Dokumentation Aufgabe 1

Bundeswettbewerb Informatik 2.Runde

Aufgabenstellung:

Es soll ein Programm geschrieben werden, das Codierungen für das Verstecken geheimer Botschaften in Ketten aus einer festen Anzahl von Arten von Perlen erstellt (unterschiedliche Farben und in Aufgabe B auch Größen); die Codes der einzelnen Buchstaben sollen so gewählt werden, dass die Ketten so kurz wie möglich sind.

Das Programm liest Anzahl der unterschiedlichen Perlenfarben , und die Größen der Perlenarten, sowie den zu codierenden Text ein und gibt eine Codetabelle (Textsymbol => Code für alle im Text vorkommenden ' beliebigen Unicode-Symbole') sowie die Länge des codierten Codes aus.

Auf

"Idee":

Im Aufgabentext wurde "präfixfreier Code" erwähnt. Weiteres Suchen führte mich auf die Methode von Huffman zur Erzeugung kleinster präfixfreier Codes. Ich habe also diese Methode programmiert.

Es ist klar, dass die Buchstaben/Symbole, die im Text am häufigsten vorkommen die kürzesten Codes erhalten müssen und nach kleinerer Häufigkeit größer werden können.

Huffmans Methode baut dazu einen Baum auf, der für diese Aufgabe immer so viele Äste hat, wie Perlfarben n. Dazu nimmt man aus der Liste der Symbole, die nach Häufigkeit geordnet sind

die ersten n Symbole in der selben Ordnung heraus und erstellt eine Liste aus den n Symbolen, dann addiert man deren Häufigkeiten zu s und steckt die diese n-Liste zurück in die Liste der Symbole und nimmt diese als Symbol mit der Häufigkeit s an. Die Liste wird wieder nach Häufigkeit sortiert und die n ersten Elemente (Symbole/n-Listen) herausgenommen, zu n-Listen gefasst, Häufigkeiten addiert (=>s') und mit als Element mit der Häufigkeit s' wieder zurück in die Symboleliste getan. Dies wird wiederholt und irgendwann ist nur eine einzige n-Liste mit Symbolen oder/und n-Listen enthalten. Diese n-Liste ist wie ein Baum mit n Ästen.

Danach kann der 'Baum' nach Tiefe-zuerst (depth-first) durchgesucht werden um den Code der Symbole zu bauen: Wenn das erste Element angesehen wird, dann wird dem Code das erste Code-Symbol angehängt (z.B. 'A'), wird das zweite Element angesehen, wird letzte angehängte Code-Symbol durch das zweite Code-Symbol (z.B. 'B') ersetzt und so weiter bis zum n-ten Element, nach dem das letzte Symbol im Code gelöscht wird und diese Liste durchgegangen ist. Ist das angesehene Element ein Textsymbol, dann wird diesem Textsymbol der gerade vorhandene Code zugeordnet und das nächste Element angesehen (also auch das letzte Symbol im Code geändert). Ist das angesehene Element eine n-liste, dann wird das erste Element der n-Liste angesehen (also ein z.B 'A'). Ist eine n-Liste beendet, gilt das Element als angesehen, d.h. in oberen n-Liste wird das nächste Element angesehen (wie oben gesagt wird dann das Code-Symbol aus dem Code gelöscht)

Da dies für jedes Element immer gleich gemacht wird, egal wie 'tief' man in den Baum geht, kann man eine 'rekursive Funktion', also eine Funktion, die sich selbst aufruft, benutzen: Die Funktion

nimmt als Argument eine n-Liste (l) und eine Liste aus Code-Symbolen (der Pfad im Baum, p). Dann wird in einer Schleife über alle Elemente in l gesehen, ob es ein Symbol ist oder eine liste. Wenn es ein Symbol ist, wird Symbol und was gerade in p steht in ein Dict getan. Wenn es eine Liste ist, wird die Funktion mit dieser Liste und p' aufgerufen (p' ist p mit dem Code-Symbol des gerade angesehenen Elements angehängt) und das zurückgegebene Dict in das oben genannte Dict getan. Das Dict wird von der Funktion zurückgegeben. So hat man, wenn man diese Funktion mit der gesamten Huffman-Liste und einem leeren p aufruft am Ende ein Dict mit den Textsymbol:Code-Paaren.

Mit diesem Dict wird der Text codiert und dessen Länge (Anzahl der Code-Symbole=Perlen) berechnet und ausgegeben.

Zu B) hat man den codierten Text, so muss das häufigste Code-Symbol die kleinste Perle sein. Es müsste vielleich der Code noch so geschrieben werden dass sicher ist, dass die Code-Symbole für die häufigsten Textsymbole die selben sind, so dass die auch die kleinsten Perlen bekommen, wenn man die häufigsten Code-Symbole den kleinsten Perlen zugeordnet werden. Dafür hatte ich aber keine Zeit.

Umsetzung:

```
import sys
import string # für das Code-Alphabet aus string.ascii_uppercase
```

Hier importiere ich die Bibliothek „sys“. Ich brauche sys, um den Filenamen der Eingabedatei (Text) auf der Kommandozeile zu bekommen.

Bibliothek string benutze ich für ein Alphabet aus Großbuchstaben für die Codes, was man mit string.ascii_uppercase bekommt.

Dies ist die Definition der Einlesefunktion read_data(). Diese nimmt den Namen der Eingabedatei

```
def read_data(filename):
    file = open(filename, "r")
    contents = [l.rstrip() for l in file.readlines()]
    return {'kinds': contents[0], 'sizes': contents[1], 'text':
        contents[2]}
```

```
# nach Zahl der aufkommen sortieren
# siehe https://stackoverflow.com/questions/10695139/sort-a-
list-of-tuples-by-2nd-item-integer-value

def sort_by_number(tuple_list):
    return sorted(tuple_list, key=lambda element: element[1])
```

sort_by_number() ist eine Hilfsfunktion mit der ich Symbole nach Häufigkeit sortiere. Dabei ist tuple_list eine Liste aus Tupeln die aus (Textsymbol, Häufigkeit), und Häufigkeit ist einfach die Zahl des Textsymbols im Text. Die Sortierung geht mit dem 2. Element der Tupel, also mit der Häufigkeit des Textsymbols.

```
def analyze_text(text):
    alphabet = {c: text.count(c) for c in text}
    return sort_by_number(alphabet.items())
```

In der Funktion analyze_text() zähle ich wieviele es von den Textsymbolen gibt, also deren Häufigkeit. Ich gebe das als Liste

zurück, die nach der Häufigkeit sortiert ist, da ich das nicht mit einem Dict machen kann.

```
def build_huffman_tree(tuple_list, num_branches):
```

Die Funktion build_huffman_tree() nimmt als Argument die nach Häufigkeit sortierte Liste aus Tupeln (Textsymbol:Häufigkeit) tuple_list und num_branches,die Zahl der verschiedenen Perlfarben, und gibt eine Liste aus num_branches vielen Listen bzw. Textsymbolen aus, der 'Huffman-Baum'.

```
tree_list = tuple_list.copy()
```

Wir müssen das kopieren, sonst ändern wir die tuple_list.

Die nach Anzahl sortierte Liste soll in einen Baum umgewandelt werden. Dabei werden die seltensten Zeichen zuerst genommen und werden zu Gruppen von 'kinds' vielen Zeichen zusammen gefasst; es wird die Gesamtanzahl der zusammengefassten Zeichen berechnet und zur Zahl der Gruppe genommen und in einem Tupel (Gruppe, Zahl) wieder in die Liste gesteckt. Die Liste wird wieder sortiert und danach werden Zeichen und Gruppen gleich behandelt und immer weiter zu Gruppen von Zeichen oder Gruppen zusammen gefasst, bis es nur ein einziges Element in der Liste gibt.
Solange es mehr als ein einziges Element in der tuple liste gibt:

```
while len(tree_list) > 1:
```

nehme die ersten 'num_branches' viele Elemente aus der Liste, wenn es nicht mehr so viele Elemente gibt, nehme alle Elemente:

```
num_group = min(kinds, len(tree_list))
group = tree_list[:num_group]
tree_list = tree_list[num_group:]
```

Zeichen und Anzahl aus den Tupeln holen:

```
symbols = [s[0] for s in group]
numbers = [s[1] for s in group]
```

Tue das Tupel (symbols, numbers) zurück in tree_list

```
tree_list.append((symbols, sum(numbers)))
```

und sortiere wieder:

```
treelist = sort_by_number(tree_list)
```

Wenn nur noch eine einzige Liste in tree_list ist, gebe ich die Liste aus Listen, also den Baum zurück, die Häufigkeit ist nicht mehr nötig.

```
return tree_list[0][0]
```

Nun definiere ich die Funktion, die durch den Huffman-Baum geht und alle Symbole sammelt. Dies ist eine sogenannte rekursive Funktion, d.h. dass in der Definition dieser Funktion ein Aufruf der Funktion gemacht wird. Wenn die Eingabe-Liste weitere Listen enthält, wird diese mit derselben Funktion bearbeitet, nur der path (der Code des Elements) ändert sich dabei. Wenn ein Symbol gefunden wird, wird dieses mit dem Wert von path in ein Dict gegeben, wenn alle Elemente verarbeitet sind, wird dieses Dict zurück gegeben.

```
def recursive_find(inlist, path):
    found_dict = {}
    for i in range(len(inlist)):
        element = inlist[i]
        nextpath = path.copy()
        nextpath.append(i)
        if type(element) is list:
            found_dict.update(recursive_find(element,
```

```
nextpath))
            else:
                found_dict[element] = nextpath.copy()
    return found_dict
```

Nun zur Ausführung. Wir lesen die Datei ein, die auf der Kommandozeile als Argument übergeben wird.

```
data = read_data(sys.argv[1])
```

Dann holen wir die Zahl der Perlenfarben, die verschiedenen Größen der Perlen und den Text aus den gelesenen Daten.

```
kinds = int(data['kinds'])
sizes = [int(s) for s in data['sizes'].split(' ')]
text = data['text']
```

Jetzt werden die Häufigkeiten der einzelnen Zeichen in sortierter Liste erhalten:

```
symbol_numbers = analyze_text(text)
```

Berechne Huffman-Baum zur Erzeugung der Codes

```
code_tree = build_huffman_tree(symbol_numbers)
```

Nehme Codes aus Baum -> Dict aus "Zeichen: Code"

```
codes = recursive_find(code_tree, [])
```

2. "Beipackzettel": Gebe Tabelle der Codes aus .

Einfach zeilenweise 'Zeichen: Coding' ausgeben (Zahlen zu

Buchstaben übersetzen). Um es etwas schöner aussehen zu lassen,

die kürzesten Codes (häufigste Zeichen) zuerst.

Code-alphabet aus der Anzahl 'kinds' erzeugen:

```
code_basis = list(string.ascii_uppercase)
if kinds > len(code_basis):
    error("Bitte die Basis des Codes vergrößern")
code_alphabet = code_basis[:kinds]
print("Übersetzungstabelle:")
print(f"Es gibt {kinds} Arten von Perlen: {code_alphabet}.")
```

Ausgabe (und gleichzeitig Umwandlung in Dict aus 'Zeichen: Codestring':

```
translation_dict = {}
for symbol in reversed(symbol_numbers):
    code_string = "".join([code_alphabet[i] for i in
        codes[symbol[0]]])
    translation_dict[symbol[0]] = code_string
    print(f""{symbol[0]}" => {code_string}")
```

3. Berechne Länge des codierten Eingabetextes:

Text zu Liste von Zeichen:

```
textlist = list(text)
```

Übersetzen

```
coded_textlist = [translation_dict[a] for a in textlist]
```

Zusammenfügen

```
coded_text = "".join(coded_textlist)
print("Der Text:")
print(text)
print("ist codiert:")
print(coded_text)
print(f"und ist {len(coded_text)} Perlen lang.")
```

Aufgabe B)

Im codierten Text sollte die häufigste Perle am kleinsten, und mit geringerer Häufigkeit größere Perlen verwendet werden.

Leider habe ich in der Zeit keine Methode finden können um sicher zu stellen, dass die häufigsten Symbole im Originaltext die kleinsten Perlen benutzen usw.

```
print("\n\nAufgabe B:")
```

Häufigkeiten:

```
numbers = analyze_text(coded_text)
```

Zuordnung Perle (Code-Buchstabe) zu Größe:

```
print({n: s for n in numbers for s in reversed(sorted(sizes))})
ordered_alphabet = [n[0] for n in numbers]
perl_sizes = dict(zip(ordered_alphabet,
    reversed(sorted(sizes))))

print("In der Codierung sind die Perlen so:")
print(", ".join([f"{n} => {perl_sizes[n]} mm"
    for n in perl_sizes]) + ".")
```

Berechnung der Länge der Kette

```
chain_length = sum([perl_sizes[c] for c in list(coded_text)])
print(f"Damit ist der codierte Text {chain_length} mm lang")
```

Anwendung und Beispiele:

Das Programm wird angewendet indem man es im Programmordner mit python3 wie folgt aufruft:

python3 ./perlencode.py <inputfile>

wobei man <inputfile> mit dem Namen der Eingabedatei ersetzt

Beispiele

schmuck00.txt

3

1 1 1

Es war mal ein kleines Gedicht Das machte ein finstres Gesicht Die Welt wird nicht heilen Durch lustige Zeilen Pointe? Auch die gibt es nicht

Output:

Übersetzungstabelle:

Es gibt 3 Arten von Perlen: ['A', 'B', 'C'].

" " => BA

"e"	=>	AB
"i"	=>	AA
"t"	=>	BCA
"n"	=>	BBB
"h"	=>	BBA
"s"	=>	ACC
"c"	=>	ACA
"l"	=>	BCCC
"r"	=>	BCCA
"a"	=>	BCBC
"u"	=>	BBCB
"D"	=>	BBCA
"d"	=>	ACBC
"g"	=>	ACBB
"G"	=>	ACBA
"m"	=>	BCCBC
"w"	=>	BCCBB
"b"	=>	BCCBA
"A"	=>	BCBBC
"?"	=>	BCBBB
"o"	=>	BCBBA
"P"	=>	BCBAC
"Z"	=>	BCBAB
"W"	=>	BCBAA

"f" => BBCCC
"k" => BBCCB
"E" => BBCCA

Der Text:

Es war mal ein kleines Gedicht Das machte ein finstres Gesicht Die Welt wird nicht heilen Durch lustige Zeilen Pointe? Auch die gibt es nicht

ist codiert:

BBCCAACCBABCCBBBCBCBCCABABCCBCBCBCBCCCBAABAABBBBABBCCBBCCCA
BAABBBABACCBAACBAABACBCAAACABBABCABABBCABCBCACCBABCCBCBCBCAC
ABBABCAABBAABAABBBBABBCCCAABBBACCBCABCCAABACCBAACBAABACCAAACA
BBABCABABBCAAAABBABCBAAABBCCCBCABABCCBBAABCCAACBCBABBBAAACABB
ABCABABBAABAABCCCABBBBBABBCABBCBBCCAACABBABABCCCBBCBACCBCAAA
ACBBABBABCBABABAABCCCABBBBBABCBACBCBBAAABBBBCAABBCBBBBABCBBCB
BCBACABBABAACBCAAABBAACBBAABCCBABCABAABACCBABBBAAACABBABCA

und ist 420 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

C => 1 mm, A => 1 mm, B => 1 mm.

Damit ist der codierte Text 420 mm lang

schmuck01.txt

5

1 1 1 1 1

Die Bäume im Ofen lodern. Die Vögel locken am Grill. Die Sonnenschirme vermodern. Im übrigen ist es still. Es stecken die Spargel aus Dosen Die zarten Köpfchen hervor. Bunt ranken sich künstliche Rosen In Faschingsgirlanden empor. Ein Etwas, wie Glockenklingen, Den Oberkellner bewegt, Mir tausend Eier zu bringen, Von Osterstören gelegt. Ein süßer Duft von Havanna Verweht in ringelnder Spur. Ich fühle an meiner Susanna Erwachende neue Natur. Es lohnt sich manchmal, zu lieben, Was kommt, nicht ist oder war. Ein Frühlingsgedicht, geschrieben Im kältesten Februar...

Output

Übersetzungstabelle:

Es gibt 5 Arten von Perlen: ['A', 'B', 'C', 'D', 'E'].

" " => C
"e" => A
"n" => ED
"r" => EB
"i" => EA
"s" => DD
"t" => DC
"a" => DB
"l" => DA
"h" => BD
"c" => BC
"g" => BA
"o" => EEE
"m" => EED
"u" => EEC
"." => EEB
"k" => EEA
"d" => ECE
"," => ECD
"E" => ECC
"b" => ECA
"D" => DEE
"w" => DED
"ü" => DEB
"p" => DEA
"I" => BEE
"v" => BED
"S" => BEC
"f" => BEB
"F" => BEA
"z" => BBE
"ö" => BBD
"V" => BBC
"O" => BBB
"G" => BBA
"ä" => ECBE
"B" => ECBD
"..." => ECBC
"W" => ECBB
"N" => ECBA

"H" => DECE
"ß" => DECD
"M" => DECC
"R" => DECB

"K" => DECA

Der Text:

Die Bäume im Ofen lodern. Die Vögel locken am Grill. Die Sonnenschirme vermodern. Im übrigen ist es still. Es stecken die Spargel aus Dosen Die zarten Köpfchen hervor. Bunt ranken sich künstliche Rosen In Faschingsgirlanden empor. Ein Etwas, wie Glockenklingen, Den Oberkellner bewegt, Mir tausend Eier zu bringen, Von Osterstören gelegt. Ein süßer Duft von Havanna Verweht in ringelnder Spur. Ich fühle an meiner Susanna Erwachende neue Natur. Es lohnt sich manchmal, zu lieben, Was kommt, nicht ist oder war. Ein Frühlingsgedicht, geschrieben Im kältesten Februar...

ist codiert:

DEEEAACECBDECBEEECEEDACEAEEDCBBBBEBAEDCDAEEEECEAEBEDEEBCDEEEAA
CBBCBBDBAADACDAEEEBCEEAAEDCDBEEDCBBAEBEADADAEEBCDEEEAACBECEEE
EDEDAEDDDBCBDEAEBEEDACBEDAEBEEDEEEECEAEBEDEEBCBEEEEDCDEBECAEBE
ABAAEDCEADDDCCADDCDDDCEADADAEEBCECCDDCDDDCABCEEAAEDCECEEEAAC
BECDEADBEBBAADACDBEECDDCDEEEEEDDAEDCDEEEAACBBEDBEBDCAEDCDECA
BBDDEABEBBCBDAEDCBDAEBBEDEEEEBEEBCECBDEECEDDCCEBDBEDEEAAEDCDD
EABCBDCEEADEBEDDDDCDAEABCBDACDECBEEEDDAEDCBEEEDCBEADBDDBCBDE
AEDBADDBAEAEBDADBEDECEAEDCAEEDDEAEEEEBEEBCECCEAEDCECCDCDEDDB
DDECDCDEDEAACBBADAEEEBCEEAAEDEEADAEAEDBAAEDECDCDEEAEDCBBBECAA
EBEEAADADAEDAEBCECAADEDABADCECDCDECCEAEBCDCDBEECDDAEDECECECCC
EAAEBCBBEEECCECAEBEAEDBAAEDECDCBBCEEEEDCBBBDDDCAEBDDDCBBDEBAE
DCBAADAABADCEEBCECCEAEDCDDDEBDECDAEBCDEEEECBEBDCCBEDEEEEDCDE
CEDBBEDDBEDEDDBCBBCAEBDEDABDDCCEAEDCEBEAEDBAADAEDECEAEBCBECD
EAEECEBEEBCBEEBCBDCBEBDEBBDDAACDBEDCEEDAEAEDAEBCBECEECDDDBEDE
DDBCECCEBDEDDBBCBDAEDECEACEDAEECACECBADBDCEECEBEEBCECCDDCDAE
EEBDEDDCCDDEABCBDCEEDDBEDBCBDEEDDBDAECDCBBEEECCDAEAAECAAEDEC
DCECBBDBDDCEEAEEEEEDEEDDCECDCEDEABCBDDCCEADDDCCEEEECEAEBCDED
DBEBEEBCECCEAEDCBEAEBDEBBDDAEAEDBADDBAAECEEABCBDDCECDCBAADDB
CBDEBEAAECAAEDCBEEEEDCEEAECBEDADCADDDCAEDCBEAAECAEBEECDBEBECB
C

und ist 1150 Perlen lang.

Aufgabe B: In der Codierung sind die Perlen so: A => 1 mm, B => 1 mm, C => 1 mm, D => 1 mm, E => 1 mm. Damit ist der codierte Text 1150 mm lang

schmuck0.txt

2

1 1

DIE SONNE SOLL DIR IMMER SCHEINEN

Output

Übersetzungstabelle:

Es gibt 2 Arten von Perlen: ['A', 'B'].

" " => BBB

"E" => BBA

"N" => ABA

"I" => AAB

"S" => AAA

"M" => BABA

"R" => BAAB

"L" => BAAA

"O" => ABBB

"D" => ABBA

"H" => BABBB

"C" => BABBA

Der Text:

DIE SONNE SOLL DIR IMMER SCHEINEN

ist codiert:

ABBAAABBBABBBAAAABBBABAABABBABBBAAAABBBBAAABAAABBBABBAAABBAABBB
BAABBABABABABBABAABBBBAAABABBABABBBBBAAABABABBAABA

und ist 113 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 1 mm, B => 1 mm.

Damit ist der codierte Text 113 mm lang

schmuck1.txt

3

1 1 2

BWINF steht für "Die Bundesweiten Informatikwettbewerbe"

Output

Übersetzungstabelle:

Es gibt 3 Arten von Perlen: ['A', 'B', 'C'].

"e" => CA

"t" => BA

" " => AC

"w" => CCA

"n" => CBC

"i" => CBB

"r" => CBA

"b" => BCC

""" => BCB

"f" => BCA

"s" => BBC

"l" => BBB

"B" => BBA

"k" => ABC

"a" => ABB

"m" => ABA

"o" => AAC

"d" => AAB

"u" => AAA

"D" => CCCC

"ü" => CCCB

"h" => CCCA

"F" => CCBC

"N" => CCBB

"W" => CCBA

Der Text:

BWINF steht für "Die Bundesweiten Informatikwettbewerbe"

ist codiert:

BBACCBABBBCCBBCCBCACBBCBACACCCABAACBCACCCBCBAACBCBCCCCCBBCA
ACBBAAAACBCAABCABBCCCACACBBBACACBCACBBBCBCBCAAACCBAABAABBBAC
BBABCCCACABABABCCCACCACACBABCCCABCB

und ist 154 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 2 mm, B => 1 mm, C => 1 mm.

Damit ist der codierte Text 197 mm lang

schmuck2.txt

2

1 5

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa bcdefgh

Output

Übersetzungstabelle:

Es gibt 2 Arten von Perlen: ['A', 'B'].

"a" => B

"h" => ABBB

"g" => ABBA

"f" => ABAB

"e" => ABAA

"d" => AABB

"c" => AABA

"b" => AAAB

" " => AAAA

Der Text:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa bcdefgh

ist codiert:

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBAAAAAAABAABAAABBABAAABABABBAABBB

und ist 65 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 5 mm, B => 1 mm.

Damit ist der codierte Text 145 mm lang

schmuck3.txt

3

1 2 3

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb cccccccccccccccccccccccccccccccccc defgh

Output

Übersetzungstabelle:

Es gibt 3 Arten von Perlen: ['A', 'B', 'C'].

"c" => B

"b" => A

"a" => CC

" " => CBC

"h" => CBB

"g" => CBA

"f" => CAC

"e" => CAB

"d" => CAA

Der Text:

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb cccccccccccccccccccccccccccccccccc defgh

ist codiert:

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCBCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACBCBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBCBCCAACABCACCBACBB

und ist 160 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 3 mm, B => 2 mm, C => 1 mm.

Damit ist der codierte Text 279 mm lang

schmuck4.txt

2

1 5

abcdefghijklmn

Output

Übersetzungstabelle:

Es gibt 2 Arten von Perlen: ['A', 'B'].

"n" => AAB

"m" => AAA

"l" => BBBB

"k" => BBBA

"j" => BBAB

"i" => BBAA

"h" => BABB

"g" => BABA

"f" => BAAB

"e" => BAAA

"d" => ABBB

"c" => ABBA

"b" => ABAB

"a" => ABAA

Der Text:

abcdefghijklmn

ist codiert:

ABAAABABABBAABBBBAAABAABBABABABBBBAABBABBBBABBBBAAAAAB

und ist 54 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 5 mm, B => 1 mm.

Damit ist der codierte Text 154 mm lang

schmuck5.txt

7

1 1 2 3 4 5 6

Summary This paper gives a method for constructing minimum-redundancy prefix codes for the general discrete noiseless channel without constraints. The costs of code letters need not be equal, and the symbols encoded are not assumed to be equally probable. A solution had previously been given by Huffman in 1952 for the special case in which all code letters are of equal cost. The present development is algebraic. First, structure functions are defined, in terms of which necessary and sufficient conditions for the existence of prefix codes may be stated. From these conditions, linear inequalities are derived which may be used to characterize prefix codes. Gomory's integer programming algorithm is then used to construct optimum codes subject to these inequalities; an elegant combinatorial approach to obtain a strictly computational experience is presented to demonstrate the practicability of the method. Finally, some additional coding problems are discussed and a problem of classification is treated.

Output

Übersetzungstabelle:

Es gibt 7 Arten von Perlen: ['A', 'B', 'C', 'D', 'E', 'F', 'G'].

" " => C

"e" => B

"t" => EG

"i" => EF

"o" => EE

"s" => ED

"a" => EC

"n" => EB

"r" => EA

"c" => DG

"d" => DF

"l" => DD

"m" => DC

"h" => DB

"u" => DA

"f" => AG

"p" => AF

"b" => AE

"y" => AD

"g" => AC

"." => AA

"," => DEF

"q" => DEE

"x" => DED

"v" => DEC

"w" => DEB

"F" => DEA

"T" => ABG

";" => ABF

"j" => ABE

"’" => ABD

"G" => ABC

"z" => ABB

"2" => ABA

"5" => DEGG

"9" => DEGF

"1" => DEGE

"H" => DEGD

"A" => DEGC

"-" => DEGB

"S" => DEGA

Der Text:

Summary This paper gives a method for constructing minimum-redundancy prefix codes for the general discrete noiseless channel without constraints. The costs of code letters need not be equal, and the symbols encoded are not assumed to be equally probable. A solution had previously been given by Huffman in 1952 for the special case in which all code letters are of equal cost. The present development is algebraic. First, structure functions are defined, in terms of which necessary and sufficient conditions for the existence of prefix codes may be stated. From these conditions, linear inequalities are derived which may be used to characterize prefix codes. Gomory's integer programming algorithm is then used to construct optimum codes subject to these inequalities; an elegant combinatorial approach to obtain a strictly computational experience is presented to demonstrate the practicability of the method. Finally, some additional coding problems are discussed and a problem of classification is treated.

ist codiert:

DEGADADCDCECEAADCABGDBEFEDCAFECAFBEACACEFDECBEDCECCDCBEGDBEE
DFCAGEEEACDGEEEBEDEGEADADGEGEFEBACCDCEFEBEFDCDADCDEGBEABDFDAE
BDFECEBDGADCAFEABAGEFDEDCDGEEDFBEDCAGEEEACEGDBBCACBEBBEAECDD
CDFEFEDDGEABEGBCEBEEEFEDBDDBEDEDCDGDBECEBEBBDDCDEBEFEGDBEEDAE
GCDGEEEBEDEGEAECEFEBEGEDAACABGDBBCDGEEEDEGEDCEEAGCDGEEDFBCDD
BEGEGBEAEDCEBBBDFCEBEEEGCAEBCBDEEDAECDDDEFCECEBDFCEGDBBCEDAD
DCAEEEDDEDCBEBDGEEDFBDFCECEABCEBEEEGCECEDEDDADCBDFCEGEECAEBC
BDEEDAECDDDDADCAFEAEEAEECAEDDBAACDEGCCEDEEDDDAEGEFEEEBCDBECD
FCAFEABDECEFEEDAEDDDADCAEBBEBCACEFDECBEBCAEADCDEGDDAAGAGDCEC
EBCEFEBCDEGEDEGFDEGGABACAGEEEACEGDBBCEDAFBDGEFECDDCDGECEDBCE
FEBCDEBDBEFDGDBCECDDDDCDGEEDFBCDDBEGEGBEAEDCECEABCEEAGCBDEE
DAECDDCDGEEEDEGAACABGDBBCAFEABEDBEBEGCDFBDECBDDEEAFDCBEBEGCE
FEDCECDDACBAEEAECEFDGAACDEAEFEAEDEGDEFCEDEGEADADGEGDAEABCAGDA
EBDGEGEFEEEBEDCECEABCDFBAGEFEBBDFDEFCEFEBCEGBEADCEDCEEAGCDEBD

BEFDGDBCEBBDGBEDEDECEAADCECEBDFCEDDAAGAGEFDGEFBEBEGCDGEEEBDF
EFEGEFEEEBEDCAGEEEACEGDBBCBDEDEFEDEGBEBDGBCEEAGCAFEABAGEFDEDC
DGEEDFBEDCDCECADCAEBCEDEGECEGBDFAACDEAEAEEDCCEGDBBEDBCDGEEEB
DFEFEGEFEEEBEDDEFCDDEFEBBECEACEFEBBDEEDAECDDEFEGEFBEDCECEABCDF
BEAEFDECBDFCDEBDBEFDGDBCDCECADCAEBCDAEDBDFCEGEECDGDBECEAECDG
EGBEAEFABBBCAFEABAGEFDEDCDGEEDFBEDAACABCEEDCEEEAADABDEDCEFEBEG
BACBEACAFEAEEACEAECDCDCEFEBACCECDDACEEEAEFEGDBDCCEFEDCEGDBBEB
CDAEDBDFCEGEECDGEEEBEDEGEADADGEGCEEAFEGEFDCDADCCDGEEDFBEDCED
DAAEABEBDGEGCEGEECEGDBBEDBCEFEBBDEEDAECDDEFEGEFBEDABFCECEBCBD
DBACECEBEGCDGEEDCAEEFEBECEGEEEAEFECDDCECAFAFEAEEECDGDBCEGEECE
EAEEGECEFEBCECCEDEGEAEFDGEGDDADCDGEEDCAFDAEGECEGEFEEEBECDDCB
DEDAFBEAEFBEBDGBCEFEDCAFEABEDBEBEGBDFCEGEECDFBDCEEEBEDEGEAECEG
BCEGDBBCAFEAECDGEGEFDGECAEEFDDEFEGADCEEAGCEGDBBCDCBEGDBEEDFA
ACDEAEFEBECDDDDADDEFCEDEEDCBCECDFDFEFEGEFEEEBECDDCDGEEDFEFEBA
CCAFEAEEAEDDBDCEDCECEABCDFEFEDDGDAEDEDBDFCECEBDFCECCAFEAEEAED
DBDCCEEAGCDGDDECEDEDEFAGEFDGECEGEFEEEBCEFEDCEGEABECEGBDFAA

und ist 1813 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

F => 6 mm, G => 5 mm, A => 4 mm, B => 3 mm, C => 2 mm, D => 1 mm, E => 1 mm.

Damit ist der codierte Text 4278 mm lang

schmuck6.txt

3

1 2 3

古英雄未遇時，都無大志，非止鄧禹希文學、馬武望督郵也。晉文公有妻有馬，不肯去齊。

Output

Übersetzungstabelle:

Es gibt 3 Arten von Perlen: ['A', 'B', 'C'].

", " => BAA

"有" => ACC

"。" => ACB

"馬" => ACA

"文" => ABC

"齊" => ABB

"去" => ABA

"肯" => AAC

"不" => AAB

"妻" => AAA

"公" => BCCC

"晉" => BCCB

"也" => BCCA

"郵" => BCBC

"督" => BCBB

"望" => BCBA

"武" => BCAC

"、" => BCAB

"學" => BCAA

"希" => BBCC

"禹" => BBCB

"鄧" => BBCA

"止" => BBBC

"非" => BBBB

"志" => BBBA

"大" => BBAC

"無" => BBAB

"都" => BBAA

"時" => BACC

"遇" => BACB

"未" => BACA

"雄" => BABC

"英" => BABB

"古" => BABA

Der Text:

古英雄未遇時，都無大志，非止鄧禹希文學、馬武望督郵也。晉文公有妻有馬，不肯去齊。

ist codiert:

BABABABBBABCBACABACBBACCBAABBAABBABBBACBBBABAABBBBBBBCBBCABBCBBBCCABCBCAABCABACABCACBCBABCBBBCBCBCCAACBBCCBABCBCCCACCAAAACCACABAAAABAACABAABBACB

und ist 144 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

C => 3 mm, A => 2 mm, B => 1 mm.

Damit ist der codierte Text 264 mm lang

schmuck7.txt

Inhalt ist zu groß für dieses Dokument.

Output

Übersetzungstabelle:

Es gibt 10 Arten von Perlen: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'].

" " => I

"e" => H

"n" => F

"i" => E

"r" => D

"s" => C

"a" => B

"d" => A

"h" => JJ

"t" => JI

"u" => JG

"l" => JF

"c" => JE

"g" => JC

"m" => JB

"o" => JA

"b" => GJ

"," => GI

"f" => GH

"w" => GG

"." => GF

"k" => GE

"z" => GC

"S" => GB

"ä" => GA

"ü" => JHJ

"ö" => JHI

"v" => JHH

"F" => JHG

"p" => JHF

"ß" => JHE

"D" => JHD

"E" => JHC

"V" => JHB

"W" => JDJ

"N" => JDI

"K" => JDH

"B" => JDG

"M" => JDF

"A" => JDE

"H" => JDD

"I" => JDC

"L" => JDB

"R" => JDA

"G" => GDJ

"T" => GDI

"Z" => GDG

"j" => GDF

"J" => GDE

"-" => GDD

";" => GDC

")" => GDB

"(" => GDA

"_" => JHAJ

":" => JHAI

"O" => JHAH

"U" => JHAG

"]" => JHAF

"[" => JHAE

"“" => JHAD

„" => JHAC

"P" => JHAB

"1" => GDHJ

"4" => GDHI

"3" => GDHH

"?" => GDHG

"5" => GDHF

"!" => GDHE

"2" => GDHD

"6" => GDHC

"0" => GDHB

"8" => GDHA

"Ä" => JHAAJ

"q" => JHAAI

"’" => JHAAH

"Q" => JHAAG

"9" => JHAAF

"7" => JHAAE

"C" => JHAAD

"y" => JHAAC

"x" => JHAAB

"Ö" => JHAAA

Der Text:

Aus dem Leben unserer Vögel. Lektion 1. Bekannte Vögel. Ich möchte wissen, wie viele

...gekürzt...

euch zu Freunden zu machen, und könnt dann viel über ihre Lebensweise erfahren.

ist codiert:

JDEJGCIAHJBIJDBHGJHFIJGFCHDHDIJHBJHIJCHJFGFIJDBHGEJIEJAFIGDHJGFIJDGHGE...gekürzt...

JHIFFJIIABFFIJHHEHJFIJHJGJHDIEJJDHIJDBHGJHFCGGHECHIHDGHBJJDHFGF

und ist 119087 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 4 mm, B => 3 mm, C => 2 mm, D => 1 mm, E => 1 mm, F => 1 mm, G => 1 mm, H => 1 mm, I => 1 mm, J => 1 mm.

Damit ist der codierte Text 153144 mm lang

schmuck8.txt

5

1 1 2 2 3

古英雄未遇時，都無大志，非止鄧禹希文學、馬武望督郵也。晉文公有妻有馬，不肯去齊。光武貧時，與李通訟逋租於嚴尤。尤奇而目之。光武歸謂李通曰：「嚴公寧目君耶」窺其意，以得嚴君一盼為榮。韓蘄王為小卒時，相士言其日後封王。韓大怒，以為侮己，奮拳毆之。都是一般見解。鄂西林相公《辛丑元日》云：「攬鏡人將老，開門草未生。」《詠懷》云：「看來四十猶如此，便到百年已可知。」皆作郎中時詩也。玩其詞，若不料此後之出將入相者。及其為七省經略，《在金中丞席上》云：「問心都是酬恩客，屈指誰為濟世才」《登甲秀樓》絕句云：「炊煙卓午散輕絲，十萬人家飯熟時。問訊何年招濟火，斜陽滿樹武鄉祠。」居然以武侯自命，皆與未得志時氣象迥異。張桐城相公則自翰林至作首相，詩皆一格。最清妙者：「柳陰春水曲，花外暮山多。」「葉底花開人不見，一雙蝴蝶已先知。」「臨水種花知有意，一枝化作兩枝看。」《扈蹕》云：「誰憐七十龍鐘叟，騎馬踏冰星滿天」《和皇上〈風箏〉》云：「九霄日近增華色，四野風多仗寶繩。」押「繩」字韻，寄托遙深。 二 楊誠齋曰：「從來天分低拙之人，好談格調，而不解風趣。何也格調是空架子，有腔口易描；風趣專寫性靈，非天才不辦。」餘深愛其言。須知有性情，便有格律；格律不在性情外。《三百篇》半是勞人思婦率意言情之事；誰為之格，誰為之律而今之談格調者，能出其範圍否況皋、禹之歌，不同乎

《三百篇》;《國風》之格，不同乎《雅》、《頌》：格豈有一定哉許渾云：「吟詩好似成仙骨，骨里無詩莫浪吟。」詩在骨不在格也。

Output

Übersetzungstabelle:

Es gibt 5 Arten von Perlen: ['A', 'B', 'C', 'D', 'E'].

"，" => EA

"。" => DD

"」" => AC

"「" => AB

"》" => AA

"《" => EEE

"：" => EED

"格" => ECC

"之" => ECB

"不" => EBE

"云" => EBD

"為" => EBC

"有" => EBB

"一" => EBA

"其" => DEE

"時" => DED

"風" => BEE

"詩" => BED

"人" => BEC

"相" => BEB

"武" => BEA

"；" => BDE

"誰" => BDD

"在" => BDC

"知" => BDB

"是" => BDA

"公" => BCE

"也" => BCD

"骨" => BCC

"律" => BCB

"情" => BCA

"性" => BBE

"調" => BBD

"天" => BBC

"花" => BBB

"者" => BBA

"作" => BAE

"皆" => BAD

"百" => BAC

"十" => BAB

"日" => BAA

"言" => AEE

"以" => AED

"意" => AEC

"而" => AEB

"嚴" => AEA

"馬" => ADE

"、" => ADD

"都" => ADC

"未" => ADB

"吟" => ADA

"乎" => EECE

"同" => EECD

"篇" => EECC

"三" => EECB

"趣" => EECA

"談" => EEBE

"好" => EEBD

" " => EEBC

"深" => EEBB

"繩" => EEBA

"枝" => EEAE

"多" => EEAD

"外" => EEAC

"水" => EEAB

"自" => EEAA

"滿" => EDEE

"何" => EDED

"才" => EDEC

"濟" => EDEB

"問" => EDEA

"上" => EDDE

"七" => EDDD

"出" => EDDC

"中" => EDDB

"已" => EDDA

"年" => EDCE

"便" => EDCD

"此" => EDCC

"四" => EDCB

"來" => EDCA

"看" => EDBE

"開" => EDBD

"將" => EDBC

"林" => EDBB

"解" => EDBA

"見" => EDAE

"後" => EDAD

"王" => EDAC

"韓" => EDAB

"得" => EDAA

"君" => ECEE

"曰" => ECED

"目" => ECEC

"尤" => ECEB

"通" => ECEA

"李" => ECDE

"與" => ECDD

"光" => ECDC

"文" => ECDB

"禹" => ECDA

"非" => ECAE

"志" => ECAD

"大" => ECAC

"無" => ECAB

"浪" => ECAA

"莫" => DECE

"里" => DECD

"仙" => DECC

"成" => DECB

"似" => DECA

"渾" => DEBE

"許" => DEBD

"哉" => DEBC

"定" => DEBB

"豈" => DEBA

"頌" => DEAE

"雅" => DEAD

"國" => DEAC

"歌" => DEAB

"皐" => DEAA

"況" => DCEE

"否" => DCED

"圍" => DCEC

"範" => DCEB

"能" => DCEA

"今" => DCDE

"事" => DCDD

"率" => DCDC

"婦" => DCDB

"思" => DCDA

"勞" => DCCE

"半" => DCCD

"須" => DCCC

"愛" => DCCB

"餘" => DCCA

"辦" => DCBE

"靈" => DCBD

"寫" => DCBC

"專" => DCBB

"描" => DCBA

"易" => DCAE

"口" => DCAD

"腔" => DCAC

"子" => DCAB

"架" => DCAA

"空" => DBEE

"拙" => DBED

"低" => DBEC

"分" => DBEB

"從" => DBEA

"齋" => DBDE

"誠" => DBDD

"楊" => DBDC

"二" => DBDB

"遙" => DBDA

"托" => DBCE

"寄" => DBCD

"韻" => DBCC

"字" => DBCB

"押" => DBCA

"寶" => DBBE

"仗" => DBBD

"野" => DBBC

"色" => DBBB

"華" => DBBA

"増" => DBAE

"近" => DBAD

"霄" => DBAC

"九" => DBAB

"箏" => DBAA

"〈" => DAEE

"皇" => DAED

"和" => DAEC

"星" => DAEB

"冰" => DAEA

"踏" => DADE

"騎" => DADD

"叟" => DADC

"鐘" => DADB

"龍" => DADA

"憐" => DACE

"蹕" => DACD

"扈" => DACC

"兩" => DACB

"化" => DACA

"種" => DABE

"臨" => DABD

"先" => DABC

"蝶" => DABB

"蝴" => DABA

"雙" => DAAE

"底" => DAAD

"葉" => DAAC

"山" => DAAB

"暮" => DAAA

"曲" => CEEE

"春" => CEED

"陰" => CEEC

"柳" => CEEB

"妙" => CEEA

"清" => CEDE

"最" => CEDD

"首" => CEDC

"至" => CEDB

"翰" => CEDA

"則" => CECE

"城" => CECD

"桐" => CECC

"張" => CECB

"異" => CECA

"迥" => CEBE

"象" => CEBD

"氣" => CEBC

"命" => CEBB

"侯" => CEBA

"然" => CEAE

"居" => CEAD

"祠" => CEAC

"鄉" => CEAB

"樹" => CEAA

"陽" => CDEE

"斜" => CDED

"火" => CDEC

"招" => CDEB

"訊" => CDEA

"熟" => CDDE

"飯" => CDDD

"家" => CDDC

"萬" => CDDB

"絲" => CDDA

"輕" => CDCE

"散" => CDCD

"午" => CDCC

"卓" => CDCB

"煙" => CDCA

"炊" => CDBE

"句" => CDBD

"絕" => CDBC

"樓" => CDBB

"秀" => CDBA

"甲" => CDAE

"登" => CDAD

"世" => CDAC

"指" => CDAB

"屈" => CDAA

"客" => CCEE

"恩" => CCED

"酬" => CCEC

"心" => CCEB

"席" => CCEA

"丞" => CCDE

"金" => CCDD

"略" => CCDC

"經" => CCDB

"省" => CCDA

"及" => CCCE

"入" => CCCD

"料" => CCCC

"若" => CCCB

"詞" => CCCA

"玩" => CCBE

"郎" => CCBD

"可" => CCBC

"到" => CCBB

"如" => CCBA

"猶" => CCAE

"懷" => CCAD

"詠" => CCAC

"生" => CCAB

"草" => CCAA

"門" => CBEE

"老" => CBED

"鏡" => CBEC

"攬" => CBEB

"元" => CBEA

"丑" => CBDE

"辛" => CBDD

"西" => CBDC

"鄂" => CBDB

"般" => CBDA

"毆" => CBCE

"拳" => CBCD

"奮" => CBCC

"己" => CBCB

"侮" => CBCA

"怒" => CBBE

"封" => CBBD

"士" => CBBC

"卒" => CBBB

"小" => CBBA

"薪" => CBAE

"榮" => CBAD

"盼" => CBAC

"窺" => CBAB

"耶" => CBAA

"寧" => CAEE

"謂" => CAED

"歸" => CAEC

"奇" => CAEB

"於" => CAEA

"租" => CADE

"逋" => CADD

"訟" => CADC

"貧" => CADB

"齊" => CADA

"去" => CACE

"肯" => CACD

"妻" => CACC

"晉" => CACB

"郵" => CACA

"督" => CABE

"望" => CABD

"學" => CABC

"希" => CABB

"鄧" => CABA

"止" => CAAE

"遇" => CAAD

"雄" => CAAC

"英" => CAAB

"古" => CAAA

Der Text:

古英雄未遇時，都無大志，非止鄧禹希文學、馬武望督郵也。晉文公有妻有馬，不肯去齊。光武貧時，與李通訟逋租於嚴尤。尤奇而目之。光武歸謂李通曰：「嚴公寧目君耶」窺其意，以得嚴君一盼為榮。韓蘄王為小卒時，相士言其日後封王。韓大怒，以為侮己，奮拳毆之。都是一般見解。鄂西林相公《辛丑元日》云：「攬鏡人將老，開門草未生。」《詠懷》云：「看來四十猶如此，便到百年已可知。」皆作郎中時詩也。玩其詞，若不料此後之出將入相者。及其為七省經略，《在金中丞席上》云：「問心都是酬恩客，屈指誰為濟世才」《登甲秀樓》絕句云：「炊煙卓午散輕絲，十萬人家飯熟時。問訊何年招濟火，斜陽滿樹武鄉祠。」居然以武侯自命，皆與未得志時氣象迥異。張桐城相公則自翰林至作首相，詩皆一格。最清妙者：「柳陰春水曲，花外暮山多。」「葉底花開人不見，一雙蝴蝶已先知。」「臨水種花知有意，一枝化作兩枝看。」《扈蹕》云：「誰憐七十龍鐘叟，騎馬踏冰星滿天」《和皇上〈風箏〉》云：「九霄日近增華色，四野風多仗寶繩。」押「繩」字韻，寄托遙深。 二 楊誠齋曰：「從來天分低拙之人，好談格調，而不解風趣。何也格調是空架子，有腔口易描；風趣專寫性靈，非天才不辦。」餘深愛其言。須知有性情，便有格律；格律不在性情外。《三百篇》半是勞人思婦率意言情之事；誰為之格，誰為之律而今之談格調者，能出其範圍否況皋、禹之歌，不同乎

《三百篇》;《國風》之格，不同乎《雅》、《頌》：格豈有一定哉許渾云：「吟詩好似成仙骨，骨里無詩莫浪吟。」詩在骨不在格也。

ist codiert:

CAAACAABCAACADBCAADDEDEAADCECABECACECADEAECAECAAECABAECDACAB
BECDBCABCADDADEBEACABDCABECACABCDDDCACBECDBBCEEBBCACCEBBADEE
AEBECACDCACECADADDECDCBEACADBDEDEAECDDECDEECEACADCCADDCADEC
AEAAEAECEBDDECEBCAEBAEBECECECBDDECDCBEACAECCAEDECDEECEAECEDEE
DABAEABCECAEEECECECEECBAAACCBABDEEAECEAAEDEDAAAEAECEEEBACBACEB
CCBADDDEDABCBAEEDACEBCCBBACBBBDEDEABEBCBBCAEEDEEBAAEDADCBBDE
DACDDEDABECACCBBEEAAEDEBCCBCACBCBEACBCCCBCDCBCEECBDDADCBDAE
BACBDAEDAEEDBADDCBDBCBDCEDBBBEBBCEEEECBDDCBDECBEABAAAAEBDEED
ABCBEBCBECBECEDBCCBEDEAEDBDCBEECCAAADBCCABDDACEEECCACCCADAAE
BDEEDABEDBEEDCAEDCBBABCCAECCBAEDCCEAEDCDCCBBBACEDCEEDDACCBC
BDBDDACBADBAECCBDEDDBDEDBEDBCDDDCCBEDEECCCAEACCCBEBECCCCEDC
CEDADECBEDDCEDBCCCCDBEBBBADDCCCEDEEEBCEDDDCCDACCDBCCDCEAEEE
BDCCCDDEDDBCCDECCEAEDDEAAEBDEEDABEDEACCEBADCBDACCECCCEDCCEE
EACDAACDABBDDEBCEDEBCDACEDECACEEECDADCDAECDBACDBBAACDBCCDBD
EBDEEDABCDBECDCACDCBCDCCCDCDCDCECDDAEABABCDDBBECCDDCCDDDC
DDEDEDDDEDEACDEAEDEDEDCECDEBEDEBCDECEACDEDCDEEEDEECEAABEACEA
BCEACDDACCEADCEAEAEDBEACEBAEEAACEBBEABADECDDADBEDAAECADDEDCEB
CCEBDCEBECECADDCECBCECCCECDBEBBCECECEEEAACEDAEDBBCEDBBAECEDC
BEBEABEDBADEBAECCDDCEDDCEDECEEABBAEEDABCEEBCEECCEEDEEABCEEEEA
BBBEEACDAAADAABEEADDDACABDAACDAADBBBEDBDBECEBEEDAEEAEBADAAEDA
BADABBEDDADABCBDBDDACABDABDEEABDABEBBBBDBEBBAECEAEBAEEAEDACAB
AEDACBEEAEEDBEDDACEEEDACCDACDAAEBDEEDABBDDDACEEDDDBABDADADAD
BDADCEADADDADEDADEDAEADAEBEDEEBBCACEEEDAECDAEDEDDEDAEEBEEDBAA
AAAAEBDEEDABDBABDBACBAADBADDBAEDBBADBBBEAEDCBDBBCBEEEEADDBBDD
BBEEEBADDACDBCAABEEBAACDBCBDBCCEADBCDDBCEDBDAEEBBDDEEBCDBDBE
EBCDBDCDBDDDBDEECEDEEDABDBEAEDCABBCDBEBDBECDBEDECBBECEAEEBDE
EBEECCBBDEAAEBEBEEDBABEEEECADDEDEDBCDECCBBDBDADBEEDCAADCABEAE
BBDCACDCADDCAEDCBABDEBEEEECADCBBDCBCBBEDCBDEAECAEBBCEDECEBED
CBEDDACDCCAEEBBDCCBDEEAEEDDDCCCBDBEBBBBEBCAEAEDCDEBBECCBCBB
DEECCBCBEBEBDCBBEBCAEEACDDEEEEECBBACEECCAADCCDBDADCCEBECDCDA
DCDBDCDCAECAEEBCAECBDCDDBDEBDDEBCECBECCEABDDEBCECBBCBAEBDCD
EECBEEBEECCBBDBBAEADCEAEDDCDEEDCEBDCECDCEDDCEEDEAAADDECDAECB
DEABEAEBEEECDEECEEEEEECBBACEECCAABDEEEEDEACBEEAAECBECCEAEBEEEC
DEECEEEEDEADAAADDEEEDEAEAAEEDECCDEBAEBBEBADEBBDEBCDEBDDEBEEBDE
EDABADABEDEEBDDECADECBDECCBCCEABCCDECDECABBEDDECEECAAADADDAC
BEDBDCBCCEBEBDCECCBCDDD

und ist 2125 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

A => 3 mm, B => 2 mm, C => 2 mm, D => 1 mm, E => 1 mm.

Damit ist der codierte Text 3615 mm lang

schmuck9.txt

Der japanische Text ist zu groß für dieses Dokument.

Output

Übersetzungstabelle:

Es gibt 4 Arten von Perlen: ['A', 'B', 'C', 'D'].

"の" => BCA

"た" => BBB

"に" => BBA

"い" => BAA

"し" => ADD

"と" => ADC

"を" => ADA

"て" => ACC

"は" => ACB

"、" => ACA

"な" => ABA

"。" => AAD

"が" => AAB

"る" => BDDD

"こ" => BDDB

"で" => BDDA

"か" => BDCC

"っ" => BDCB

"う" => BCDD

"ら" => BCDC

"れ" => BCDB

"す" => BCCD

"り" => BCBB

"ー" => BCBA

"き" => BBDC

"ま" => BBDA

"そ" => BBCB

"ン" => BADD

"く" => BADC

"だ" => BABB

"よ" => BABA

"さ" => ADBD

"あ" => ADBC

"も" => ADBB

"わ" => ACDA

"　" => ABDC

" " => ABDB

"・" => ABDA

"お" => ABCD

"」" => ABBD

"「" => ABBC

"け" => ABBB

"ル" => ABBA

"ん" => AAAB

"ち" => AAAA

"え" => BDDCD

"ス" => BDDCC

"つ" => BDDCB

"ラ" => BDDCA

"ト" => BDCDD

"見" => BDCDC

"イ" => BDCDB

"人" => BDCDA

"カ" => BDCAD

"レ" => BDABA

"ウ" => BDAAD

"ジ" => BDAAC

"や" => BDAAB

"ど" => BDAAA

"彼" => BCDAD

"聖" => BCDAC

"大" => BCDAB

"者" => BCDAA

"エ" => BCCCD

"上" => BCCCC

"め" => BCCCB

"思" => BCBCC

"ア" => BCBCB

"せ" => BCBCA

"出" => BBDDD

"フ" => BBDDC

"建" => BBDDB

"堂" => BBDDA

"車" => BBDBD

"サ" => BBDBC

"ば" => BBDBB

"手" => BBDBA

"タ" => BBCDD

"言" => BBCDC

"チ" => BBCDB

"み" => BBCDA

"物" => BBCCD

"事" => BBCCC

"ョ" => BBCCB

"道" => BBCCA

"む" => BBCAD

"中" => BBCAC

"海" => BBCAB

"げ" => BABDC

"ク" => BABDB

"べ" => BABDA

"会" => BABCD

"口" => BABCC

"一" => BABCB

"地" => BABCA

"司" => ADBAD

"任" => ADBAC

"主" => ADBAB

"ね" => ADBAA

"入" => ACDDD

"ょ" => ACDDC

"通" => ACDDB

"雨" => ACDDA

"乗" => ACDCD

"築" => ACDCC

"水" => ACDCB

"立" => ACDCA

"び" => ACDBD

"ろ" => ACDBC

"オ" => ACDBB

"ほ" => ACDBA

"マ" => ABDDD

"部" => ABDDC

"祭" => ABDDB

"気" => ABDDA

"分" => ABCCD

"一" => ABCCC

"合" => ABCCB

"家" => ABCCA

"ぶ" => ABCBD

"ド" => ABCBC

"ア" => ABCBB

"員" => ABCBA

"馬" => ABCAD

"自" => ABCAC

"決" => ABCAB

"川" => ABCAA

"場" => AACDD

"不" => AACDB

"重" => AACDA

"奏" => AACCD

"友" => AACCC

"ぎ" => AACCB

"前" => AACCA

"降" => AACBD

"駅" => AACBC

"目" => AACBB

"ご" => AACBA

"行" => AACAD

"明" => AACAC

"利" => AACAB

"ッ" => AACAA

"取" => AAADD

"市" => AAADC

"ざ" => AAADB

"港" => AAADA

"後" => AAACD

"送" => AAACC

"内" => AAACB

"線" => AAACA

"塔" => BDADCD

"素" => BDADCC

"安" => BDADCB

"差" => BDADCA

"リ" => BDADBD

"医" => BDADBC

"ガ" => BDADBB

"ノ" => BDADBA

"ミ" => BDADAD

"教" => BDADAC

"介" => BDADAB

"紹" => BDADAA

"意" => BDACDD

"若" => BDACDC

"ひ" => BDACDB

"ぼ" => BDACDA

"側" => BDACCD

"実" => BDACCC

"数" => BDACCB

"用" => BDACCA

"着" => BDACBD

"ワ" => BDACBC

"過" => BDACBB

"敷" => BDACBA

"式" => BDACAD

"屋" => BDACAC

"運" => BDACAB

"年" => BDACAA

"交" => BDABDD

"外" => BDABDC

"鉄" => BDABDB

"広" => BDABDA

"方" => BDABCD

"振" => BDABCC

"ぐ" => BDABCB

"防" => BDABCA

"身" => BDABBD

"得" => BDABBC

"四" => BDABBB

"陸" => BDABBA

"眠" => BCCCAD

"晴" => BCCCAC

"廊" => BCCCAB

"特" => BCCCAA

"ゃ" => BCCBDD

"指" => BCCBDC

"づ" => BCCBDB

"示" => BCCBDA

"仕" => BCCBCD

"助" => BCCBCC

"格" => BCCBCB

"話" => BCCBCA

"空" => BCCBBD

"巨" => BCCBBC

"散" => BCCBBB

"少" => BCCBBA

"ぞ" => BCCBAD

"歩" => BCCBAC

"じ" => BCCBAB

"パ" => BCCBAA

"音" => BCCADD

"足" => BCCADC

"荷" => BCCADB

"到" => BCCADA

"ず" => BCCACD

"商" => BCCACC

"修" => BCCACB

"説" => BCCACA

"古" => BCCABD

"保" => BCCABC

"復" => BCCABB

"走" => BCCABA

"北" => BCCAAD

"小" => BCCAAC

"々" => BCCAAB

"町" => BCCAAA

"舞" => BCBDDD

"越" => BCBDDC

"南" => BCBDDB

"味" => BCBDDA

"草" => BCBDCD

"牧" => BCBDCC

"堤" => BCBDCB

"石" => BCBDCA

"考" => BCBDBD

"代" => BCBDBC

"生" => BCBDBB

"民" => BCBDBA

"他" => BCBDAD

"河" => BCBDAC

"名" => BCBDAB

"撃" => BCBDAA

"ダ" => BCBCDD

"船" => BCBCDC

"無" => BCBCDB

"量" => BCBCDA

"山" => BBCAAD

"積" => BBCAAC

"様" => BBCAAB

"丈" => BBCAAA

"べ" => BADBDD

"注" => BADBDC

"匂" => BADBDB

"濡" => BADBDA

"役" => BADBCD

"何" => BADBCC

"達" => BADBCB

"対" => BADBCA

"度" => BADBBD

"ぬ" => BADBBC

"楽" => BADBBB

"シ" => BADBBA

"下" => BADBAD

"区" => BADBAC

"尊" => BADBAB

"キ" => BADBAA

"参" => BADADD

"頷" => BADADC

"以" => BADADB

"多" => BADADA

"直" => BADACD

"職" => BADACC

"暗" => BADACB

"套" => BADACA

"深" => BADABD

"ポ" => BADABC

"神" => BADABB

"根" => BADABA

"進" => BADAAD

"ヴ" => BADAAC

"ピ" => BADAAB

"感" => BADAAA

"面" => BACDDD

"頭" => BACDDC

"全" => BACDDB

"十" => BACDDA

"覆" => BACDCD

"床" => BACDCC

"テ" => BACDCB

"ホ" => BACDCA

"ム" => BACDBD

"ブ" => BACDBC

"老" => BACDBB

"心" => BACDBA

"厳" => BACDAD

"買" => BACDAC

"券" => BACDAB

"等" => BACDAA

"三" => BACCDD

"午" => BACCDC

"汽" => BACCDB

"時" => BACCDA

"委" => BACCCD

"工" => BACCCC

"支" => BACCCB

"社" => BACCCA

"便" => BACCBD

"団" => BACCBC

"集" => BACCBB

"望" => BACCBA

"信" => BACCAD

"能" => BACCAC

"有" => BACCAB

"都" => BACCAA

"治" => BACBDD

"街" => BACBDC

"輪" => BACBDB

"グ" => BACBDA

"境" => BACBCD

"動" => BACBCC

"窓" => BACBCB

"ゆ" => BACBCA

"日" => BACBBD

"儀" => BACBBC

"波" => BACBBB

"風" => BACBBA

"美" => BACBAD

"負" => BACBAC

"持" => BACBAB

"権" => BACBAA

"羊" => BACADD

"放" => BACADC

"知" => BACADB

"伸" => BACADA

"横" => BACACD

"細" => BACACC

"流" => BACACB

"砂" => BACACA

"由" => BACABD

"世" => BACABC

"隻" => BACABB

"隊" => BACABA

"寄" => BACAAD

"二" => BACAAC

"岸" => BACAAB

"今" => BACAAA

"所" => BABDDD

"呼" => BABDDC

"元" => BABDDB

"ォ" => BABDDA

"図" => AACDCD

"拭" => AACDCC

"ペ" => AACDCB

"ギ" => AACDCA

"登" => BDCACDD

"天" => BDCACDC

"関" => BDCACDB

"定" => BDCACDA

"ぜ" => BDCACCD

"応" => BDCACCC

"連" => BDCACCB

"胆" => BDCACCA

"！" => BDCACBD

"載" => BDCACBC

"背" => BDCACBB

"形" => BDCACBA

"円" => BDCACAD

"半" => BDCACAC

"諺" => BDCACAB

"印" => BDCACAA

"刻" => BDCABDD

"妻" => BDCABDC

"稲" => BDCABDB

"割" => BDCABDA

"枠" => BDCABCD

"筋" => BDCABCC

"跡" => BDCABCB

"詰" => BDCABCA

"瓦" => BDCABBD

"煉" => BDCABBC

"裂" => BDCABBB

"亀" => BDCABBA

"抱" => BDCABAD

"莫" => BDCABAC

"構" => BDCABAB

"架" => BDCABAA

"頑" => BDCAADD

"岩" => BDCAADC

"軽" => BDCAADB

"値" => BDCAADA

"吟" => BDCAACD

"ケ" => BDCAACC

"簡" => BDCAACB

"高" => BDCAACA

"崇" => BDCAABD

"毛" => BDCAABC

"眉" => BDCAABB

"答" => BDCAABA

"夫" => BDCAAAD

"導" => BDCAAAC

"尋" => BDCAAAB

"脇" => BDCAAAA

"験" => BDBDDDD

"経" => BDBDDDC

"五" => BDBDDDB

"百" => BDBDDDA

"千" => BDBDDCD

"案" => BDBDDCC

"弁" => BDBDDCB

"雄" => BDBDDCA

"漏" => BDBDDBD

"冷" => BDBDDBC

"醸" => BDBDDBB

"囲" => BDBDDBA

"雰" => BDBDDAD

"独" => BDBDDAC

"傘" => BDBDDAB

"念" => BDBDDAA

"観" => BDBDCDD

"逃" => BDBDCDC

"褒" => BDBDCDB

"快" => BDBDCDA

"折" => BDBDCCD

"摘" => BDBDCCC

"徴" => BDBDCCB

"嫌" => BDBDCCA

"機" => BDBDCBD

"料" => BDBDCBC

"務" => BDBDCBB

"護" => BDBDCBA

"惜" => BDBDCAD

"虚" => BDBDCAC

"謙" => BDBDCAB

"寧" => BDBDCAA

"丁" => BDBDBDD

"揖" => BDBDBDC

"恭" => BDBDBDB

"識" => BDBDBDA

"認" => BDBDBCD

"充" => BDBDBCC

"逆" => BDBDBCB

"万" => BDBDBCA

"蔑" => BDBDBBD

"侮" => BDBDBBC

"態" => BDBDBBB

"肩" => BDBDBBA

"受" => BDBDBAD

"伝" => BDBDBAC

"理" => BDBDBAB

"管" => BDBDBAA

"売" => BDBDADD

"秀" => BDBDADC

"優" => BDBDADB

"二" => BDBDADA

"演" => BDBDACD

"拝" => BDBDACC

"礼" => BDBDACB

"ヤ" => BDBDACA

"問" => BDBDABD

"疑" => BDBDABC

"貶" => BDBDABB

"扱" => BDBDABA

"相" => BDBDAAD

"調" => BDBDAAC

"敬" => BDBDAAB

"短" => BDBDAAA

"回" => BDBCDDD

"誇" => BDBCDDC

"業" => BDBCDDB

"泊" => BDBCDDA

"付" => BDBCDCD

"別" => BDBCDCC

"聞" => BDBCDCB

"申" => BDBCDCA

"己" => BDBCDBD

"同" => BDBCDBC

"使" => BDBCDBB

"彩" => BDBCDBA

"拶" => BDBCDAD

"挨" => BDBCDAC

"巻" => BDBCDAB

"首" => BDBCDAA

"袖" => BDBCCDD

"領" => BDBCCDC

"返" => BDBCCDB

"男" => BDBCCDA

"席" => BDBCCCD

"歌" => BDBCCCC

"薄" => BDBCCCB

"閉" => BDBCCCA

"雲" => BDBCCBD

"押" => BDBCCBC

"》" => BDBCCBB

"《" => BDBCCBA

"帳" => BDBCCAD

"革" => BDBCCAC

"戸" => BDBCCAB

"扉" => BDBCCAA

"撥" => BDBCBDD

"服" => BDBCBDC

"急" => BDBCBDB

"墓" => BDBCBDA

"飛" => BDBCBCD

"襟" => BDBCBCC

"子" => BDBCBCB

"帽" => BDBCBCA

"届" => BDBCBBD

"開" => BDBCBBC

"御" => BDBCBBB

"板" => BDBCBBA

"拔" => BDBCBAD

"停" => BDBCBAC

"門" => BDBCBAB

"荘" => BDBCBAA

"的" => BDBCADD

"秘" => BDBCADC

"遥" => BDBCADB

"姿" => BDBCADA

"描" => BDBCACD

"像" => BDBCACC

"想" => BDBCACB

"浮" => BDBCACA

"郭" => BDBCABD

"幕" => BDBCABC

"紗" => BDBCABB

"台" => BDBCABA

"被" => BDBCAAD

"エ" => BDBCAAC

"蒸" => BDBCAAB

"昇" => BDBCAAA

"霧" => BDBBDDD

"砕" => BDBBDDC

"然" => BDBBDDB

"沛" => BDBBDDA

"待" => BDBBDCD

"ゴ" => BDBBDCC

"ィ" => BDBBDCB

"デ" => BDBBDCA

"幾" => BDBBDBD

"緑" => BDBBDBC

"払" => BDBBDBB

"追" => BDBBDBA

"篠" => BDBBDAD

"抑" => BDBBDAC

"声" => BDBBDAB

"嘆" => BDBBDAA

"セ" => BDBBCDD

"聳" => BDBBCDC

"角" => BDBBCDB

"駆" => BDBBCDA

"完" => BDBBCCD

"耐" => BDBBCCC

"間" => BDBBCCB

"突" => BDBBCCA

"藁" => BDBBCBD

"ぷ" => BDBBCBC

"裕" => BDBBCBB

"余" => BDBBCBA

"客" => BDBBCAD

"好" => BDBBCAC

"喜" => BDBBCAB

"族" => BDBBCAA

"ズ" => BDBBBDD

"属" => BDBBBDC

"配" => BDBBBDB

"院" => BDBBBDA

"養" => BDBBBCD

"終" => BDBBBCC

"労" => BDBBBCB

"苦" => BDBBBCA

"威" => BDBBBBD

"換" => BDBBBBC

"約" => BDBBBBB

"節" => BDBBBBA

"費" => BDBBBAD

"旅" => BDBBBAC

"ロ" => BDBBBAB

"激" => BDBBBAA

"毎" => BDBBADD

"消" => BDBBADC

"珍" => BDBBADB

"当" => BDBBADA

"託" => BDBBACD

"末" => BDBBACC

"粗" => BDBBACB

"比" => BDBBACA

"去" => BDBBABD

"性" => BDBBABC

"設" => BDBBABB

"要" => BDBBABA

"必" => BDBBAAD

"善" => BDBBAAC

"改" => BDBBAAB

"織" => BDBBAAA

"組" => BDBADDD

"正" => BDBADDC

"表" => BDBADDB

"体" => BDBADDA

"ゼ" => BDBADCD

"議" => BDBADCC

"選" => BDBADCB

"往" => BDBADCA

"鄙" => BDBADBD

"長" => BDBADBC

"七" => BDBADBB

"捨" => BDBADBA

"幹" => BDBADAD

"ザ" => BDBADAC

"幅" => BDBADAB

"界" => BDBADAA

"湖" => BDBACDD

"浸" => BDBACDC

"移" => BDBACDB

"再" => BDBACDA

"誰" => BDBACCD

"眺" => BDBACCC

"階" => BDBACCB

"暴" => BDBACCA

"切" => BDBACBD

"断" => BDBACBC

"束" => BDBACBB

"拘" => BDBACBA

"昔" => BDBACAD

"忘" => BDBACAC

"共" => BDBACAB

"潮" => BDBACAA

"東" => BDBABDD

"渡" => BDBABDC

"抗" => BDBABDB

"抵" => BDBABDA

"産" => BDBABCD

"プ" => BDBABCC

"峡" => BDBABCB

"低" => BDBABCA

"造" => BDBABBD

"路" => BDBABBC

"真" => BDBABBB

"償" => BDBABBA

"埋" => BDBABAD

"沢" => BDBABAC

"塩" => BDBABAB

"計" => BDBABAA

"類" => BDBAADD

"貌" => BDBAADC

"変" => BDBAADB

"縮" => BDBAADA

"縦" => BDBAACD

"へ" => BDBAACC

"探" => BDBAACB

"易" => BDBAACA

"貿" => BDBAABD

"州" => BDBAABC

"ふ" => BDBAABB

"泥" => BDBAABA

"沈" => BDBAAAD

"域" => BDBAAAC

"残" => BDBAAAB

"輝" => BDBAAAA

"史" => BDADDDD

"歴" => BDADDDC

"緒" => BDADDDB

"迎" => BDADDDA

"攻" => BDADDCD

"紀" => BDADDCC

"六" => BDADDCB

"戦" => BDADDCA

"艦" => BDADDBD

"敵" => BDADDBC

"単" => BDADDBB

"作" => BDADDBA

"制" => BDADDAD

"測" => BDADDAC

"国" => BDADDAB

"英" => BDADDAA

Der Text:

英国陸地測量部制作の地図ではカラン・ウォ―フ、地元の人には単にカランと呼ば ... gekürzt...

にオッサ山を積み重ねたような気がしてならず、交差部の巨大
BDADDAABDADDABBDABBABABCABDADDACBCBCDAABDDCBDADDADBDADDBA...gekürzt...

ACDBBACCADBADAAAACBAACDCCBAABABABCDDADBBABABDCCBDCBBBBAAD

und ist 19771 Perlen lang.

Aufgabe B:

In der Codierung sind die Perlen so:

C => 4 mm, D => 3 mm, A => 2 mm, B => 1 mm.

Damit ist der codierte Text 45129 mm lang