Übungen zu Algorithmen
WSI für Informatik
Kaufmann/Kuckuk/Förster

Wintersemester 2021/22
Universität Tübingen
17. Dezember 2021

Aufgabensammlung Teil 9

Thema: Suchen

Aufgabe 1: Suchen in unendlich großen Arrays

Dies ist eine alte Klausuraufgabe!

Sei ein unendliches Array A gegeben, das bis auf die ersten n Einträge mit dem Wert '∞' gefüllt ist. Die ersten n Einträge sind aufsteigend sortierte natürliche Zahlen. Sie kennen den Wert von n **nicht**. Außerdem ist eine natürliche Zahl x gegeben.
Geben Sie einen $\mathcal{O}(\log(n))$-Algorithmus in Pseudocode an, der die Position von x im Array A zurückgibt. Falls x nicht in A vorkommt, soll -1 zurückgegeben werden. Beschreiben Sie die Idee und begründen Sie die Korrektheit sowie die Laufzeit Ihres Algorithmus.

Aufgabe 2: Ternäre Suche

Bei der binären Suche ist der Input ein sortiertes Array A und eine Zahl x (die nicht notwendig im Array A vorkommen muss). Dabei wird A in zwei gleich große Teile A_1 und A_2 geteilt und ermittelt, in welchem der beiden Teile sich x befinden müsste. Dieses Verfahren wird *rekursiv* fortgesetzt. Betrachten Sie nun die *ternäre Suche*, bei der A statt in zwei Teile, in *drei* etwa gleich große Teile A_1, A_2 und A_3 geteilt wird.

a) Geben Sie ein Array A und ein zu suchendes Element x an, so dass die binäre Suche mit weniger Vergleichen auskommt als die ternäre Suche.

b) Geben Sie ein Array A und ein zu suchendes Element x an, so dass die ternäre Suche mit weniger Vergleichen auskommt als die binäre Suche.

c) Analysieren Sie die Zeitkomplexität der ternären Suche. Was können Sie über die asymptotische Laufzeit der ternären Suche im Vergleich zur binären Suche sagen?

d) Bei jedem Rekursionsschritt werden ein oder zwei Vergleiche benötigt, um zu entscheiden, in welchem Teil des Arrays A das Element x liegt. Was ist die minimale, die durchschnittliche und die maximale Anzahl an Vergleichen die benötigt wird, wenn x nicht in A liegt.

Aufgabe 3: Ausführen von Suchalgorithmen

Gegeben sind die folgenden zwei Arrays mit je 16 Elementen.

$$S_1 = [2, 3, 6, 9, 10, 11, 14, 17, 18, 25, 26, 28, 29, 33, 36, 39],$$
$$S_2 = [1, 2, 5, 10, 17, 34, 63, 127, 266, 519, 1025, 2043, 4099, 8183, 16379, 32773, 65555].$$

Führen Sie auf diesen Arrays binäre Suche und Interpolationssuche nach $a = 63$ aus. Runden sie 'next' dabei auf, falls nötig. Nutzen Sie zur Bestimmung von *next* folgende Formeln:

Binäre Suche: $next \leftarrow \lceil (oben - unten)/2 \rceil + unten$
Interpolationssuche: $next \leftarrow \lceil \frac{a - S[unten]}{S[oben] - S[unten]} \cdot (oben - unten) \rceil + unten$

Nutzen Sie auch für die Interpolationssuche das folgende im Vergleich zur Vorlesung vereinfachte[1] Vorgehen:

```
1  while oben − unten > 1 do
2  |  Berechne next;
3  |  if S[next] < a then
4  |  |  unten ← next;
5  |  end
6  |  if S[next] > a then
7  |  |  oben ← next;
8  |  end
9  |  if S[next] = a then
10 |  |  return next;
11 |  end
12 end
13 return nil;
```

[1] Das hier vorgestellte Verfahren entspricht z.B. der Beschreibung der Interpolationssuche auf Wikipedia. Das in der Vorlesung vorgestellte verbesserte Verfahren ist auch also *Quadratische Binärsuche* bekannt.

Lösungen:

1. Prüfe iterativ die Einträge von A an Stelle 2^i für i aufsteigend. Nach $i^* \leq \min\{\log x + 1, \log n + 1\}$ Schritten wird eine Zahl, die größer ist als x, oder ∞ gefunden. Finde dann x mit binärer Suche auf den Array Elementen zwischen Stellen 2^{i^*-1} und 2^{i^*} (dies sind $\mathcal{O}(\min\{x, n\})$ Elemente).

2. a) $A = [1, 2, 3, 4, 5, 6, 7, 8]$ und $x = 5$
   b) $A = [1, 2, 3, 4, 5, 6, 7, 8]$ und $x = 3$
   c) Laufzeit ist $\Theta(\log_3 n)$. Dies ist aber asymptotisch nicht besser als binäre Suche.
   d) Minimum: etwa $\log_3(n)$
   Maximum: etwa $2\log_3(n)$
   Durchschnitt: etwa $\frac{5}{3}\log_3(n)$

3. Binäre Suche auf S_1:

   Iteration 1: $unten = 0, oben = 15, next = 8$

   Iteration 2: $unten = 8, oben = 15, next = 12$

   Iteration 3: $unten = 12, oben = 15, next = 14$

   Iteration 4: $unten = 14, oben = 15, next = 15$

   Interpolationssuche auf S_1:

   Iteration 1: $unten = 0, oben = 15, next = 25$

   Binäre Suche auf S_2:

   Iteration 1: $unten = 0, oben = 15, next = 8$

   Iteration 2: $unten = 0, oben = 8, next = 4$

   Iteration 3: $unten = 4, oben = 8, next = 6$

   Interpolationssuche auf S_2:

   Iteration 1: $unten = 0, oben = 15, next = 1$

   Iteration 2: $unten = 1, oben = 15, next = 2$

   Iteration 3: $unten = 2, oben = 15, next = 3$

   Iteration 4: $unten = 3, oben = 15, next = 4$

   Iteration 5: $unten = 4, oben = 15, next = 5$

   Iteration 6: $unten = 5, oben = 15, next = 6$