

9

Verhaltens-Algorithmen

9.1 Teile und Herrsche

Beispiel: Suchen nach der Bisektionsmethode

Will man die Suchliste pflegen, dann kommen zwei weitere Prozeduren hinzu, nämlich das Einfügen und Entfernen von Einträgen. Dazu wird die vorhandene Suchprozedur etwas umgestaltet. Der eigentliche Suchprozess wird als eigenständige Prozedur ausgelagert. So kann sie auch von den anderen Prozeduren genutzt werden.

Code 9-1 Liste nach der Bisektionsmethode verwalten

```
Option Explicit

Sub SuchListe()
    Dim MyDoc As Object
    Dim Text As String
    Dim s As Long

    Set MyDoc = ThisWorkbook.Worksheets("Suchliste")

    'Eingabe
    Text = InputBox("Suchbegriff angeben!")

    Call Begriff_Suche(Text, s)

    'Ausgabe
    Cells(s, 1).Activate
    MsgBox "Zeile:" & Str(s) & vbCrLf & Cells(s, 1), vbOKOnly,
        "Suchergebnis von: " & Text
End Sub

Sub Begriff_Suche(Text, s)
    Dim MyDoc As Object
    Dim v, b, m As Long

    Set MyDoc = ThisWorkbook.Worksheets("Suchliste")
```

```
'Start-Suchbereich
'von 1 bis Anzahl Einträge
v = 1
b = MyDoc.UsedRange.Rows.Count

'Suchschleife
Do While b >= v
  m = Int((v + b) / 2)
  If Text > Cells(m, 1) Then
    v = m + 1
  Else
    b = m - 1
  End If
  s = v
Loop
End Sub

Sub Begriff_Einfügen()
  Dim Text, z As String
  Dim i, s As Long
  Dim MyDoc As Object

  Set MyDoc = ThisWorkbook.Worksheets("Suchliste")

  'Eingabe
  Text = InputBox("Neuen Suchbegriff eingeben!")
  Call Begriff_Suche(Text, s)
  If Not Text = "" And s > 0 Then
    z = LTrim(Str(s))
    Rows(z & ":" & z).Select
    If MsgBox(Text & " in Zeile" & Str(s) & " einfügen?",
      vbYesNo) = vbYes Then
      Selection.Insert Shift:=xlDown
      Cells(s, 1) = Text
    End If
  End If
End Sub

Sub Begriff_Löschen()
  Dim Text, z As String
  Dim s As Long
  Dim MyDoc As Object

  Set MyDoc = ThisWorkbook.Worksheets("Suchliste")

  'Eingabe
```

```

Text = InputBox("Zu löschenden Suchbegriff eingeben!")
Call Begriff_Suche(Text, s)
If InStr(Cells(s, 1), Text) > 0 Then
  z = LTrim(Str(s))
  Rows(z & ":" & z).Select
  If MsgBox(Text & " in Zeile" & Str(s) & " löschen?",
    vbYesNo) = vbYes Then
    Selection.Delete Shift:=xlUp
  End If
End If
End Sub

```

Beispiel: Suche in Hashlisten

Eine sehr effiziente Methode, in Listen Informationen zu finden, ist die Hash-Methode. Die Methode wurde in den fünfziger Jahren bei IBM unter dem Begriff gestreute Speicherung entwickelt.

Ordnungsschlüssel

↓ f: Hashfunktion

Satz-Nr.



Abb. 9-1 Hashfunktionen

Die Grundidee - Elemente einer Liste oder Datei werden durch ganzzahlige Schlüssel gekennzeichnet und mit Hilfe einer Funktion (Hashfunktion) einer Adresse (Hashindex) zugeordnet.

Hash-Funktionen sind im Allgemeinen nicht eindeutig (injektiv). Das heißt, zwei unterschiedliche Schlüssel können denselben

Hash-Wert erzeugen. Dieses Ereignis wird als Kollision bezeichnet. In diesem Fall muss die Hashtabelle mehrere Werte an derselben Stelle aufnehmen. Um dieses Problem zu handhaben, gibt es diverse Kollisionsauflösungsstrategien.

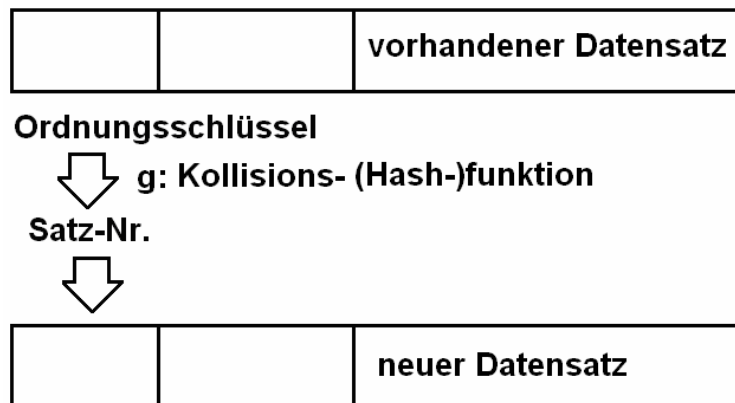


Abb. 9-2 Kollision

Eine Möglichkeit heißt Offenes Hashing. Trifft dabei ein Eintrag auf eine schon belegte Stelle, wird stattdessen eine andere freie Stelle genommen. Diese Stelle wird meistens nach folgender Vorgehensweise gefunden:

1. Lineares Sondieren
Dabei wird um ein konstantes Intervall verschoben nach einer freien Stelle gesucht. Die Intervallgröße ist oft 1.
2. Quadratisches Sondieren
Nach jedem erfolglosen Suchschritt wird das Intervall quadriert.
3. Doppeltes Hashen
Eine weitere Hash-Funktion liefert das Intervall.

Eine andere Strategie ist die Kollisionsauflösung durch Verkettung. Anstelle der gesuchten Daten enthält die Hashliste Behälter (Buckets), die alle Daten mit gleichem Hash-Wert aufnehmen.

Bei einer Suche wird also zunächst innerhalb der Hashliste gesucht und danach innerhalb des Behälters. Der extremste Fall (Worst Case) liegt dann vor, wenn alle Elemente gleiche Hash-Werte haben und damit in den gleichen Buckets abgelegt werden. Wichtig ist daher für die Praxis die richtige Hash-Funktion, die eine gleichmäßige Verteilung in der Hashliste erzeugt.

Hashfunktionen sind eine Wissenschaft für sich. Möglichst wenige Kollisionen erhält man, wenn man als Hashfunktion die Divisionsreste bei Teilung durch eine Primzahl p wählt, da die Reste mod p stark streuen.

Für unser Beispiel wählen wir die ersten fünf Buchstaben im ASCII-Code, verknüpfen sie miteinander und bestimmen den Primzahlrest. Weitere Funktionen sind die multiplikative Methode, die Zerlegungsmethode, die Mittquadratmethode, die Quersummenmethode, um nur einige zu nennen.

Ausgangspunkt ist die vorhandene Inhaltsliste. Über die Hashfunktion erzeugen wir eine neue Liste.

Tab. 9-1 Speicherung mittels Hashfunktion

$p=200$
Max = Anzahl Einträge in der Inhaltsliste
$i = 1, 1, \text{Max}$
$f = \left(\sum_{j=1}^5 b_j \right) \text{mod}(p)$

		Ist Zelle f leer	
		Ja	Nein
Speicher Zelle f	Begriff in		Kollisionsstrategie

Tab. 9-2 Kollisionsstrategie

g=201
Solange Zelle g nicht leer
g=g+1
Speicher Begriff in Zelle g

In der Mappe mit der Suchliste erstellen wir eine weitere Tabelle mit dem Namen Hashliste. Diese Tabelle erhält die nachfolgenden Prozeduren.

Code 9-2 Liste speichern mit Hashfunktion und Kollisionsstrategie

```
Option Explicit

Sub SuchListe()
    Dim MyDoc1, MyDoc2 As Object
    Dim Text As String
    Dim f, i, j, z, a As Long

    Set MyDoc1 = ThisWorkbook.Worksheets("Suchliste")
    Set MyDoc2 = ThisWorkbook.Worksheets("Hashliste")
    MyDoc2.Cells.Clear

    z = MyDoc1.UsedRange.Rows.Count - 1

    For i = 1 To z
        Text = MyDoc1.Cells(i, 1)
        a = 0
        For j = 1 To 5
            a = a + Asc(Mid(Text, j, 1))
        
```

```

Next j
f = a Mod 200
If MyDoc2.Cells(f, 1) = "" Then
    MyDoc2.Cells(f, 1) = Text
Else
    Call Kollision(Text)
End If
Next i
End Sub

Sub Kollision(Text)
Dim MyDoc2 As Object
Dim g As LoadPictureConstants

Set MyDoc2 = ThisWorkbook.Worksheets("Hashliste")

g = 201
Do While Not MyDoc2.Cells(g, 1) = ""
    g = g + 1
Loop
MyDoc2.Cells(g, 1) = Text
End Sub

```

Wie zu erwarten war, liefert eine derart einfache Hashfunktion jede Menge Kollisionen. Ich überlasse es dem Leser, hier weiter zu optimieren. Sowohl bei der Hashfunktion als auch bei der Kollisionstrategie. Natürlich gehören zur Verwaltung einer Hashliste auch Lese-, Lösch- und EingabeprozEDUREN.

9.2 Die Greedy-Methode

Beispiel: Auftragsfolgenproblem

Übungen

Wählen Sie statt des Kriteriums

$$\frac{\text{Wert}}{\text{Belegzeit}}$$

und vergleichen Sie beide Ergebnisse.

Dazu wird bei der Datenübernahme eine zusätzliche Anweisung ergänzt.

Code 9-3 Ergänzung der Datenübernahme

Option Explicit

```
'Daten übernehmen
For i = 1 To n
  a(i, 2) = Cells(i + 1, 2)
  a(i, 3) = Cells(i + 1, 3)
  a(i, 4) = Cells(i + 1, 4)
  a(i, 2) = a(i, 2) / a(i, 3)
Next i
```

Das Ergebnis ist in der Tat ein anderes.

	A	B	C	D	E	F	G
			<i>Beleg. Zeit [h]</i>	<i>Zeitraum zur Erl. [h]</i>			
1	<i>Auf.Nr.</i>	<i>Wert</i>				<i>Auf.Nrn.</i>	<i>Ges.Wert</i>
2	1	73	12	30		1-3-2-6	22
3	2	61	10	40		2-3-1-6	22
4	3	55	8	50		3-1-2-6	22
5	4	12	11	30		4-3-2-6	17
6	5	48	14	35		5-3-2-6	19
7	6	33	10	45		6-3-1-2	22

Abb. 9-3 Auswertung der Beispieldaten