Blatt 15

Aufgabe 1:

a) uniform: Die 1. Schleife wird x mal durchlaufen.
In x steht danach $x^{(2^x)}$.
Die zweite Schleife wird folglich $x^{(2^x)}$ mal durchlaufen. In x steht danach $(x^{(2^x)})^{(2^{x^{(2^x)}})}$
Insgesamt wird also $x^{(2^x)} + x$ mal multipliziert.

$\Rightarrow O(x^{(2^x)})$

Hinweis zur O-Notation:

$O(g(n)) = \{f(n) \mid \exists c, n_0 > 0 \quad 0 \le f(n) \le c \cdot g(n) \quad \forall n \ge n_0\}$

$O(g(n))$ gibt also eine obere Schranke an.

Logarithmisch: Beim logarithmischen Kostenmaß wird zusätzlich noch die Länge der Variablen berücksichtigt und zwar als $\log(\text{Zahl})$, da diese als binär codiert angenommen wird.

$\Rightarrow O(x^{(2^x)} \cdot \log([x^{(2^x)}]^{2^{x^{(2^x)}}}))$

b) Das Programm mache n Schritte.
Die Variable (Zahl) habe am Anfg. b Bits.
Mit jeder Addition kann max. je ein Bit hinzukommen. $\Rightarrow$ nach n Schritten ist meine Maximallänge $b+n$.
Das logarithmische Kostenmaß ist somit
$O(n \cdot (n+b)) = O(n^2)$ (Da wir hier sowieso schon Bits betrachtet haben, brauchen wir den Logarithmus nicht...)

uniform wäre $O(n)$.

$\Rightarrow f(n) = n^2$

Aufgabe 2:

Wie sieht so ein LOOP-Programm im schlimmsten Fall aus?

```
LOOP x DO
      x := x+c;
    :   :
    LOOP x DO
            x := x+c;
          :   :
    END
    :
END
```

hier worst case: $x \to x \cdot k$

das x mal

k wird beliebig groß genug gewählt, um sämtliche Additionen abzudecken $\left[(x := x+c; x := x+c; \ldots x := x+c;) \in O(x \cdot k) \right]$

Am Ende ist unsere Zahl also $\in O(x \cdot k^x)$.
Uniform ergeben sich also $O(x \cdot k^x)$-Schritte.

Nehmen wir an, unsere Eingabe sei n.

Wir schätzen die Länge der Zahl großzügig mit 2^n nach oben ab. Da wir nur Additionen haben, wird die Zahl in keinem Berechnungsschritt länger sein.

Pro While-Schritt geht die TM schlimmstenfalls das ganze Band durch:

$$\Rightarrow \bigcup_k \text{DTIME}\left(\underbrace{\left[2^n \cdot k^{(2^n)}\right]}_{\#\,\text{Schritte}} \cdot \underbrace{\left[2^n \cdot k^{(2^n)}\right]}_{\#\,\text{pro Schritt}} \right)$$

$$= \bigcup_k \text{DTIME}\left(\left[2^n \cdot k^{(2^n)}\right]^2 \right)$$

Aufgabe 3:

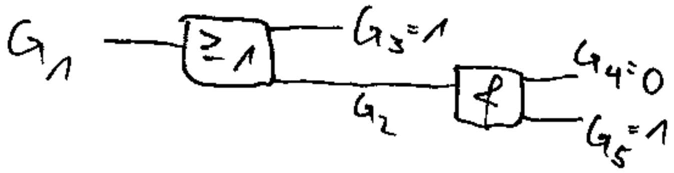
Was sollen wir tun?

Das Problem in eine Grammatik überführen!

	aus	mache
	$g_i = 0$	/
	$g_i = 1$	$G_i \Rightarrow a$
$\boxed{\geq 1}$	$g_i = (+, j, k)$	$G_i \Rightarrow G_j \mid G_k$
$\boxed{\&}$	$g_i = (\cdot, j, k)$	$G_i \Rightarrow G_j\, G_k$

$$S \Rightarrow G_1$$

Beispiel:



$G_1 \Rightarrow G_3 \mid G_2$

$G_2 \Rightarrow G_4\, G_5$

$G_3 \Rightarrow a$

$G_5 \Rightarrow a$

Aufgabe 4:

a) Rate Dir nichtdeterministisch eine Rundreise und prüfe dann, ob sie erschwinglich ist.

b) Ritter $\rightarrow$ Städte

(R_i, R_j) verfeindet $\Rightarrow$ Kosten zwischen R_i und $R_j = n+1 = c_{R_i, R_j}$

(R_i, R_j) nicht verf. $\Rightarrow$ Kosten $= 1 = c_{R_i, R_j}$

Etat $= n$

Damit ist das König Arthus-Problem auf das des Handlungsreisenden reduziert und somit auch $\in NP$!