Binäre Suche – ein kurzer Algorithmus

Steffen Solyga*

21. Juni - 3. Juli 2010

Folgender Algorithmus liefert für jedes aufsteigend sortierte Array v beliebiger Anzahl n gültiger Komponenten den linken Einfügeindex, d.h. jenen Index i, für den $v[i-1] < x \leq v[i]$ gilt, sowie die Information, ob dort sogar Gleichheit vorliegt:

```
VAR
  v: ARRAY [ 0 .. max - 1 ] OF CARDINAL;      (* v[i] <= v[i+1] *)
  n: CARDINAL;                                (* n <= max *)

PROCEDURE Find( x: CARDINAL; VAR i: CARDINAL ): BOOLEAN;
VAR a, b, c: CARDINAL;
BEGIN
  a := 0; b := n;
  WHILE a < b DO
    c := ( a + b ) DIV 2;
    IF x <= v[c] THEN b := c ELSE a := c + 1 END;
  END;
  i := a;
  RETURN ( a < n ) & ( x = v[a] );
END Find;
```

Er ist [1] entnommen und in mehrerer Hinsicht interessant: Er ist kurz, erlaubt sowohl Indizierung ab 0 als auch leere Tabellen ($n = 0$), benötigt keinen `ELSIF`-Zweig, läßt sich ohne Probleme in eine „rechter Einfügeindex"-Version verwandeln und hat es inzwischen auf die Titelseite geschafft, [2]. Jedoch ist er asymmetrisch, und es hat mich einige Zeit gekostet, ihn zu verstehen, seine Allgemeingültigkeit einzusehen. Letztlich mußte ich ihn formal bezwingen.

1 Trivialfall $n = 0$

Nach Initialisierung von a, b gilt $a = b = n = 0$, die `WHILE`-Schleife wird nicht betreten, so daß $i = 0$ und `FALSE` zurückgeliefert werden.

*solyga@gmx.de

2 Halteproblem ($n > 0$)

Nach Initialisierung von a, b gilt $a < b$, die `WHILE`-Schleife wird wenigstens einmal durchlaufen. Ist ihr Verlassen gesichert?

Beim Eintritt in die Schleife gilt stets $a < b$; nach jedem Setzen von c ist folglich $a \leq c < b$. Sollte der `TRUE`-Zweig durchlaufen werden, wird b sicher verringert, anderenfalls wird a sicher vergrößert. Unabhängig von x und v wird der Zustand $a \geq b$ und damit der Austritt aus der Schleife also immer erreicht.

3 Bereichsproblem

Auf Komponenten von v wird nur an zwei Stellen zugegriffen: Bei Rückkehr aus der Prozedur und in der Schleife. Ersterer ist per $a < n$ geschützt, allein letzterer verdient eine nähere Betrachtung: Initial gilt $b \leq n$. Da b niemals vergrößert wird (und n eine Konstante ist), entpuppt sich $b \leq n$ als Schleifeninvariante. Vor Auswertung der `IF`-Bedingung ist aber stets $c < b$ (siehe oben) und folglich $c < n$. Auf ungültige Komponenten wird also niemals zugegriffen.

4 Invarianten

Satz 0 *An jeder Stelle der Prozedur gilt*

$$\forall i : \ 0 \leq i < n-1 \ \Rightarrow \ v[i] \leq v[i+1]. \tag{1}$$

Beweis: Hier handelt es sich lediglich um eine formale Darstellung der Ordnungsbedingung. Sie schließt auch die Fälle $n = 0$ und $n = 1$ ein, wenn man (wie üblich) eine Implikation mit falscher Voraussetzung als wahr ansieht. □

Satz 1 *Nach Initialisierung von a, b gilt an jeder Stelle der Prozedur*

$$0 \ \leq \ a \ \leq \ b \ \leq \ n. \tag{2}$$

Beweis: Aus den Initialwerten und der Monotonie von a und b ergeben sich die äußeren beiden Relationen sofort. Verbleibt die innere: Aus $n \in$ `CARDINAL` folgt $0 \leq n$, so daß $a \leq b$ jedenfalls initial gilt ($a = 0 \leq n = b$). Modifiziert werden a und b nur in der Schleife. Im Falle $n = 0$ ist $a = b$, die Schleife wird nicht betreten, so daß für alle Zeiten $a = b$ gilt. Im Falle $n > 0$ wird sie betreten; vor der `IF`-Anweisung gilt stets $a \leq c < b$, also $a \leq c$ und $c + 1 \leq b$. Bezeichnet man die Werte von a und b unmittelbar nach der `IF`-Anweisung mit a' bzw. b', so hat man im `TRUE`-Zweig $b' = c \geq a = a'$ und im `FALSE`-Zweig $a' = c + 1 \leq b = b'$, also nach der `IF`-Anweisung in jedem Falle $a' \leq b'$. Mithin gilt $a \leq b$ an jeder Stelle der Schleife (und damit überall in der Prozedur nach Initialisierung). □

Satz 2 *Hinter der Schleife gilt*

$$a \ = \ b. \tag{3}$$

Beweis: Abbruchbedingung $a \geq b$ und Invariante (2) ergeben dies sofort. □

Satz 3 *Nach Initialisierung von a, b gilt an jeder Stelle der Prozedur*

$$\forall i: \ 0 \le i < a \ \Rightarrow \ v[i] < x, \tag{4}$$

$$\forall i: \ b \le i < n \ \Rightarrow x \ \le v[i]. \tag{5}$$

Beweis: Wegen (2) ist zunächsteinmal sichergestellt, daß alle angesprochenen Komponenten von v gültig sind. Initial sind beide Aussagen trivialerweise wahr, weil die Indexmengen leer sind, siehe Satz 0. Wird die Schleife nicht betreten, sind wir fertig. Anderenfalls gelten sie initial vor der `IF`-Anweisung. Bei *jedem* Durchlauf gilt im `TRUE`-Zweig $x \le v[c] \le v[c+1] \le \ldots$, nach Setzen von b also (5); weil a nicht modifiziert wurde, bleibt der Wahrheitswert von (4) bestehen. Im `FALSE`-Zweig gilt bei jedem Durchlauf $\ldots \le v[c-1] \le v[c] < x$, mit $a := c+1$ also $\ldots \le v[a-2] \le v[a-1] < x$, d.h. (4), und der Wahrheitswert von (5) bleibt bestehen. Induktiv folgt daraus die Richtigkeit beider Aussagen für alle Zeiten. □

Satz 4 *Hinter der Schleife gilt*

$$v[a-1] \ < \ x \ \le \ v[a], \tag{6}$$

sofern die angesprochenen Komponenten gültig sind.

Beweis: Voraussetzung für die *linke Relation* ist $1 \le a < n+1$, woraus sofort $0 \le a-1 < n$ folgt. Wir haben also einen nichttrivialen Fall[1] für (4), und mit $i = a-1$ folgt $v[a-1] < x$. Die *rechte Relation* hat $0 \le a < n$ zur Voraussetzung. Mit $a = b$ gemäß Satz 2 haben wir also einen nichttrivialen Fall[2] für (5), und mit $i = a$ folgt $x \le v[a]$. □

Also ist a der linke Einfügeindex für x in v.

5 Kurzformen

5.1 Linksform

```
i := 0; j := n;
WHILE i < j DO
  (* v[0] <= .. <= v[i-1] < x <= v[j] <= .. <= v[n-1] *)
  m := ( i + j ) DIV 2;
  IF v[m] < x THEN i := m + 1 ELSE j := m END;
END;
(* i = j, v[i-1] < x <= v[i] *)
```

5.2 Rechtsform

```
i := 0; j := n;
WHILE i < j DO
  (* v[0] <= .. <= v[i-1] <= x < v[j] <= .. <= v[n-1] *)
  m := ( i + j ) DIV 2;
  IF v[m] <= x THEN i := m + 1 ELSE j := m END;
END;
(* i = j, v[i-1] <= x < v[i] *)
```

[1] Die Indexmenge enthält wenigstens ein Element, nämlich $a-1$.
[2] Die Indexmenge enthält wenigstens ein Element, nämlich a.

Letzterer ist WIRTHS Titelseitencode von [2] – bereinigt um einen „Fehler“ im Kommentar: Auf dem Deckel steht $\ldots x < v[j] < \ldots < v[n-1]$, was (nicht ganz offensichtlich) falsch ist, z.B. im Falle $x=0$, $n=2$, $v=(1,1)$, bei dem mit $j=0$ geendet wird.[3]

Literatur

[1] Niklaus Wirth: Algorithmen und Datenstrukturen mit Modula-2.
B.G. Teubner, Stuttgart, 4. Auflage, 1986. ISBN 3-519-02260-5

[2] Niklaus Wirth: Algorithmen und Datenstrukturen mit Modula-2.
B.G. Teubner, Stuttgart, 5. Auflage, 1996. ISBN 3-519-12260-X

[3] Man möge einwenden, WIRTHS Kommentar behaupte die Invariante ja nicht beim Verlassen sondern nur zum Beginn der Schleife. Aber auch diese Behauptung läßt sich Falsifizieren: Im Falle $x = 0$, $n = 3$, $v = (1,1,1)$ startet der zweite Durchlauf mit $j=1$, so daß $v[j] < v[n-1]$ sicherlich nicht gilt.