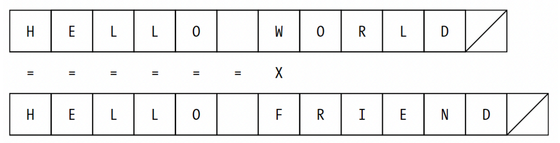
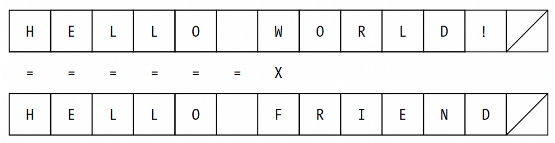
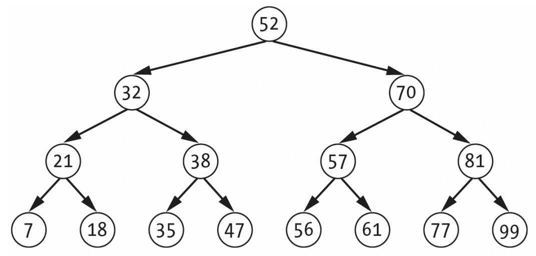
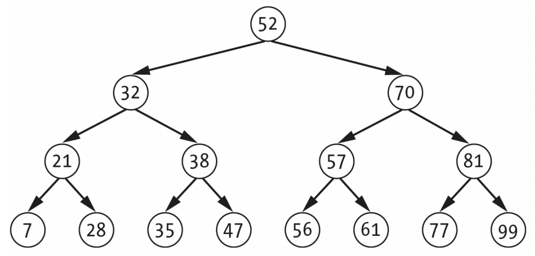
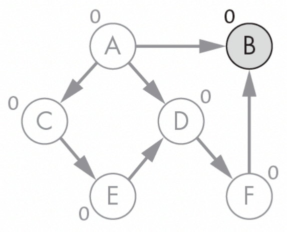
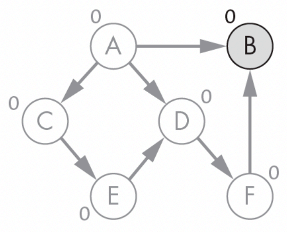
Data Structures the Fun Way

An Amusing Adventure with Coffee-Filled Examples

by Jeremy Kubica

Errata updated to print 2

Page	Error	Correction	Print corrected
8	`InsertionSort(array: A):`	`InsertionSort(Array: A):`	Pending
11	 H E L L O W O R L D = = = = = = X H E L L O F R I E N D *Figure 1-9: A comparison of two strings*	 H E L L O W O R L D ! = = = = = = X H E L L O F R I E N D *Figure 1-9: A comparison of two strings*	Pending
34	Done naively, this approach requires traversing the entire **array** to reach the end, but, as we will see in the next chapter, there are ways to avoid this additional cost.	Done naively, this approach requires traversing the entire **linked list** to reach the end, but, as we will see in the next chapter, there are ways to avoid this additional cost.	Pending
41	In the next chapter, we will show how can build on these fundamental concepts to create two data structures, stacks and queues, that enable different behavior.	In the next chapter, we will show how **we** can build on these fundamental concepts to create two data structures, stacks and queues, that enable different behavior.	Pending
58	 52 32 70 21 38 57 81 7 18 35 47 56 61 77 99 *Figure 5-4: The values of the nodes in a binary search tree are ordered by the binary search tree property.*	 52 32 70 21 38 57 81 7 28 35 47 56 61 77 99 *Figure 5-4: The values of the nodes in a binary search tree are ordered by the binary search tree property.*	Pending
169	We start by **competing** the minimum distance from the point to the node's spatial bounds along each individual dimension:	We start by **computing** the minimum distance from the point to the node's spatial bounds along each individual dimension:	Pending

Page	Error	Correction	Print corrected
200	Formally, we define the size of B-tree node with a size parameter *k*, which provides bounds on how many elements a non-root node can store.	Formally, we define the size of **a** B-tree node with a size parameter *k*, which provides bounds on how many elements a non-root node can store.	Pending
205	We'll explore the later approach, which results in a two-stage algorithm for inserting new keys.	We'll explore the **latter** approach, which results in a two-stage algorithm for inserting new keys.	Pending
222	B-trees combine indexing and storage in such a way as minimize the number of accesses we need.	B-trees combine indexing and storage in such a way as **to** minimize the number of accesses we need.	Pending
227	Let's examine how we can extend the hashing techniques we learned Chapter 10 to this prefiltering question.	Let's examine how we can extend the hashing techniques we learned **in** Chapter 10 to this prefiltering question.	Pending
262	 Next: A Sorted: A,C,E,D,F,B (7) *Figure 15-11: A topological sort on a directed acyclic graph*	 Next: Sorted: A,C,E,D,F,B (7) *Figure 15-11: A topological sort on a directed acyclic graph*	Pending