Aufgabe 5

Sei x unser Eingabewort.

Ausgang

- Eingabe x liegt auf Band 1.
- Kopf $k1$ für Band 1 liegt auf dem ersten Zeichen von x, Kopf $k2$ für Band 2 liegt über dem letzten Zeichen von x (Kann man auch in $\mathcal{O}(|x|)$ einstellen).

Funktionsweise

1. Lese Zeichen $(x)_i$ von $k1$ und schreibe es auf $k2$.
2. Schiebe $k1$ nach Rechts, $k2$ nach links.
3. Gehe zu 1), falls $k1$ nicht auf B liegt.
4. Schiebe $k1$ wieder einmal nach Links und $k2$ solange nach Rechts, bis es auf B liegt. Dann wieder einmal nach Links ($k1$ und $k2$ liegen also übereinander).
5. Vergleiche Zeichen unter $k1$ und $k2$. Verwerfe, wenn Zeichen ungleich. Sonst schiebe beide Köpfe nach links und wiederhole solange, bis beide auf B liegen. Dann akzeptiere.

Speicherbedarf

Der Speicherbedarf liegt offentsichtlich hier bei $2 \cdot |x|$. Einmal zur Speicherung des Eingabewortes ww^{-1} auf Band 1 und einmal um das Invertiere Eingabewort, also x^{-1} auf Band 2 zu speichern.

Laufzeit

Die Ausgangssituation der Köpfe kann man, wenn nicht schon eingestellt, manuell in $\mathcal{O}(|x|)$ einstellen (abhängig, wie die Köpfe eben am Anfang stehen).
Das Kopieren das Zeichens unter $k1$ auf $k2$ geht in konstanter Zeit.
Dann wird $k1$ und $k2$ insgesamt $|x|$ Mal verschoben, also liegt unser Vorgang, um x invertiert auf Band 2 zu schreiben in $\mathcal{O}(|x|)$.
Unsere Überprüfung, ob das Eingabewort wirklich ein Palidrom wie in der Aufgabenstellung definiert ist geht auch in $\mathcal{O}(|x|)$, da wir das Wort nur von Rechts nach Links durchgehen.
Unsere gesamte Laufzeit liegt somit auch in $\mathcal{O}(|x|)$.

Aufgabe 6

$$\sqrt{2^n} = 2^{\frac{n}{2}} \Rightarrow \text{Exponentiell in } \frac{n}{2}$$

Idee: Ich lade die Zahl $2^{\frac{n}{2}}$ und zähle dann bis 0 herunter:

```
// 1. Teil: Laden die Zahl 2^(n/2)
1: CLOAD 1
2: STORE 2
3: LOAD 1
4: IF c(0) = 0 THEN GOTO 11
5: CDIV 4
6: STORE 1
7: LOAD 2
8: CMUL 2
9: STORE 2
10: GOTO 3
// 2. Teil: Bis 0 runterzaehlen
11: LOAD 2
12: IF c(0) = 0 THEN GOTO 15
13: CSUB 1
14: GOTO 12
15: END
```

Uniformes Kostenmaß

Obda: $m = 2^n$. Für $2^n < m > 2^n - 1$ führt es zum selben Ergebnis wie $m = 2^n$.
Nach dem i-ten Durchlauf der Schleife im 1. Teil ist $m = 2^{n-2i}$ und $c(2) = 2^i 4$.
Wenn Zeile 11 erreicht wird, ist $c(2) = 2^{\frac{n}{2}}$.
Die Laufzeit von Teil 1 ist offensichtlich linear in n, da die Schleife $\frac{n}{2}$ mal durchlaufen wird.
Die Laufzeit von Teil 2 hingegen liegt in $2^{\frac{n}{2}}$, da die Schleife auch $2^{\frac{n}{2}}$ mal durchlaufen wird.
Also liegt die Gesamtlaufzeit in $2^{\frac{n}{2}}$.

Logarithmisches Kostenmaß

Die Laufzeit des 1. Teils liegt in $\frac{n}{2} * n$, da bei jedem der $\frac{n}{2}$ Schleifendurchläufe die Zahl zwei mal nach rechts geshiftet werden muss (lineare Laufzeit).
Die Laufzeit des 2. Teils liegt in $2^{\frac{n}{2}} * n$, da bei jedem der $2^{\frac{n}{2}}$ Schleifendurchläufe um die Zahl 1 subtrahiert wird (lineare Laufzeit).

Aufgabe 7

a) R ist offensichtlich eine Teilmenge von R'^* mit $R' := \{a, b, c, *, +, (,), \emptyset\}$.
Zeigen wir also, dass R'^* abzählbar ist, so muss R auch abzählbar sein:
Sei $h : R' \to 0, 1, 2, 3, 4, 5, 6, 7$ (bijektiv) mit:

$$\begin{array}{cccc} h(a) = 0 & h(b) = 1 & h(c) = 2 & h(*) = 3 \\ h(+) = 4 & h(() = 5 & h()) = 6 & h(\emptyset) = 7 \end{array}$$

Dann gibt es eine surjektive Funktion $f : R'^* \to \mathbb{N}$. Sei hier $r \in R'^*$ mit $n := |r|$ und $r = r_1...r_n$ mit $\forall_{i \in 1...n} : r_i \in R'$:

$$f(r) = \sum_{i=0}^{n} h(r_i) \cdot 8^i$$

Damit ist also R'^* abzählbar, wodurch auch R abzählbar sein muss.

b) Beweis, dass A nicht regulär ist durch Diagonalisierung:
Angenommen es gibt einen regulären Ausdruck r_i, sodass $L(r_i) = A$ gilt. Dann gibt es für das Wort a^i zwei Fälle:

1) a^i ist in $A \Rightarrow a^i$ ist in $L(r_i) \Rightarrow a^i$ ist nicht in A ↯Widerspruch
2) a^i ist nicht in $A \Rightarrow a^i$ ist nicht in $L(r_i) \Rightarrow a^i$ ist in A ↯Widerspruch

Also ist A nicht regulär.

c) Aus Fosap wissen wir, dass man einen regulären Ausdruck r in einen NFA umwandeln kann und damit für ein Wort x entscheiden kann, ob $x \in L(r)$ gilt. Also gibt es auch eine TM NFA_{conv}, welche einen für einen gegebenen regulären Ausdruck r eine Gödelnummer $< R >$ berechnen kann, sodass $< R >$ unseren regulären Ausdruck als NFA simuliert. Für unseren Algorithmus benötigen wir also noch eine Universalturingmaschine NFA_{sim}, welche eine solche Gödelnummer mit einem Wort $< R > x$ entgegennimmt und genau dann akzeptiert, wenn $x \in L(r)$ und sonst verwirft.
Unser Algorithmus für eine TM T, welche A_i entscheidet, sieht also wie folgt aus:

1. Übergebe Eingabe a^i an eine TM, welche die Anzahl an a's zählt und diese dann ausgibt. Diese TM verwirft, wenn ein Fehler auftritt (z.B. anderes Zeichen als a), aber die gesamte TM T akzeptiert dann.
2. Übergebe i an eine wie in der Teilaufgabe beschriebene TM, die wir M nennen. M berechnet uns also aus i den regulären Ausdruck r_i.
3. Den regulären Ausdruck geben wir nun weiter an NFA_{conv}, welche uns also die Gödelnummer $< R_i >$ einer TM zurückgibt, welche einen NFA über r_i simuliert.
4. Anschließend übergeben wir NFA_{sim} die Gödelnummer und das Eingabewort: $< R_i > a^i$
5. Im letzten Schritt invertieren wir nun das Ergebnis: Wenn also $a^i \in L(r_i)$ gilt, sollen wir verwerfen, sonst akzeptieren

Korrektheit

Nehmen wir an unsere Eingabe besteht nur aus der Form a^i, sonst würden wir eh akzeptieren, wodurch die Eingabe in A wäre.

$$
\begin{aligned}
a^i \in A &\Rightarrow \text{Schritt 1. gibt uns } i \\
&\Rightarrow \text{M berechnet uns } r_i \\
&\Rightarrow NFA_{conv} \text{ berechnet } < R_i > \\
&\Rightarrow NFA_{sim} \text{ verwirft} \\
&\Rightarrow T \text{ akzeptiert}
\end{aligned}
$$

$$
\begin{aligned}
a^i \notin A &\Rightarrow \text{Schritt 1. gibt uns } i \\
&\Rightarrow \text{M berechnet uns } r_i \\
&\Rightarrow NFA_{conv} \text{ berechnet } < R_i > \\
&\Rightarrow NFA_{sim} \text{ akzeptiert} \\
&\Rightarrow T \text{ verwirft}
\end{aligned}
$$

Damit entscheidet also T die Sprache A.