Aufgabe 2: Kontrollflussorientiertes Testen

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl `until` die Summe aller Zahlen bildet, die kleiner als `until` und Vielfache von 4 oder 6 sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$$\texttt{until} > 0 \Rightarrow \texttt{specialSums(until)} = \sum\{y \mid 0 < y < \texttt{until} \wedge (y\%4 = 0 \vee y\%6 = 0)\}$$
$$\texttt{until} \leq 0 \Rightarrow \texttt{specialSums(until)} = 0$$

wobei % den Modulo-Operator bezeichnet.

```
4   public static long specialSums(int until) {
5     long sum = 0; // 0
6     if (until > 0) { // 1
7       for (int i = 1; i <= until; i++) { // 2 // 5
8         if (i % 4 == 0 || i % 6 == 0) { // 3
9           sum += i; // 4
10        }
11      }
12    }
13    return sum; // 6
14  }
```

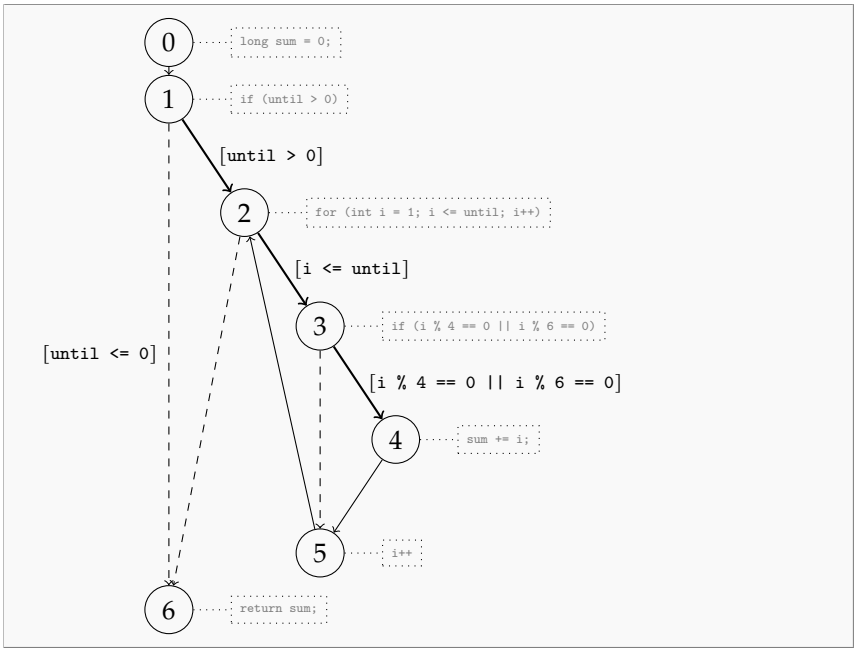
github: raw

Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife. Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

$$i \leq \texttt{until} \text{ in } i < \texttt{until}$$

geändert würde.

(a) Zeichnen Sie das zum Programm gehörige Ablaufdiagramm.



(b) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Der Fehler fällt nur dann auf, wenn `until` durch 4 oder 6 ohne Rest teilbar ist. `until = 0%4` oder `until = 0%6`. Wähle daher den Testfall $\{(1, 0)\}$. Alternativ kann für die Eingabe auch 2, 3, 5, 7, 9, 10, 11, 13, ... gewählt werden.

(c) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Betrachte den Testfall $\{(0, 0), (5, 4)\}$.

erste if-Bedingung Das erste Tupel mit until $= 0$ stellt sicher, dass die erste `if`-Bedingung `false` wird.

Bedingung der for-Schleife Für die zweite Eingabe until $= 5$ werden für i die Werte $1, 2, 3, 4, 5, 6$ angenommen. Wobei für $i = 6$ die Bedingung der for-Schleife `false` ist.

Innere if-Bedingung Für $i = 1, 2, 3, 5$ wird die innere if-Bedingung jeweils `false`, für $i = 4$ wird sie `true`.

(d) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungsüberdeckung und Zweigüberdeckung ihres Testfalls.

Anweisungsüberdeckung Wähle $\{(4, 0)\}$. Durch die fehlerhafte Bedingung in der for-Schleife wird der Wert $i = 4$ akzeptiert. Da alle Anweisungen ausgeführt werden, wird eine Anweisungsüberdeckung mit 100% erreicht.

Verzweigungsüberdeckung Da die erste Verzweigung nur zur Hälfte überdeckt wird und die anderen beiden vollständig, gilt für die Verzweigungsüberdeckung:
$\frac{1+2+2}{2+2+2} = \frac{5}{6}$

(e) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an, der nicht erreichbar ist. Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar (I, O) die Eingabe I und die zu dieser erwartete Ausgabe O darstellt.

Ein gültiger Pfad im Kontrollflussgraphen wäre ⓪ - ① - ② - ③ - ④ - ⑤ - ② - ⑥. Der Übergang von ③ auf ④ ist hier aber nicht möglich, da beim ersten Durchlaufen der for-Schleife (②) i immer 1 ist und 1 weder durch 4 noch durch 6 teilbar ist. Somit kann die Bedingung des inneren ifs (③) beim ersten Durchlauf nie *wahr* sein, womit immer der Übergang ③ - ⑤ zu Beginn genommen werden muss. Alle Pfade, die zu Beginn ③ - ④ enthalten, sind somit nicht überdeckbar.