Bundeswettbewerb Informatik: Aufgabe 2

Lösungsidee

Es gibt 8 Tasten, die mit 26 Buchstaben belegt werden müssen. Von der Handytastatur werden nur die Tasten 2 – 9 betrachtet. Mindestens ein Buchstabe muss immer auf einer Taste sein, da schon eine Taste ohne Buchstaben die Kosten einer Tastenbelegung steigert. Es ergibt sich somit, dass maximal 19 Buchstaben auf einer Taste setzbar sind.

Meine Lösungmethode ist die Brute-Force-Methode: Um eine optimale Tastenbelegung anhand von Buchstabenhäufigkeiten zu erhalten gehe ich alle Möglichkeiten, Buchstaben auf einer Handytastatur zu positionieren durch, berechne für jede erhaltene Handytastatur die Gesamtkosten und merke mir eine Handytastatur mit den geringsten Gesamtkosten.

Gibt es mehrere optimale Tastaturbelegungen mit den selben Kosten, so wird trotzdem nur eine ausgegeben.

Zur Kostenberechnung:

Die Kosten einer Tastenbelegung ergeben sich durch die Häufigkeitswerte der einzelnen Buchstaben und ihrer Position auf einer Taste. Wenn beispielsweise ein Buchstabe an zweiter Stelle auf einer Taste steht, dann zählt der Häufigkeitswert des Buchstabens doppelt, da die Taste zweimal gedrückt werden muss um den Buchstaben auszuwählen. Formal lassen sich die Kosten einer Tastenbelegung also wie folgt berechnen:

$$K = \sum_{t=1}^{T} \sum_{p=1}^{P_t} p \cdot h(\textit{Buchstabe an Position } p \textit{ auf Taste } t)$$

Formel zur Berechnung der Kosten einer Tastenbelegung

K: Kosten einer Tastenbelegung
T: Anzahl der Tasten
P_t: Anzahl der Buchstaben auf der Taste t
h(b): Häufigkeitswert des Buchstaben b

Zum Brute-Force-Algorithmus:

Mein Brute-Force-Algorithmus findet alle Möglichkeiten, wie man das Alphabet in 8 Abschnitte einteilen kann. Dazu ein Beispiel:

	ABC	DEFG	HI	JKLM	NO	PQ	RSTUV	WXYZ
	1	2	3	4	5	6	7	8

Die Handytastatur würde so aussehen:

1	2 ABC Taste 1	3 DEFG Taste 2
4 HI Taste 3	5 JKLM Taste 4	6 NO Taste 5
7 PQ Taste 6	8 RSTUV Taste 7	9 WXYZ Taste 8
*	0	#

Jeder Abschnitt steht symbolisch für eine Taste: Abschnitt 1 steht demnach für Taste 1, Abschnitt 2 für Taste 2, usw. Die Abschnitte kann man auch anders notieren:

	3	4	2	4	2	2	5	4

Diese Darstellung bedeutet nichts anderes, als dass sich auf der Taste 1 drei Buchstaben (ABC) des Alphabets befinden, auf der Taste 2 die nächsten vier Buchstaben (DEFG), usw.

Die erste Möglichkeit die Buchstaben auf 8 Tasten zu platzieren ist dabei wie folgt festgelegt:

	1	1	1	1	1	1	1	19

Das heißt, dass sich auf den Tasten 1 bis 7 jeweils nur ein Buchstabe befindet und auf der Taste 8 die übrigen Buchstaben (H – Z) platziert sind.

Die nächste Möglichkeit erhält man, indem man den Wert des letzten Elements dekrementiert und den Wert des vorletzten Elements erhöht:

	1	1	1	1	1	1	**2**	**18**

Nach einigen Wiederholungen dieser Regel beträgt der Wert des letzten Elements 0. Dieser Zustand wird als Überlauf bezeichnet und muss speziell behandelt werden, da diese Kombination keine gültige Tastenbelegung darstellt:

	1	1	1	1	1	1	**20**	**0**

Das Element mit dem Überlauf ist dabei jedoch nicht das letzte Element sondern das vorletzte Element. Bei dieser Tastenbelegung wurden bereits alle 26 Buchstaben auf 7 Tasten gelegt und es ist daher nicht mehr möglich einen Buchstaben auf die letzte Taste zu platzieren. Um dieses Problem zu beheben und den Überlauf zu behandeln wird der Wert des Elements vor dem Überlauf-

Element erhöht. Das Überlauf-Element wird wieder mit 1 initialisiert. Anschliessend wird der Wert des letzten Elements mit folgender Formel berechnet:

$$E_T = B - \sum_{n=1}^{T-1} E_n \quad \Rightarrow \quad E_8 = 26 - \sum_{n=1}^{7} E_n \quad \Rightarrow \quad E_8 = 26 - (E_1 + E_2 + E_3 + E_4 + E_5 + E_6 + E_7)$$

Formel zur Berechnung des Wertes des letzten Elements (nach Behandlung eines Überlaufs)

E_i: Wert des Elements i
B: Anzahl Buchstaben
T: Anzahl der Tasten

1	1	1	1	1	1	2	1	**18**

Weitere Überläufe müssen behandelt werden solange der berechnete Wert des letzten Elements 0 ergibt. Der Überlauf wandert dabei immer auf das vorherige Element:

1	1	1	1	1	19	1	1	→ gültige Kombination
1	1	1	1	1	19	**2**	**0**	→ 1. Überlauf tritt auf (Wert des letzten Elements ist 0)
1	1	1	1	1	**20**	**1**	***0***	→ 1. Überlauf behandelt; Wert des letzten Elements berechnet
1	1	1	1	**2**	**1**	1	***0***	→ 2. Überlauf behandelt, es folgt Berechnung des letzten Elements
1	1	1	1	2	1	1	**18**	→ Wert des letzten Elements berechnet; gültige Kombination
1	1	1	1	2	1	**2**	**17**	→ nächste gültige Kombination

Tritt ein Überlauf beim ersten Element auf, gibt es keine weiteren Kombinationen und der Algorithmus terminiert. Da für jede gültige Kombinationen die Kosten berechnet werden und immer die Kombination mit den geringsten Kosten (in Form einer Tastenbelegung) gemerkt wird, kann auf diese Weise eine Tastenbelegung mit optimalen Kosten ermittelt werden.

Programm-Dokumentation

Die Lösungsidee ist von mir in der Programmiersprache C umgesetzt.

Mein Programm benötigt als Parameter den Dateinamen der Datei, in der die einzelnen Buchstabenhäufigkeiten der Sprache abgespeichert sind.

Als Grundlage werden die Strukturen **`Tastatur`** und **`Taste`**, sowie das Häufigkeitenarray **`Haeufigkeiten`** benötigt. In dem Häufigkeitenarray werden die Häufigkeitswerte der einzelnen Buchstaben gespeichert. Die Struktur **`Tastatur`** besteht aus einem Array der Struktur T**`aste`** mit der Länge 8 und enthält zudem eine **`Integer`**-Variable, in der die Kosten der Tastenbelegung gespeichert werden. Die Struktur **`Taste`** enthält einen **`String`** auf dem maximal 19 Buchstaben „abgelegt“ werden können.

In der **`main`**-Funktion werden zuerst die Häufigkeitswerte der einzelnen Buchstaben aus der angegebenen Datei ausgelesen und in das Häufigkeitenarray gespeichert. Danach wird die Funktion **`finde_optimale_tastatur`** aufgerufen, welche als Parameter das Häufigkeitenarray übergeben bekommt. Diese Funktion geht alle Möglichkeiten, Buchstaben auf einer Tastatur zu platzieren, durch und merkt sich eine Tastenbelegung mit den geringsten Kosten. Eine optimale Tastenbelegung wird an die **`main`**-Funktion zurückgegeben, wo sie schließlich ausgegeben wird.

In der Funktion **`finde_optimale_tastenbelegung`** werden die Werte der Elemente einer Tastaturbelegung in einem **`Integer`**-Array gespeichert (Wert-Array), welches mit der ersten Kombination **`{1, 1, 1, 1, 1, 1, 1, 19}`** initialisiert wird.

Dann beginnt eine Schleife, die den eigentlichen Algorithmus enthält: Zu Beginn jedes Schleifendurchlaufs konvertiert die Funktion **`erstelle_tastaturbelegung`** aus einem Wert-Array eine Tastaturbelegung (Struktur **`Tastatur`**) und gibt einen Zeiger auf diese zurück.

Dann wird die nächste Kombination für das Wert-Array ermittelt und eventuelle Überläufe behandelt.

Ist keine weitere Kombination mehr vorhanden, wird die bis dahin gefunde optimale Tastaturbelegung zurückgegeben.

Programm-Ablaufprotokoll

Beim Aufruf des Programms ohne Parameter erscheint eine Fehlermeldung und dem Benutzer wird mitgeteilt, dass das Programm einen Dateinamen als Parameter benötigt.

Aus Platzgründen sollen hier nur einige Aufrufe des Programms abgedruckt werden.

Mein Programm wird mit den Buchstabenhäufigkeiten im Deutschen aufgerufen:

```
$ ./aufgabe2 deutsch.txt
optimale Tastenbelegung fuer deutsch.txt:
Taste 2: AB
Taste 3: CD
Taste 4: EFG
Taste 5: HIJK
Taste 6: LM
Taste 7: NOPQ
Taste 8: RS
Taste 9: TUVWXYZ
Kosten der Tastenbelegung: 158780
```

Zur Kontrolle der Musterlösung ist hier der Aufruf des Programms mit den englischen Buchstabenhäufigkeiten:

```
optimale Tastenbelegung fuer englisch.txt:
Taste 2: AB
Taste 3: CD
Taste 4: EFG
Taste 5: HIJK
Taste 6: LM
Taste 7: NOPQ
Taste 8: RS
Taste 9: TUVWXYZ
Kosten der Tastenbelegung: 164682
```

Mein Programm ermittelt für polnisch folgende optimale Tastenbelegung:

```
$ ./aufgabe2 polnisch.txt
optimale Tastenbelegung fuer polnisch.txt:
Taste 2: ABCD
Taste 3: EFGH
Taste 4: IJ
Taste 5: KLM
Taste 6: NOPQ
Taste 7: RSTUV
Taste 8: WX
Taste 9: YZ
Kosten der Tastenbelegung: 178678
```

Für finnisch:

```
./aufgabe2 finnisch.txt
optimale Tastenbelegung fuer finnisch.txt:
Taste 2: ABCD
Taste 3: EFGH
Taste 4: IJ
Taste 5: KLM
Taste 6: NOPQ
Taste 7: RS
Taste 8: TUVWX
Taste 9: YZ
Kosten der Tastenbelegung: 154931
```

Sehr geringe Gesamtkosten gibt es bei der italienischen Tastenbelegung:

```
./aufgabe2 italienisch.txt
optimale Tastenbelegung fuer italienisch.txt:
Taste 2: AB
Taste 3: CD
Taste 4: EFGH
Taste 5: IJK
Taste 6: LM
Taste 7: NOPQ
Taste 8: RS
Taste 9: TUVWXYZ
Kosten der Tastenbelegung: 148640
```

Ruft man das Programm ohne oder mit mehr als einen Parameter auf erscheint folgende Fehlermeldung:

```
$ ./aufgabe2
Bitte den Dateinamen der Datei mit den Haeufigkeitswerten als
Parameter angeben!
```