Technische Universität Berlin **SoSe 2024**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Dorothea Kiefer-Hoeft
Prof. Dr. Max Klimm
Dr. Tobias Hofmann, Martin Knaack, Marcel Wack

5. Programmieraufgabe Computerorientierte Mathematik II

Abgabe: 31.05.2024 über den Comajudge bis 17:00 Uhr

Huffman Code

Ziel dieser Aufgabe wird es sein, einen `String` mithilfe eines Huffman Codes zu codieren und komprimieren. Dafür definieren Sie zunächst ein Binärbaum durch ein `mutable` oder `non-mutable` `struct Node` mit den `fields`:

1. `value::Union{Char,Nothing}`: Der Wert des Knoten
2. `freq::Int`: Die Häufigkeit der Auftritte des Symbols in dem String.
3. `left::Union{Node,Nothing}`: Das linke Kind.
4. `right::Union{Node,Nothing}`: Das linke Kind.

Für den Rest der Aufgabe verwenden wir nur den kanonischen Konstruktor für `Node`, Sie müssen nicht aber können weitere Konstruktoren definieren.

> **Hinweis:** Der Typ `Char` in `Julia` ist ein abstrakter Typ, der alle Zeichen repräsentiert. Ein `Char` wird in `Julia` durch einfache Anführungszeichen dargestellt, z.B. `'a'`. Ein `String` ist eine Sequenz von `Char`s und wird in `Julia` durch doppelte Anführungszeichen dargestellt, z.B. `"abc"`.

Beginnen Sie mit der Implementierung einer Funktion `getFrequencies`, die für einen gegebenen Text ein Dictionary erstellt, das als Key-Value-Paare einen Buchstaben und seine Häufigkeit im Text speichert.

```
julia> Example = "Wenn der Physiker nicht weiter weiß, gründet er ein Arbeitskreis";

julia> freqs = getFrequencies(Example)
Dict{Char, Int64} with 21 entries:
  'n' => 5
  'w' => 2
  'd' => 2
  'e' => 11
    ..
```

Huffman Tree

Um den Baum zu erstellen, verwenden wir den Ansatz mit zwei Listen. Wir schreiben also zunächst eine Funktion, die die beiden kleinsten Knoten zweier aufsteigender Listen ausgibt und diese aus der entsprechenden Liste löscht. Wir verwenden die Ausgabe der Funktion, um einen

unbenannten Knoten zu erzeugen und ihn der zweiten Liste hinzuzufügen. Konkret wird zunächst eine Funktion `findLowestTwo(q1::Vector{Node}, q2::Vector{Node})::Tuple{Node, Node}` implementiert. Die Funktion, kann davon ausgehen, dass in den Listen zwei Elemente existieren. Mit kleinstem Element ist das kleinste Element der kanonischen Ordnung entlang des `field: freq` gemeint.
Schreiben Sie dann eine Funktion `huffman_tree(freqs::Dict{Char,Int})::Node`, die den Huffman-Baum für die gegebenen Frequenzen erzeugt. Die Funktion sollte die Wurzel des Baumes zurückgeben. Grob gesagt sollte sie so funktionieren, dass für jeden `Char` ein Knoten ohne Kinder erzeugt und in einer aufsteigenden (ersten) Liste gespeichert wird. Die zweite Liste ist zunächst leer. Dann werden die beiden kleinsten Knoten aus beiden Listen zusammengefügt und die zweite Liste hinzugefügt, bis nur noch ein Knoten übrig ist.

```
julia> x = [Node(nothing,1,nothing,nothing), Node(nothing,4,nothing,nothing)];

julia> y = [Node(nothing,2,nothing,nothing), Node(nothing,3,nothing,nothing)];

julia> findLowestTwo(x, y)
(Node(nothing, 1, nothing, nothing), Node(nothing, 2, nothing, nothing))

julia> x
1-element Vector{Node}:
 Node(nothing, 4, nothing, nothing)
```

Encode - Decode

Dann schreiben Sie zwei Funktionen, um den Text zu kodieren und zu dekodieren, nämlich eine Funktion `Encode(text::String, code::Dict{Char, String})` und eine Funktion `Decode(encoded::String, tree::Node)`. Die Funktion `Encode` soll den Text mit dem im Dictionary angegebenen Code kodieren, indem sie jedes `Char` durch seinen Code ersetzt. Die Funktion `Decode` sollte den Text mit Hilfe des Baums decodieren. Beide Funktionen sollten ihre Ausgabe als `String` zurückgeben.

```
julia> encoded = Encode(Example, huffcode)
"01111110101110111000011110100100011111111101111100111
11111011110110011101001000111011001111111011111001100
00111111010110011010010001111110101101111111111001111
11111001111111001011111111101110011110100110001001000 1
01101110001111111111101001111101011001101111001110010
1011011110"

# Decode

julia> decoded = Decode(encoded, hufftree)
"Wenn der Physiker nicht weiter weiß, gründet er ein Arbeitskreis"
```

Zusammengefasst

Implementieren Sie die folgenden Funktionen und `structs` in `Julia`:

1. `struct Node`: Ein Binärbaum mit den `fields` `value`, `freq`, `left`, `right`.
2. `getFrequencies(text::String)::Dict{Char,Int}`: Erstellt ein Dictionary, das die Häufigkeit der Buchstaben im Text speichert.
3. `findLowestTwo(q1::Vector{Node}, q2::Vector{Node})::Tuple{Node, Node}`: Gibt die beiden Knoten mit der kleinsten Häufigkeit aus den beiden Listen zurück.
4. `huffman_tree(freqs::Dict{Char,Int})::Node`: Erzeugt den Huffman-Baum für die gegebenen Frequenzen.
5. `Encode(text::String, code::Dict{Char, String})::String`: Kodiert den Text mithilfe des Codes.
6. `Decode(encoded::String, tree::Node)::String`: Dekodiert den Text mithilfe des Baumes.