Real World Algorithms: A Beginners Guide
Errata to the Second Printing

Last updated 12 April 2019

This document lists the changes that should be made to *Real World Algorithms* to correct mistakes that made their way to printing, to improve infelicities that the author spotted too late, or update the material with something that the author did not know at the time of writing the book.

There are three different kinds of changes noted here. In all of them the date that they became known to the author is given at the first line of each item. The name of the person who suggested the change is also given at the end of each change.

► **Page 1**, line 1 ———— 1 Jan 1

These are technical or typographical errors.

Page 1, line 1 ———— 1 Jan 1

These as changes that improve the book, even if they do not correct an error. They include small rewordings, or material that became known to the author after the book was published.

Page 1, line 1 ———— 1 Jan 1

These are minor fixes that although they do not make a big difference they do hurt the author. Some of them might strain the reader's eye to see where the improvement is exactly.

Page 15, line 15 — 10 Feb 2019

greater than $\rightsquigarrow$ not less than

Page 17, line −16 — 10 Feb 2019

in the stack $\rightsquigarrow$ on the stack

Page 17, line −6 — 10 Feb 2019

at the top $\rightsquigarrow$ on the top

► **Page 20**, line −1 — 14 Feb 2018

we cannot execute line 7 more than n times. $\rightsquigarrow$ we cannot execute line 7 more than $n-1$ times; note that the last day is pushed, but not popped. (K. Marinakos)

► **Page 32**, line −2 — 16 Feb 2018

2.5×10^{25}, or 7 septillion $\rightsquigarrow$ 2.5×10^{19}, or 25 quintillion (K. Marinakos)

► **Page 32**, line 8 — 16 Feb 2018

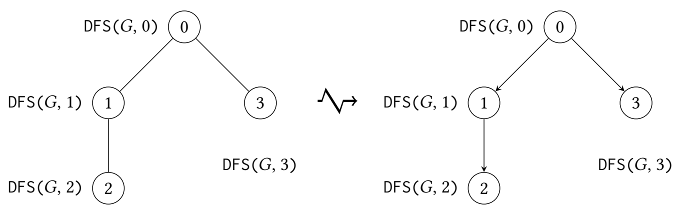
In an adjacency matrix, vertices are represented by row and column indices, and vertices are represented by the contents of the matrix. $\rightsquigarrow$ In an adjacency matrix, the vertices are represented by row and column indices, and the edges are represented by the contents of the matrix. (K. Marinakos)

► **Page 39**, line −5 — 16 Feb 2018

Similarly, if $|E|$ is the number of edges in the graph, $\rightsquigarrow$ Similarly, if $|E|$ is the number of edges in the graph, counting undirected edges twice, (K. Marinakos)

► **Page 44**, figure 2.21 — 17 Feb 2018

Add arrows so that the graph is directed:



(K. Marinakos)

► **Page 49**, algorithm 2.3, line 7 — 16 Feb 2018

$c \leftarrow \text{Pop}(s)$ $\rightsquigarrow$ $c \leftarrow \text{Pop}(S)$ (K. Marinakos)

► **Page 50**, lines 2–4 — 16 Feb 2018

Line 2 is executed $|V|$ times, once per each vertex. Then DFS(G, *node*) is called exactly once per edge, in line 4, that is, $|E|$ times. ⇝ Line 4 is executed $|V|$ times, once per each vertex. The condition in line 3 is called exactly once for each edge of every adjacency list, that is, $|E|$ times.

▶ **Page 54**, line 5 — 16 Feb 2018

we only node ⇝ we only note (K. Marinakos)

▶ **Page 49**, line 4 — 17 Feb 2018

the same as algorithm 2.4 ⇝ the same as algorithm 2.3 (K. Marinakos)

▶ **Page 55**, figure 2.28a — 17 Feb 2018

rename nodes 7 and 8 to 6 and 7 respectively (K. Marinakos)

▶ **Page 61**, line 7 — 26 Feb 2018

with number in different number systems ⇝ with numbers in different number systems (K. Marinakos)

▶ **Page 61**, lines 17–18 — 26 Feb 2018

The binary number 1010 has value 14 ⇝ The binary number 1110 has value 14 (K. Marinakos)

▶ **Page 65**, line 10 — 26 May 2018

in 32 bits ⇝ in 33 bits (M. Chatzidavid)

▶ **Page 69**, line −13 — 26 Feb 2018

Each element of the priority tree ⇝ Each element of the priority queue (K. Marinakos)

▶ **Page 72**, line −2 — 26 Feb 2018

larger than its parent ⇝ lighter than its parent (K. Marinakos)

▶ **Page 73**, lines 3–6 — 11 Mar 2019

In this algorithm we assume we have function `AddLast(`pq, c`)` that adds a node c as last node to the priority queue pq, function `Root(`pq`)` that returns the root of a priority queue

⇝

In this algorithm we use function `AddLast(`pq, c`)` that adds a node c as last node to the priority queue pq, functions `GetRoot(`pq`)` and `SetRoot(`pq, c`)` to handle the root of a priority queue (A. Sotiropoulos)

▶ **Page 73**, line −1 — 26 Feb 2018

larger than its children ⇝ smaller than its children (K. Marinakos)

▶ **Page 74**, algorithm 3.2, line 2 ________ 11 Mar 2019

`Root(`pq`)` ⇝ `GetRoot(`pq`)` (A. Sotiropoulos)

▶ **Page 75**, algorithm 3.3, lines 1–2 ________ 11 Mar 2019

$c \leftarrow$ `Root(`pq`)`
`Root(`pq`)` $\leftarrow$ `ExtractLastFromPQ(`pq`)`
⇝
$c \leftarrow$ `GetRoot(`pq`)`
`SetRoot(`pq`, ExtractLastFromPQ(`pq`))` (A. Sotiropoulos)

Page 91, line −6 to −5 ________ 31 May 2018

verify that with such an input, the Huffman encoder will not perform better than a fixed-length encoding
⇝
investigate how the Huffman encoder will perform with such an input in comparison to a fixed-length encoding (I. Lazaridou)

▶ **Page 94**, line 15 ________ 24 Jun 2018

is encoded as the plaintext ⇝ is encoded as the ciphertext (K. Marinakos)

▶ **Page 122**, table 4.10, table row 6 ________ 05 Apr 2019

229 ⇝ 292

▶ **Page 126**, line 2 ________ 24 Jun 2018

to keep a message ⇝ to keep a message secret (K. Marinakos)

▶ **Page 161**, line 14 ________ 26 Mar 2018

Beceause ⇝ Because (K. Marinakos)

▶ **Page 161**, line 15 ________ 24 Jun 2018

each time we find a longest path ⇝ each time we find a longer path (K. Marinakos)

▶ **Page 173**, figure 7.4 ________ 19 Mar 2018

was astonished whenever it shone in her face. Close by
was astonished whenever it shone in her face. Close by the
⇝
was astonished whenever it shone in her face. Close by
was astonished whenever it shone in her face. Close by the

▶ **Page 180**, line −17 to −16 ________ 19 Mar 2018

then the number of nodes cannot be more than the number of edges ⇝ then the number of nodes minus the source cannot be more than the number of edges

▶ **Page 192**, figure 8.3 (c)–(h) ________ 26 May 2018

$1/R_1$ ⇝ 1/D (I. Lazaridou)

▶ **Page 192**, figure 8.3 (h), line R_5 ________ 21 Mar 2018

$\underline{5/R_2}$ ⇝ $\underline{5/R_3}$ (M. E. Kostopoulou)

▶ **Page 194** line −4 ________ 26 Mar 2018

exactly one ⇝ exactly once (K. Marinakos)

▶ **Page 196** line −7 ________ 26 Mar 2018

$(2, 1)$ ⇝ $(2, 2)$ (K. Marinakos)

▶ **Page 196** line −1 ________ 26 Mar 2018

eighth ⇝ seventh (K. Marinakos)

▶ **Page 198** line 12 ________ 26 Mar 2018

they story short ⇝ the story short (K. Marinakos)

▶ **Page 212**, line −14 to −13 ________ 24 Jun 2018

the importance of the page $|P_j|$ ⇝ the importance of the page P_j (K. Marinakos)

▶ **Page 216**, line −14 to −12 ________ 09 Apr 2019

The successive matrix multiplications form the *power method* because they involve raising a vector, in our instance the vector of PageRank values, to successive powers.

⇝

The successive multiplications form the *power method* because they are equivalent to multiplying the initial vector of PageRank values with successive powers of the hyperlink matrix. (K. Marinakos)

▶ **Page 231** lines 8–9 ________ 18 Apr 2018

A beats B by 60 to 40, B beats C by 60 to 40, and C beats A by 60 to 40 ⇝ A beats B by 60 to 30, B beats C by 60 to 30, and C beats A by 60 to 30 (K. Marinakos)

▶ **Page 232** line 1 ________ 18 Apr 2018

$i = 1, 2, \ldots n$ ⇝ $i = 1, 2, \ldots, n$

▶ **Page 232** line −11 ________________ 18 Apr 2018

This requires $\Theta(|B|^2)$ time. ⇝ This requires $\Theta(|C|^2)$ time. (K. Marinakos)

▶ **Page 233** line 2 ________________ 18 Apr 2018

runs in $O(|C|^2 + |B|^2)$ time. ⇝ runs in $O(|C|^2 + |B||C|^2)$ time (K. Marinakos)

▶ **Page 241**, algorithm 10.3, Input ________________ 18 Apr 2018

S, an array of size $n \times n$ with the strongest paths between nodes; $s[i, j]$ is the strongest path between nodes i and j

⇝

S, an array of size $n \times n$ with the strengths of the strongest paths between nodes; $s[i, j]$ is the strength of the strongest path between nodes i and j

▶ **Page 241**, algorithm 10.3, Output ________________ 18 Apr 2018

wins, a list of size n; item i of *wins* is a list containing m integer items $j_1, j_2, \ldots, j_m$ for which $S[i, j_k] > S[j_k, i]$

⇝

wins, an array of size n; item i of *wins* is a list containing m integer items $j_1, j_2, \ldots, j_m$ for which $S[i, j_k] > S[j_k, i]$

▶ **Page 241**, algorithm 10.3, line 1 ________________ 18 Apr 2018

wins ← `CreateList()`

⇝

wins ← `CreateArray(`*n*`)`

▶ **Page 241**, algorithm 10.3, line 4 ________________ 18 Apr 2018

`InsertInList(`*wins*, NULL, *list*`)`

⇝

wins[*i*] ← *list*

▶ **Page 241**, lines 3–4 ________________ 18 Apr 2018

a list *wins* such that item *i* of the list *wins* ⇝ an array *wins* such that item *i* of the array *wins* (K. Marinakos)

▶ **Page 241** line −7 ________________ 18 Apr 2018

$O(|C|^2 + |B|^2)$ time ⇝ $O(|C|^2 + |B||C|^2)$ time (K. Marinakos)

▶ **Page 248**, line 2 ________________ 24 Jun 2018

An fundamental distinction ⇝ A fundamental distinction (K. Marinakos)

► **Page 260**, line 2 ________ 24 Jun 2018

take it from its place it and move it

⇝

take it from its place and move it (K. Marinakos)

► **Page 263**, line −3 to −2 ________ 24 Jun 2018

pick up the last one in the pile

⇝

then indicate failure somehow (K. Marinakos)

► **Page 265**, lines 19–20 ________ 24 Jun 2018

$O(m/2+(n+1)/2 = O(n/2e+(n-1)/2) = O(n)$

⇝

$O(m/2+(n+1)/2) = O(n/2e+(n+1)/2) = O(n)$ (K. Marinakos)

► **Page 284**, line 6 ________ 03 Sep 2018

an item is counted ⇝ an item is found

► **Page 284**, line 14 ________ 03 Sep 2018

most-to-front ⇝ move-to-front

► **Page 291**, line −10 ________ 24 Jun 2018

as long as $A[j]$ is higher than the $A[j-1]$

⇝

as long as $A[j-1]$ is higher than $A[j]$

(K. Marinakos)

Page 305, lines 2–5 ________ 06 Aug 2018

If one pile runs out before the other, it means that all the remaining cards in that pile have larger face values than the cards in the third pile

⇝

When one pile runs out of cards, it means that all the remaining cards in the other pile have larger face values than the cards in the third pile

Page 306, line 1 ________ 24 Jun 2018

If one of the sorted arrays runs out of elements ⇝ When one of the sorted arrays runs out of elements (K. Marinakos)

► **Page 308**, line −3 to −2 ________ 12 Apr 2019

to copy the items we need to loop $h-l$ times: $|C| = h-l$

⇝

to copy the items we need to loop $h-l+1$ times: $|C| = h-l+1$

Page 311, line −2 ________ 06 Aug 2018

a midpoint ⇝ the midpoint

▶ **Page 311**, line −1 ________ 24 Jun 2018

`MergeSort(`A, m, h`)` ⇝ `MergeSort(`A, l, m`)` (K. Marinakos)

▶ **Page 319**, algorithm 12.10, Result ________ 24 Jun 2018

A is partitioned so that $A[0], \ldots, A[p-1] < A[p]$ and $A[p+1], \ldots, A[n-1] \geq A[p]$, for $n = |A|$

⇝

A is partitioned so that $A[0], \ldots, A[b-1] < A[b]$ and $A[b+1], \ldots, A[n-1] \geq A[b]$, for $n = |A|$ (K. Marinakos)

▶ **Page 323**, line 18 ________ 06 Aug 2018

the smallest element the first time is $1/n$, if we suppose that all

⇝

the smallest or the (equally bad) biggest element the first time is $2/n$, if all (K. Marinakos)

▶ **Page 323**, line −18 to −17 ________ 06 Aug 2018

the smallest element the second time is $1/(n-1)$ ⇝ the smallest or biggest element the second time is $2/(n-1)$ (K. Marinakos)

▶ **Page 323**, line −16 ________ 06 Aug 2018

an array with two elements, when the probability is $1/2$ ⇝ an array with three elements, when the probability is $2/3$ (K. Marinakos)

▶ **Page 323**, line −14 ________ 06 Aug 2018

$$\frac{1}{n} \times \frac{1}{n-1} \times \cdots \times \frac{1}{2} = \frac{1}{1 \times 2 \times \cdots \times n} = \frac{1}{n!}$$

⇝

$$\frac{2}{n} \times \frac{2}{n-1} \times \cdots \times \frac{2}{3} = \frac{2^{n-2}}{3 \times \cdots \times n} = \frac{2^{n-1}}{1 \times 2 \times 3 \times \cdots \times n} = \frac{2^{n-1}}{n!}$$

(K. Marinakos)

▶ **Page 323**, line −13 to −12 ________ 06 Aug 2018

The value $1/n!$ is small indeed; for just ten elements we get $1/10! = 1/3628800$, less than one chance in 3.5 million.

⇝ The value $2^{n-1}/n!$ is small indeed; for just fifteen elements we get $2^{14}/15! \approx 1/79{,}814{,}109$. (K. Marinakos)

Page 341, line 2 ———— 20 May 2018

$$v_4 = 3,276,858 + \texttt{Ordinal}(\text{“O”}) = +3,276,858 + 14 = 3,276,872$$

⇝

$$v_4 = 3{,}276{,}858 + \texttt{Ordinal}(\text{“O”}) = 3{,}276{,}858 + 14 = 3{,}276{,}872$$

► **Page 354**, line 1 ———— 24 Jun 2018

size $2n$ ⇝ size $\lfloor n/2 \rfloor + 1$ (K. Marinakos)

► **Page 366**, line −7 to −6 ———— 24 Jun 2018

The words in our example take up 41 bytes, equal to 328 bits ⇝ The words in our example take up 33 bytes, equal to 264 bits (K. Marinakos)

► **Page 366**, line −5 ———— 24 Jun 2018

$328/16 \approx 20$ ⇝ $264/16 = 16.5$ (K. Marinakos)

► **Page 367**, figure 13.17, caption ———— 24 Jun 2018

false positive for “trade-offs” ⇝ false positive for “certain”

► **Page 424**, line −11 ———— 24 Jun 2018

and the text ⇝ and of the text (K. Marinakos)

► **Page 426**, line 3 ———— 24 Jun 2018

gives as ⇝ gives us (K. Marinakos)

► **Page 427**, line 7 ———— 24 Jun 2018

we actually wasting ⇝ we are actually wasting (K. Marinakos)

► **Page 428**, line 4 ———— 24 Jun 2018

BABABAABABC ⇝ BABABABCABC (K. Marinakos)

Page 443, algorithm 15.4, line 6 ———— 20 May 2018

$rt[\texttt{Ord}(p[i])] \leftarrow m - i - 1$

⇝

$rt[\texttt{Ordinal}(p[i])] \leftarrow m - i - 1$

Page 443, line −4 ———— 20 May 2018

The function `Ord(c)` ⇝ The function $\texttt{Ordinal}(c)$

Page 445, algorithm 15.5, line 13 ———— 20 May 2018

$i \leftarrow i + rt[\texttt{Ord}(c)]$
$\rightsquigarrow$
$i \leftarrow i + rt[\texttt{Ordinal}(c)]$

► **Page 446**, line −4 to −3 ———— 20 May 2018

The time to create table rt is $O(m)$ $\rightsquigarrow$ The time to create rt is $O(m+s)$

► **Page 446**, line −2 ———— 20 May 2018

longer than m $\rightsquigarrow$ longer than $m+s$

Page 456, line 10 ———— 20 May 2018

But a whole lot more of them. $\rightsquigarrow$ But a whole lot more of them before it starts repeating itself.

► **Page 463**, line −4, ———— 20 May 2018

from a *scr* $\rightsquigarrow$ from a source *src*

► **Page 463**, lines −3, −1 ———— 20 May 2018

scr $\rightsquigarrow$ *src*

► **Page 464**, algorithm 16.5 signature, input, output, lines 1, 3, 5 ———— 20 May 2018

scr $\rightsquigarrow$ *src*

► **Page 464**, line 1 ———— 20 May 2018

creating s $\rightsquigarrow$ creating S

► **Page 464**, lines 2, 4, −6 ———— 20 May 2018

scr $\rightsquigarrow$ *src*

► **Page 464**, line −6 ———— 20 May 2018

we return s $\rightsquigarrow$ we return S

► **Page 478**, figure 16.7, line 2 ———— 23 May 2018

F F T T T T T T T T T T T T F F T T T T T T T T T T T T T T T T

$\rightsquigarrow$

F F T T T T T T T T T T T T T T T T T T T T T T T T T T T T T T

► **Page 484**, algorithm 16.10, output ———— 23 May 2018

with probability $(1/4)^t$ $\rightsquigarrow$ with error probability $(1/4)^t$

Page 491, reference 64 ——— 07 Aug 2018

08 1989 ⇝ August 1989

Page 491, reference 677 ——— 07 Aug 2018

11 2002 ⇝ November 2002

► **Page 504**, first column ——— 11 Mar 2019

add `GetRoot`

► **Page 507**, second column ——— 11 Mar 2019

remove `Root`

► **Page 508**, first column ——— 11 Mar 2019

add `SetRoot`