

Ackermann-Funktion $ack: \mathbb{N}^2 \rightarrow \mathbb{N}$

$$ack(x, y) = \begin{cases} y + 1 & \text{wenn } x = 0 \\ ack(x - 1, 1) & \text{wenn } x > 0 \text{ und } y = 0 \\ ack(x - 1, ack(x, y - 1)) & \text{sonst (also } x > 0 \text{ und } y > 0) \end{cases}$$

Ackermannfunktion $a: \mathbb{N}^2 \rightarrow \mathbb{N}$

$$a(x, y) = \begin{cases} 1 & \text{wenn } x = 0 \text{ oder } y = 0 \\ 3y + 1 & \text{wenn } x = 1 \text{ (und } y > 0) \\ \underbrace{a(x - 1, a(x - 1, \dots a(x - 1, y) \dots))}_{y \text{ mal 'a(x-1, '}} & \text{sonst (also } x > 0 \text{ und } y > 0) \end{cases}$$

Beispiel: $a(2,3) = a(1, a(1, a(1,3))) = a(1, a(1,10)) = a(1,31) = 94$

Bemerkung: ack und a sind total und intuitiv berechenbar.

Lemma: Für jedes LOOP-Programm P , das nur die Variablen x_0, \dots, x_m benutzt, gibt es ein $k \in \mathbb{N}$ mit $f_P(n) < a(k, n)$ für alle $n \geq k$. Dabei ist $f_P(n)$ die maximale Summe der Endwerte in x_0, \dots, x_m für Berechnungen von P mit Anfangswerten, deren Summe höchstens n ist.

Satz: ack und a sind nicht LOOP-berechenbar (also auch nicht primitiv-rekursiv).

INPUT(x, y); (*Eingabe der Argumente $x = x_0, y = y_0$ *)

INIT(stack): (*Erzeuge leeren Stapel: $stack = []$ *)

PUSH(x, stack); (* $stack = [x_0]$ *)

PUSH(y, stack); (* $stack = [y_0, x_0]$ *)

WHILE size(stack) \neq 1 DO

*(*stack = $[x_k, \dots, x_1]$ entspricht $a(x_1, \dots, a(x_{k-1}, x_k) \dots)$ *)*

y := POP(stack); (* $y = x_k$ *)

x := POP(stack); (* $x = x_{k-1}$ *)

IF (x = 0) OR (y = 0) THEN PUSH(1, stack)

ELSE IF x=1 THEN PUSH(3*y+1, stack)

(stack = $[a(x_{k-1}, x_k), x_{k-2}, \dots, x_1]$ *)*

ELSE LOOP y DO

PUSH(x-1, stack)

END;

PUSH(y, stack)

(stack = $[y, x-1, \dots, x-1, x_{k-2}, \dots, x_1]$ *)*

entspricht $a(x_1, \dots, a(x_{k-2}, a(x-1, a(x-1, \dots, a(x-1, y) \dots)))$

*= $a(x_1, \dots, a(x_{k-2}, a(x_{k-1}, x_k) \dots)$ *)*

END

END

END;

result := POP(stack);

OUTPUT(result)