Data Structures and Algorithms
The Basic Toolbox
Corrections and Remarks

Kurt Mehlhorn and Peter Sanders

September 2, 2013

In this document, we collect corrections and remarks. Negative line numbers count from the bottom of a page.

Corrections

Page 2, line 4 of second paragraph: section $\rightarrow$ chapter

Page 7, line 1: $s_c \rightarrow c$

Page 9, line 16: n-bit $\rightarrow$ n-digit

Page 9, line -5: $a_1 \cdot b_0 \rightarrow a_0 \cdot b_0$

Page 11, line 17: n-bit $\rightarrow$ n-digit

Page 11, statement of Theorem 1.7: Then $T_K(n) \leq 99n^{\log 3} + 48 \cdot n + 48 \cdot logn$ $\rightarrow$ Then $T_K(n) \leq 207 \cdot n^{\log 3}$.

Page 16, line -3 to Page 17, line 17: We shall show that

$$T_K(2^k + 2) \leq 69 \cdot 3^k - 24 \cdot 2^k - 12$$

for $k \geq 0$. For $k = 0$, we have

$$T_K(2^0 + 2) = T_K(3) \leq 3 \cdot 3^2 + 2 \cdot 3 = 33 = 69 \cdot 3^0 - 24 \cdot 2^0 - 12 \ .$$

For $k \geq 1$, we have

$$\begin{aligned}
T_K(2^k+2) &\leq 3T_K(2^{k-1}+2)+12\cdot(2^k+2) \\
&\leq 3\cdot\left(69\cdot 3^{k-1}-24\cdot 2^k-12\right)+12\cdot(2^k+2) \\
&= 69\cdot 3^k-24\cdot 2^k-12\ .
\end{aligned}$$

Again, there is no magic in coming up with the right induction hypothesis. It is obtained by repeated substitution. Namely,

$$\begin{aligned}
T_K(2^k+2) &\leq 3T_K(2^{k-1}+2)+12\cdot(2^k+2) \\
&\leq 3^kT_K(2^0+2)+12\cdot\left(3^0(2^k+2)+3^1(2^{k-1}+2)+\ldots+3^{k-1}(2^1+2)\right) \\
&\leq 33\cdot 3^k+12\cdot\left(2^k\frac{(3/2)^k-1}{3/2-1}+2\frac{3^k-1}{3-1}\right) \\
&\leq 69\cdot 3^k-24\cdot 2^k-12\ .
\end{aligned}$$

It remains to extend the bound to all n. Let k be the minimal integer such that $n \leq 2^k+2$. Then $k \leq 1+\log n$. Also, multiplying n-digit numbers is no more costly than multiplying (2^k+2)-digit numbers, and hence

$$\begin{aligned}
T_K(n) &\leq 69\cdot 3^k-24\cdot 2^k-12 \\
&\leq 207\cdot 3^{\log n} \\
&\leq 207\cdot n^{\log 3}\ ,
\end{aligned}$$

where the equality $3^{\log n}=2^{(\log 3)\cdot(\log n)}=n^{\log 3}$ has been used.

Page 17, Exercise 1.9 Replace the exercise by

Solve the recurrence

$$T_R(n) \leq \begin{cases} 3n^2+2n & \text{if } n < n_0, \\ 3\cdot T_R(\lceil n/2\rceil+1)+12n & \text{if } n \geq n_0, \end{cases}$$

where n_0 is a positive integer. Optimize n_0.

Page 22, line -4: constant $\rightarrow$ constant c.

Page 38, line -9: $d=b=4 \rightarrow d=b=2$.

Page 39, line 1: W $\rightarrow$ We

Page 40, third paragraph: replace the third paragraph by the following:

We make the following additional assumptions: $b \geq 2$ is integral and $a \leq c(n_0+1)+da$.

We first show $R(n-1) \leq R(n)$ for all n by induction on n. If $n \leq n_0$, the claim is obvious. We have $R(n_0) = a \leq c(n_0+1)+da = R(n_0+1)$ by the additional assumption. For $n > n_0+1$, we have $R(n) = cn + R(\lceil n/b \rceil + e) \geq c(n-1) + R(\lceil (n-1)/b \rceil + e) = R(n-1)$, where the inequality follows from the Induction Hypthesis. Observe that $\lceil n/b \rceil + e < n$ since $n > n_0$. We now have monotonicity of R and hence $R(n) \leq R(s(n))$.

We next turn to the solution of the recurrence. We need the additional assumption[1] $b^0 + z \leq n_0$.

Let k_0 be maximal such that $b^{k_0} + z \leq n_0$. Then $k_0 \geq 0$, by the assumption. The recurrence for numbers of the form $b^k + z$, $k \geq 0$ is

$$R(b^k+z) = \begin{cases} a & \text{if } k \leq k_0 \\ c(b^k+z) + dR(b^{k-1}+z) & \text{if } k > k_0. \end{cases}$$

For $k \geq k_0$, the solution to this recurrence is

$$R(b^k+z) = ad^{k-k_0} + c \sum_{0 \leq i \leq k-k_0-1} d^i (b^{k-i}+z)$$

as an induction of k shows.

We now distinguish cases as in the proof of the master theorem: $d < b$, $d = b$, and $d > b$.

Page 44, line 7: $(p_i - p_j) \to (p_j - p_i)$.

Page 47, line 2 of second paragraph: smallest L primes with $\to$ smallest L primes $p_1, \ldots, p_L$ with

Page 47, line -12: $L = 10^{12}n \to L = 10^{12}(n/k)$

Page 55, line 6: $K_{33} \to K_{3,3}$.

Page 55: Exercise 2.9 $\to$ Exercise 2.20

Page 56: Exercise 2.20 $\to$ Exercise 2.21

[1] As it stands, we only know $z \leq n_0$ since by defintion of z, $\lceil z/b \rceil + e = z$

Page 56: Exercise 2.21 $\to$ Exercise 2.22

Page 61, line -13: elements $\to$ items

Page 61, line -2: positions $\to$ position

Page 63, lines -15, -9, and -7: element $\to$ item

Page 71, line 7: replace "followed by $k-1$ zeros" by "followed by k zeros".

Page 74/75: *stacks*, *queue*s, and *deque*s also support the operation *isEmpty*.

Page 83, line 3: m's $\to$ m_ℓ's

Page 83, lines -6 and -4: $h(e) = k \to key(e) = k$

Page 85, line 5: We have to define X to exclude a possible list element with key k which will certainly be in the list regardless how the hash function is choosen.

Page 85, line 10: $X_i \to X_e$.

Page 86, line 20: $X_i \to X_e$.

Page 87, line -14: "$g : 0..\alpha n \to \{0,1,2\}$" $\longrightarrow$ "$g : 0..m-1 \to \{0,1,2\}$".

Page 87, lines 4, 15, -4: $h_a(x) \to h_a(\mathbf{x})$.

Page 97, line -5 k-wise $\to$ k-way

Page 110, Theorem 5.6: $1.45n\log n \to 1.39n\log n$.

Page 110, line -2: Sect. 2.8 $\to$ Sect. 2.7

Page 114, line 9: $e_{\lfloor n/2 \rfloor} \to e'_{\lfloor n/2 \rfloor}$.

Page 123, line 4: $C := 0 \to C := 1$.

Function *ABItem::locateRec*(k : *Key*, h : $\mathbb{N}$) : *Handle*
  i:=*locateLocally(k)*
  if $h = 1$ **then**
    $\boxed{\textbf{if } c[i] \rightarrow e \geq k \textbf{ then}}$ **return** $c[i]$
    $\boxed{\textbf{else return } c[i] \rightarrow next}$
  else return $c[i] \rightarrow$*locateRec*(k, $h-1$)

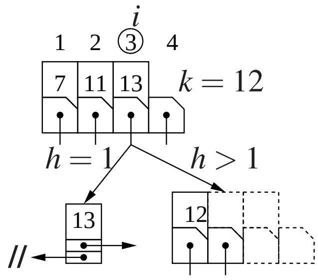

Figure 1: Corrected version of function *locateRec* for (a,b)-trees. The framed pieces are new.

Page 131, line 17: no larger $\rightarrow$ no smaller.

Page 140, line 19: Change "If the minimum is in Q' and comes from sequence S_i, the next largest element of S_i is inserted into Q'" into "If the minimum is in Q' and comes from sequence S_i, the first element of S_i is moved to Q'" for increased clarity.

Pages 147 and 150: Implementation of *locate*. Once again, we see that the phrase "It is clear" is dangerous. Our invariants for both binary search trees and (a,b)-trees do not guarantee that the leaf x we reach is actually the list element specified as the result of *locate*(k). When we remove an element somewhere used as a splitter, we may end up at an element $x < k$. This is easy to fix however: If $x < k$ return the successor of x in the linked list. Figure 1figure.1 shows that we need to change only two lines of pseudocode. We also need to change the insertion method. If we end up at an element $x < k$ we simply swap it with the element to be inserted before proceeding with the insertion.

Page 158, line 1: $h_h \rightarrow h_k$.

Page 162, line -6: two kind of operation $\rightarrow$ two kind of operations.

Page 169, line 5: out of $V \rightarrow$ out of v.

Page 170, line -15: discussed in Chap. 2.9 $\rightarrow$ discussed in Section 2.9.

Page 172, line 3: Change "disconnected" into "not connected" for increased linguistic quality.

Page 173, line -17: Change

> The algorithms should access the graph data structure only through a small set of operations, such as .... The interface can be captured in an interface description, and a graph algorithm can be run on any representation that realizes the interface.

into

> The algorithms should access the graph data structure only through a small interface – a set of operations, such as .... ldots. An algorithm that only accesses the graph only through this interface can be run on any representation realizing the interface.

for increased clarity.

Page 187, line -8: than $\to$ that

Page 211, Line -9– -7: $d(\cdot) \to d[\cdot]$.

Page 222, Line 4: $O(V) \to O(n)$.

Page 227: The original edge (u_0, v_0) stored at the end of the priority queue tuples should consistently be put into parentheses.

Page 247: The instance used here violates the assertion that the profit densities should be sorted. Better use the profit vector $\mathbf{p} = (10, 24, 14, 20)$. Furthermore, $\sum_{k<i} x_i w_i \to \sum_{k<i} x_k w_k$, $\sum_{k<i} x_i p_i \to \sum_{k<i} x_k p_k$ and $\sum_{j<i} x_i w_i \to \sum_{j<i} x_j w_j$.

Remarks

Page 40, Section 2.6.2, Theorem 2.6, master theorem for the analysis of recurrences: Recent papers with many interesting generalizations of the master theorem are [1, 3, 2]. .

References

[1] M. Akra and L. Bazzi. On the solution of linear recurrence equations. *Computational Optimization and Applications*, 10(2):195–210, 1998.

[2] T. Leighton. Notes on better master theorems for divide and conquer recurrences. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.1636`.

[3] C. Yap. A real elementary approach to the master recurrence and generalizations. `http://www.cs.nyu.edu/cs/faculty/yap/papers/SYNOP.htm`.