Dr. Frank Schöpfer

Einführung in Matlab

Übung 7

Aufgabe 1:

(a) **Erzeugen der elementaren Codewörter für einen gegebenen Huffman-Baum**:
Zum Kodieren eines Textes müssen wir wissen, welche Bitfolge ("elementares Codewort") zu Zeichen z_i gehört. Dazu müssen wir **alle möglichen Zweige des Baumes von der Wurzel bis zum Ende durchlaufen**. Die dabei **auftretenden Bitfolgen werden dann jeweils dem Zeichen zugeordnet, welches sich am jeweiligen Ende befindet**. Die folgende Funktion `codewort` speichert **alle elementaren Codewörter im Structure Array** `cw` mit Feldnamen `Zeichen` und `Codewort`.

Hier die Version mit Huffman-Baum als Structure t.

```
function cw=codewort(t)
cw=struct('Zeichen',[],'Codewort',[]);
i=1; % Zeichenindex
knoten=t; % Wurzel
folge_zweig(knoten.a,0);
folge_zweig(knoten.b,1);
% eingenestete Funktion teilt cw,i mit Hauptfunktion
    function folge_zweig(knoten,c)
        ...
        end
    end
end
```

Ergänzen Sie die eingenestete Funktion `folge_zweig(knoten,c)` **anhand folgender Überlegung**: Ist `knoten` eine Structure, so folgen wir wiederum beiden Zweigen `knoten.a` und `knoten.b`, wobei wir das bisherige Codewort `c` um 0 bzw. 1 ergänzen. Ansonsten befinden wir uns an einem Blatt/Zeichen des Baumes, und in diesem Fall soll in `cw(i).Zeichen` das aktuelle Zeichen und in `cw(i).Codewort` das aktuelle Codewort gespeichert werden, und wir müssen den Zeichenindex `i` um 1 erhöhen.

Testen Sie Ihr Programm an dem Beispiel aus der Vorlesung:

```
>> cw=codewort(t)
cw =
  1x4 struct array with fields:
    Zeichen
    Codewort
>> cw(3)
ans =
  struct with fields:

     Zeichen: 'K'
    Codewort: [1 1 0]
```

Zusatz: Lassen Sie sich **alle Zeichen mit ihrem elementaren Codewort tabellarisch in eine Textdatei** schreiben.

(b) **Kodierung**: Schreiben Sie eine Funktion `function c=kodiere(s,cw)`, welche aus dem String `s` mittels der elementaren Codewörter in `cw` den Code `c` als Vektor aus Nullen und Einsen macht (der Einfachheit halber z.B. mit zwei geeigneten for-Schleifen).
Test für das Beispiel aus der Vorlesung:

```
>> c=kodiere('KAFFEE',cw)
c =
     1     1     0     1     1     1     0     0     1     0     1     0
```

(c) **Bestimmen der relativen Häufigkeiten**:
Schreiben Sie eine Funktion `function w=haeufigkeit(s,z)` welche die relativen Häufigkeiten `w` der Zeichen `z` in einem String `s` bestimmt (auch hier der Einfachheit halber z.B. mit zwei geeigneten for-Schleifen).
Test:

```
>> w=haeufigkeit('KAFFEEAFFE','KAEF')
w =
    0.1000    0.2000    0.3000    0.4000
```

(d) **Anwendung:**

- **Wählen Sie** sich nun einen **längeren Text** aus und**bestimmen Sie die relativen Häufigkeiten** `w` **der von Ihnen vorgegebenen Zeichen**, z.B. alle Klein-, Großbuchstaben, Zahlen, Sonderzeichen wie ".?!" etc. mit `z=[char(32:127),'äöüÄÖÜß']`.
  Bemerkung: Sie können einen **Text aus einer Textdatei in einen String einlesen** mit `fscanf` durch

```
>> fileID = fopen('mytext.txt','r');  % zum lesen (read) öffnen
>> s=fscanf(fileID,'%c');  % mit dem Formatzeichen c (statt s) werden
                           % Strings auch mit Leerzeichen eingelesen
>> fclose(fileID);
```

- **Erstellen** Sie den **zugehörigen Huffman-Baum** `t` **sowie die Codewörter** `cw`.
- **Kodieren Sie damit den Text** und **vergleichen** Sie die benötigte Anzahl an **Bits zur Speicherung des Original-Textes und des Huffman-kodierten Textes** (nehmen Sie z.B. an, dass jedes Zeichen des Original-Textes 1 Byte=8 Bits benötigt.)