5. Übung Informatik I

Marcus Rickert

30. Dezember 1995

Aufgabe 1

Erläuterungen

Der Algorithmus beruht darauf, daß bei jedem Schritt ermittelt wird, wie viele Elemente sich *insgesamt* links vom aktuellen Element befinden. Es erfolgt keine Plausibilitätsüberprüfung, ob der gewünschte Index außerhalb der Menge S liegt, d.h. ob $k < |S|$ oder $k > |S|$.

- Ist am Anfang der Schleife der gewünschte Index gleich der Anzahl der linken Elemente plus eins, so wurde das Element gefunden und die Schleife wird abgebrochen.
- Ist der Index kleiner oder gleich als die Anzahl der linken Elemente, so wird zum linken Sohn verzweigt.
- Ist der Index mehr als eins grösser als die Anzahl der linken Elemente, so wird zum rechten Sohn verzweigt.

Abkürzungen

root	Zeiger auf die Wurzel des Suchbaumes
such	Zeiger auf aktuellen Knoten
links	Anzahl der insgesamt linken Elemente
memo	Zwischenspeicher für Anzahl der insgesamt linken Elemente bei der letzten Verzweigung nach rechts

Algorithmus

```
PROCEDURE Ord (k,root)
    BEGIN
    such:=root                Suchzeiger initialisieren
    links:=0                  Zwischenspeicher initialisieren
    WHILE links+1<>k          so lange nicht gefunden
        BEGIN
        IF k>links
            BEGIN             gehe in rechten Teilbaum
            memo:=links       gesamten linken Teilbaum merken
            such:=such→rechter_sohn
            links:=links+1+such→linke_anzahl    hänge linken Teilbaum des neuen Knoten an
            END
        ELSE
            BEGIN             gehe in linken Teilbaum
            such:=such→linker_sohn
            links:=memo+such→linke_anzahl       alten linken Teilbaum zuaddieren
            END
        END
    return such→daten
    END
```

Laufzeit

Da im ungünstigsten Fall der Baum von der Wurzel aus nur einmal bis zum untersten Blatt durchsucht wird ist die Laufzeit trivialerweise $O(Höhe(T))$.

Aufgabe 2, 3 u. 4

Erläuterungen

- Der Wert $faktor$ bestimmt die Größe des Intervalls, aus dem die Zufallszahlen gezogen werden, relativ zur Feldgröße $anzahl$.
- Es werden für jede Kombination von $faktor$ und $anzahl$ mindestens *MIN_FALSCH* erfolglose und *MIN_RICHTIG* erfolgreiche Versuche durchgeführt.
- Die Prozedur *interpolations_suche* wird abgebrochen, falls der gewünschte Wert nicht mehr innerhalb des gebildeten Teilintervalls liegt.

Siehe Listing **info5.pas**

Aufgabe 5

Erläuterungen

Der Algorithmus der Vorlesung muß bei der Implementations um folgende Elemente erweitert werden:

- Die Abbruchbedingung der Rekursion ist dann erreicht, wenn *quadratische_suche* mit einem Intervall aufgerufen wird, in dem der kleinste Wert größer als der Suchwert oder der größte Wert kleiner als der Suchwert ist.
- Das erste bzw. letzte Teilintervall hat unter Umständen nicht die maximale Länge w. Daher ist eine gesonderte Behandlung dieses Falls nötig.
- Vor dem Aufruf von *quadratische_suche* müssen $s(0)$ und $s(n+1)$ auf das jeweilige Minimum bzw. Maximum der Menge $S = \{s(1), \cdots, s(n)\}$ gesetzt werden (oder zumindest auf eine untere bzw. obere Schranke).

Abkürzungen

anfang	Anfangsindex des aktuellen Intervalls
ende	Endindex des aktuellen Intervalls
x	Suchwert
n	Länge des Intervalls
w	Schrittweite beim Suchen
a	ausgezeichneter Index im aktuellen Intervall
i	relative Nummer des Teilintervalls von a aus
gefunden	Boolsche Variable falls Suchwert im Teilintervall

Algorithmus

PROCEDURE quadratische_suche (anfang,ende,x)

```
BEGIN
  n := ende − anfang + 1;                Intervallänge berechnen
  IF (x < s(anfang + 1)) or (s(ende − 1) > x)        Suchwert liegt außerhalb
      return −1              nicht gefunden
  w := ⌈√n⌉              Schrittweite berechnen
  a := ⌈ (x − s(anfang)) / (s(ende) − s(anfang)) · n ⌉          augezeichneten Index berechnen
  IF x = s(a)
      return a
  gefunden:=FALSE
  i := 0
  IF x > s(a)              rechten Teil durchsuchen
      BEGIN
      WHILE (aktuell + (i + 1)w ≤ ende − 1) and not gefunden          solange nicht Ende des Intervalls
          BEGIN
          IF (s(aktuell + iw) < x) and (s(aktuell + (i + 1)w ≥ x)
              gefunden:=TRUE
          ELSE
              i := i + 1              nächstes rechtes Teilintervall
          END
      IF gefunden
          return quadratische_suche(a + iw − 1, a + (i + 1)w + 1, x)          ganzes Teilintervall
      ELSE
          return quadratische_suche(a + iw − 1, ende, x)          rechtes Ende des Intervalls
      END
  ELSE              linken Teil durchsuchen
      BEGIN
      WHILE (a − (i + 1)w ≥ anfang + 1) and not gefunden          solange nicht Anfang des Intervalls
          BEGIN
          IF (s(aktuell − (i + 1)w) ≤ x) and (s(aktuell − iw) > x)
              gefunden:=TRUE
          ELSE
              i := i − 1              nächstes linkes Teilintervall
          END
      IF gefunden
          return quadratische_suche(a − (i + 1)w − 1, a − iw + 1, x)          ganzes Teilintervall
      ELSE
          return quadratische_suche(anfang, a − iw + 1, x)          linkes Ende des Intervalls
      END
  END
```