Vigenère-Kodierung: Automatische Dechiffrierung

Ziel der Arbeit ist eine Funktion `dechiffrieren(geheimtext)`, die einen mit dem Vigenere-Verfahren verschlüsselten Text entschlüsselt.

Einzelarbeit (10 Minuten)
Top-Down-Verfahren, Autokorrelationstest

Stammgruppenphase (30 Minuten)
Formuliert Teilaufgaben für das Entschlüsseln, die von je einer Funktion erledigt werden können. Dabei gilt: Die Funktionen sollten so „klein" sein, dass man für den Algorithmus zumindest eine Idee hat. Eine Ausformulierung ist hier aber noch nicht sinnvoll.

Legt für jede der Hilfsfunktionen fest

• Name der Funktion	`maximum`
• Zweck der Funktion	Findet das größte Element in einem Array von Zahlen
• Parameter (Name, Typ und Beispielwerte)	Array von Zahlen, z.B. `[1, 2, 4, 3]`
• Rückgabewerte (Typ und Beispielwerte)	Größtes Element (zahl), z.B. `4`

Notiert, in welcher Reihenfolge die Funktionen aufgerufen werden müssen, damit am Ende der entschlüsselte Text entsteht.

Plenum (20 Minuten)
Vorstellung der Ergebnisse, Einigung auf eine Vorgehensweise

Stammgruppen (10 Minuten)
Zuordnung der Experten zu Teilfunktionen

Expertengruppen (70 Minuten)
Entwicklung der Funktion(en)
→ Entwurf eines Algorithmus'
→ Implementation des Algorithmus'
→ Testen

Stammgruppen (20 Minuten)
Integration der Teilfunktionen zu einem funktionsfähigen Programm
Dechiffrieren der Beispieltexte

Vigenère-Kodierung: Automatische Dechiffrierung

Information: Top-Down und Bottom-Up

Für die Softwarentwicklung gibt es zwei im Grundsatz verschiedene Strategien. Die Top-Down-Methode eignet sich in der Regel dann, wenn eine spezifische Aufgabe zu lösen ist. Dafür zerlegt man sie Schritt für Schritt in Teilaufgaben, bis jede der Teilaufgaben direkt lösbar ist. Das Verfahren wird daher auch „schrittweise Verfeinerung“ genannt.

Im Gegensatz dazu geht der Bottom-Up-Ansatz davon aus, kleine (Standard-)Bausteine zu verwenden, die je nach Bedarf zu einem großen Programm zusammengesetzt werden. Oft werden solche Bausteine von Programmiersprachen bereits in Bibliotheken zur Verfügung gestellt. Man könnte diesen Ansatz auch den „LEGO-Approach“ nennen.

Dieser Ansatz eignet sich immer dann, wenn die Aufgaben allgemeiner formuliert oder sehr komplex sind, z.B. beim Entwurf eines Betriebssystems.

Information: Autokorrelationstest

Das Finden der Brückenlängen mit dem Kasiski-Verfahren ist recht schwierig zu programmieren. Leichter umsetzbar ist der so genannte Autokorrelationstest. Dabei wird der zu untersuchende Text in jedem Schritt um eine Stelle verschoben und die in Original und verschobenem Text übereinstimmenden Zeichen gezählt. In der vorgestellten Variante werden dabei einzelne gleiche Zeichen ignoriert und nur Gruppen gezählt.

Durch das Quadrieren der Brückenbreite werden Brücken aus vielen Zeichen deutlich höher gewichtet als schmale Brücken. Eine einzelne Brücke der Breite 10 zählt damit genauso viel wie 25 Brücken der Breite 2.

Quelltext

```
def autokorrelation(text):
    praefix = ""
    text = text.upper()
    laenge = len(text)
    bruecken = [0]

    for i in range(1, len(text)):
        praefix = praefix + "."
        text2 = + text[:(laenge-i)]
        breite = zaehler = 0
        for j in range(laenge):
            if text[j] == text2[j]:
                breite = breite + 1
            else:
                if breite > 1:
                    zaehler = zaehler + breite * breite
                breite = 0
        bruecken.append(zaehler)
        print(text+“\n“+text2)
        print("Verschiebung %d: %d Zeichen"%(i, zaehler))

    return bruecken
```

Beispielausgabe für den Text `"bcabcdbce"`

```
BCABCDBCE
.BCABCDBC
Verschiebung 1: 0 Brücken
BCABCDBCE
..BCABCDB
Verschiebung 2: 0 Brücken
BCABCDBCE
...BCABCD
Verschiebung 3: 8 Brücken
BCABCDBCE
....BCABC
Verschiebung 4: 0 Brücken
BCABCDBCE
.....BCAB
Verschiebung 5: 0 Brücken
BCABCDBCE
......BCA
Verschiebung 6: 4 Brücken
BCABCDBCE
.......BC
Verschiebung 7: 0 Brücken
BCABCDBCE
........B
Verschiebung 8: 0 Brücken
```

Ergebnis der Funktion

```
[0,0,0,8,0,0,4,0,0]
```

Vigenère-Kodierung: Automatische Dechiffrierung

Vorschlag für ein Vorgehen

- fett gedruckt sind direkt implementierbaren Funktionen, nicht fett die darauf zurückgreifenden Funktionen.
- „Haken“: Die Suche nach den größten Brücken erfordert die Hilfsfunktion auf dem Gruppenarbeitsauftrag

text entschlüsseln

- schlüsselwort ermitteln
  - Länge des Schlüsselwortes ermitteln
    - **Brückenlängen ermitteln**
    - Schlüsselwortlänge berechnen
      - **Verschiebungen mit größten Brücken suchen**
      - **ggT der Verschiebungen ermitteln**
  - Schlüsselwort ermitteln
    - **Text in Gruppen teilen**
    - Schlüsselbuchstaben für jede Gruppe finden
      - **Häufigstes Zeichen in jeder Gruppe suchen**
      - **Schlüsselbuchstaben berechnen**
    - **Schlüsselwort zusammensetzen**
- **text entschlüsseln**

Name	Zweck	Parameter	Rückgabe
textEntschluesseln		geheimtext (String)	Klartext (String)
schluessellaengeBerechnen		geheimtext (String)	Länge des Schlüssels (Zahl)
autokorrelation	Bestimmung der Brücken	geheimtext (String)	Array mit Brücken für jede Verschiebung
sucheBesteVerschiebungen	Berechnen der Verschiebungen für gute Brücken	bruecken (Array) n (Zahl)	Array mit den Verschiebungen für die n größten Brücken
bestimmeSchluessellaenge	Berechnung der Schlüssellänge	verschiebungen (Array)	Schlüssellänge (Zahl)
SchluesselwortBestimmen		geheimtext (String) schluessellaenge (Zahl)	Schlüsselwort (String)
teiltextBilden		text (String) anzahl (Zahl) index (Zahl)	String mit jedem anzahl-ten Zeichen beginnend mit dem index-ten Zeichen von text