Exercise 1: Knapsack with Integer Values

Given \( n \) items \( 1, \ldots, n \) with weights \( w_i \in \mathbb{R} \) and values \( v_i \in \mathbb{N} \) and a bag capacity \( W \), we want to find a subset \( S \subseteq \{1, \ldots, n\} \) that maximizes \( \sum_{i \in S} v_i \) under the restriction \( \sum_{i \in S} w_i \leq W \).

Give an efficient\(^1\) algorithm for this problem that uses the principle of dynamic programming.

**Hint:** Define a function that computes for a \( k \in \{1, \ldots, n\} \) and an integer \( V \) the minimum weight of a collection of items from \( \{1, \ldots, k\} \) that has value \( V \).

Exercise 2: Dynamic Programming

Consider the following functions \( f_i : \mathbb{N} \to \mathbb{N} \):

\[
\begin{align*}
f_1(n) &= n - 1 \\
f_2(n) &= \begin{cases} 
\frac{n}{2} & \text{if } 2 \text{ divides } n \\
n & \text{else}
\end{cases} \\
f_3(n) &= \begin{cases} 
\frac{n}{3} & \text{if } 3 \text{ divides } n \\
n & \text{else}
\end{cases}
\end{align*}
\]

"m divides n" means there is a \( k \in \mathbb{N} \) with \( k \cdot m = n \).

For a given \( n \geq 1 \), we want to find the minimal number of applications of the functions \( f_1, f_2, f_3 \) needed to reach \( 1 \). Formally: Find the minimal \( k \) for which there are \( i_1, \ldots, i_k \in \{1, 2, 3\} \) with \( f_{i_1}(f_{i_2}(\ldots(f_{i_k}(n))\ldots)) = 1 \).

Devise an algorithm in pseudocode to solve the problem and analyze the runtime.

Exercise 3: Amortized Analysis

Suppose a sequence of \( n \) operations are performed on an (unknown) data structure in which the \( i \)-th operation costs \( i \) if \( i \) is an exact power of 2, and 1 otherwise.

<table>
<thead>
<tr>
<th>Operation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>...</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Cost</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>...</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

Tabelle 1: Operations and their actual costs

Use the potential function method to show that each operation has constant amortized cost.

**Hint:** The number of consecutive operations that are not an exact power of 2 and are performed immediately before operation \( (i + 1) \) is \( i - 2^\ell(i) \) where \( \ell(i) := \lfloor \log_2 i \rfloor \).