Schleife läuft nicht, da i < 0. Deswegen haben wir die for-Schleife herausgenommen.
Die while-Schleife läuft solange bis c > 0, da am Ende c -c gerechnet wird, ist diese Bedingung sofort nicht mehr erfüllt. Die Schleife läuft also nur einmal durch und hat keinen Einfluss auf den Rest der Operation. Daher haben wir auch das Stück rausgenommen, genau wie int c, da dieser dadurch in der Funktion keinen weiteren Sinn hat.
Somit bleibt nur noch x = 0 und y = a.
In der Operation steht nun, dass x == y, also 0 == a ist.
Der Sinn dieser Operation besteht also darin „istaNull“ auszuführen.

Daher konstante Komplexität O(1).

Teil 2

```
int doSomethingDifferent(int b) {
    int randomInt = 0;

    for(int i = b; i > 0; i = i/2) {         //solange i>0 wird i/2
ganzzahlige division, aufjden fall kommt 0 raus log b zur basis 2 von b,
gibt anzahl der schleifendurchläufe an
        randomInt *= i;                      //0 *= i
    }

    return randomInt;                        //gibt 0 zurück
}
```

```
int doSomethingDifferent(int b) {
    int randomInt = 0;

    for(int i = b; i > 0; i = i/2) {        //solange i>0 wird i/2  ganzzahlige division, a
ufjden fall kommt 0 raus log b zur basis 2 von b, gibt anzahl der schleifendurchläufe an
        randomInt *= i;                     //0 *= i
    }

    return randomInt;                       //gibt 0 zurück
}
```

randomInt ist 0.
Rechnung in der for-Schleife: Solange i>0 wird i/2 als ganzzahlige Division gerechnet. Durch die ganzzahlige Division kommt am Ende 0 heraus. Daher können wir mit dem log b zur Basis 2 rechnen, welcher auch den Durchlauf der Schleifen angibt.
randomInt *= i ergibt 0, da 0*0=0
Daher return 0, denn randomInt = 0

Die Laufzeit beträgt also O($\log_2 n$)

Teil 3

```
void doSomething(int firstVar, int secondVar) {

    int specialVar = doSomethingDifferent(secondVar); //specialvar = 0

    while(!breakLoop(firstVar,secondVar)) {
        specialVar = firstVar % secondVar;
        firstVar = secondVar;
        secondVar = specialVar;
    }                                                  // passiert nichts
}
```

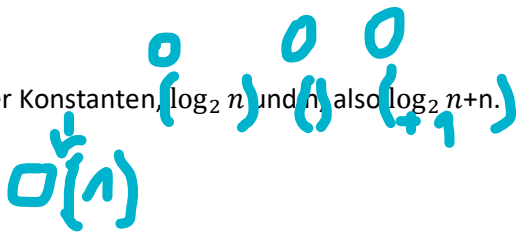
```
void doSomething(int a, int b) {

    int rest = doSomethingDifferent(b); //rest = 0

    while(0!=b) {
        rest = a % b;  // zwischen 0 und b-1
        a = b;               //mindestens 1 kleiner
        b = rest;           // läuft max b mal durch
    }                                          // passiert „nichts“
}
```

Wir haben die Variablen zur besseren Übersicht umbenannt. firstVar = a, secondVar = b, specialVar = rest.
Der rest wird durch doSomethingDifferent(b) = 0. Da in dieser Funktion aber gerechnet wird, unabhängig davon, dass doSomethingDiffrent(b) immer 0 ergibt, können wir rest nicht direkt 0 setzten. (Wichtig für Laufzeit)
breakLoop haben wir in istaNull umbenannt (Teil1), da der Sinn a == 0 ist. Zudem haben wir die erste Variable von istaNull gestrichen (Teil 1).
Somit while (!istaNull(b)), daraus können wir folgen, dass die Schleife läuft, solange b nicht 0 ist. Also while(0!=b).
Da wir Modulo rechnen, wird das erste Ergebnis zwischen 0 und b-1 liegen (wenn rest =b wäre glatter Teiler)
a = b ist dann mindestens um 1 kleiner.
Da b unser neuer rest ist und die schleife bis b==0 durchläuft, ist die Dauer des Durchlaufs der Schleife maximal b.

Somit beträgt die Laufzeit im Worstcase O(n).

Teil 4

Die gesamte Laufzeit besteht also aus einer Konstanten, $\log_2 n$ und n, also $\log_2 n$+n.
Nach der Absorption also O(n).



Teil 5

Wenn main ausgeführt wird:

Für Teil 1
rechnet nichts.
konstante

Für Teil 2
for(int i = 3; i > 0; i = i/2) mit ganzzahliger Division
3/2 = 1
½ = 0
2 Durchläufe

Für Teil 3
rest = 12%3
→ rest = 0
a = 3
b = 0
somit ist die Rechnung in der while-Schleife fertig ausgerechnet
1 Durchlauf