cphbusiness

COPENHAGEN BUSINESS ACADEMY

Algorithms and Data Structure Summary

Sorting Algorithms

Algorithms	Basic description	Recursive	Time complexity
Quicksort	Assign a pivot and divide the array into two subarray. One less than pivot and one greater than pivot.	Yes	O(n^2) – worst case O(n log(n)) - average
Merge sort	Recursively divide the array into subarray and sor t the suparray	Yes	O(n log(n))
Bubble sort	Repeatedly move the largest element to the highest index position but successive adjacent pairs are checked	No	O(n^2)
Insertion sort	Repeatedly take an element from the array and inserted into the sorted array. Sorted part of the array grows	No	O(n^2)
Selection sort	we repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array. Searches the whole array. In place comparison	No	O(n^2)
Heap sort	Array is represted into a abstract heap data structure. Then heapify the heap to identify sorted part of the array.	Yes	O(n log(n))

Data structure

Type	**Description**
Array/ArrayList	One static and one dynamic
Linked List	Each element is a separate object called node in a list. Each node has two elements a data and reference to the next node. The last node has a reference to null. The entry point into a **linked list** is called the head of the **list**.
Binary Tree	a binary tree is a tree data structure in which each node has at most two children -left child and the right child. PreOrder traversal - visit the parent first and then left and right children; InOrder traversal - visit the left child, then the parent and the right child; PostOrder traversal - visit left child, then the right child and then the parent;
Binary Search Tree	A Binary Tree that follows following condition: left child node is smaller than its parent Node right child node is greater than its parent Node. Traversal same as above.
HashTable	Associative array. It maps key to a value. Eacy key generates an index based on a hash-function.

Data structure

	Access	Search	Insert	Delete	Sorting
Array/ArrayList	O(1)	O(n)	O(n)	O(n)	Quicksort for primitive type and mergesort for objects
Linked List	O(n)	O(n)	O(1)	O(1)	mergesort
Binary Search Tree	O(n) O(log(n)) -average	O(n) O(log(n))-average	O(n) O(log(n))-average	O(n) O(log(n))-average	Not required
HashTable	O(1)	O(1)	O(1)	O(1)	Not Required

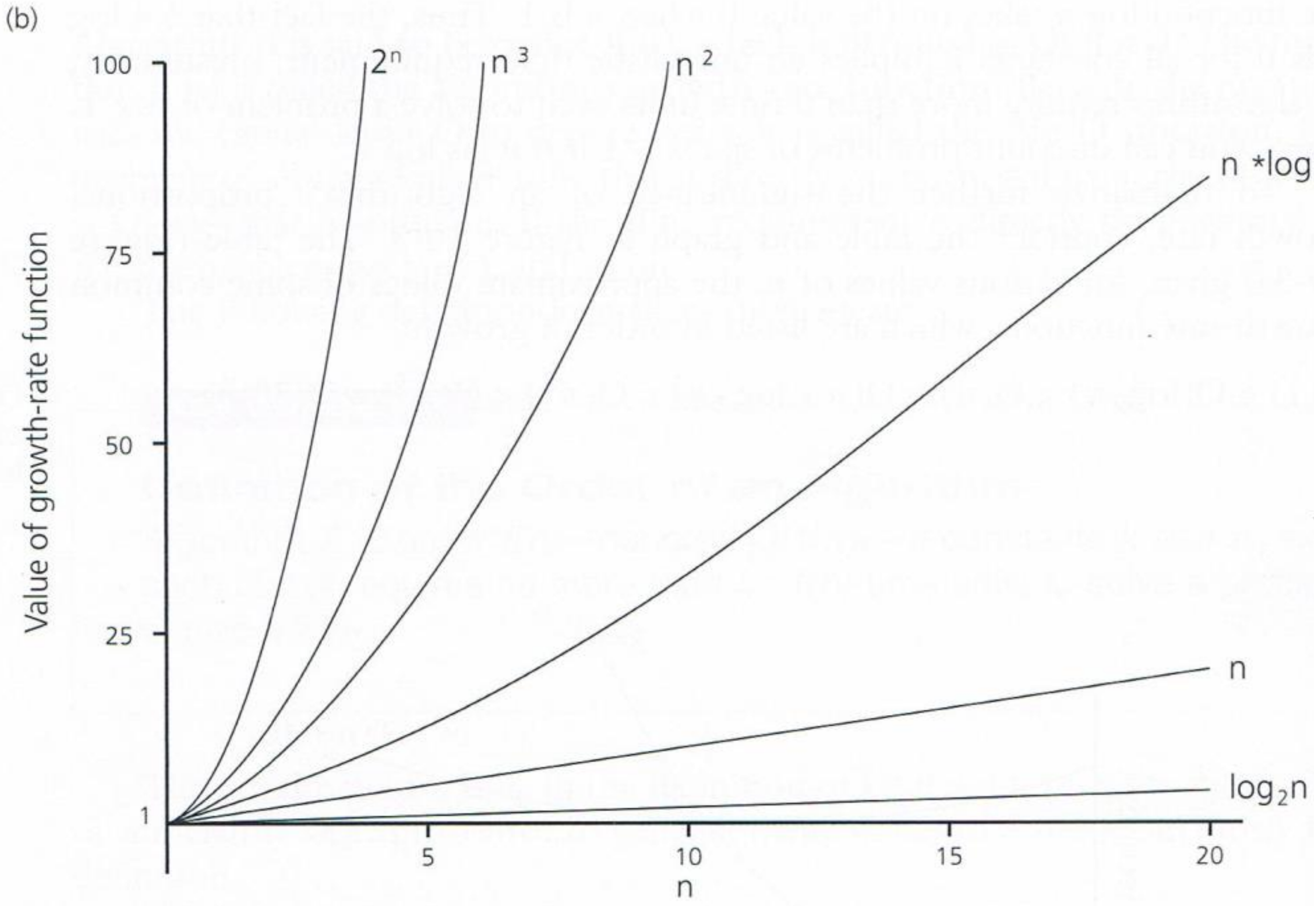
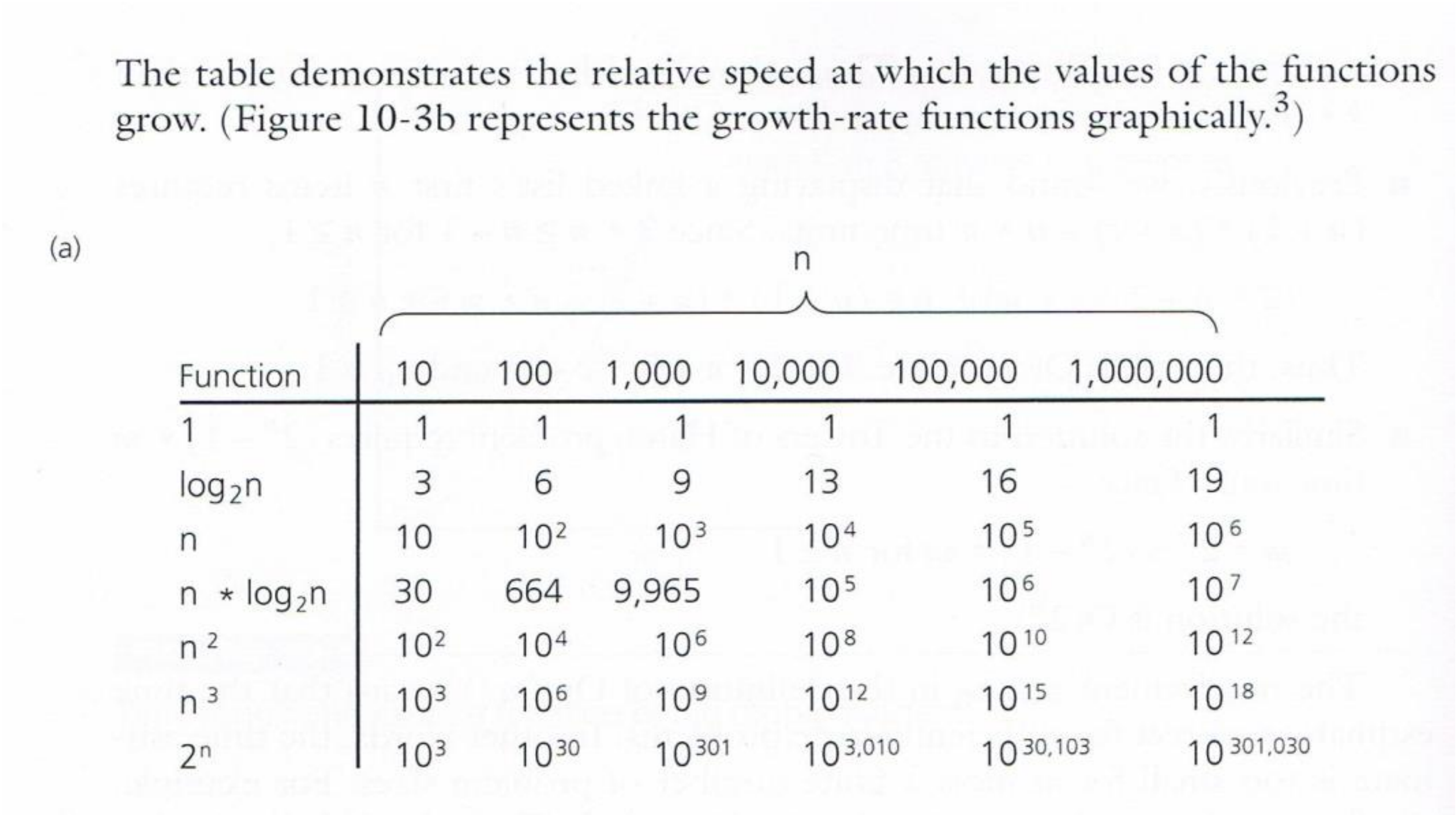


FIGURE 10-3

A comparison of growth-rate functions: (a) in tabular form; (b) in graphical form

[3] The graph of $f(n) = 1$ is omitted because the scale of the figure makes it difficult to draw. It would, however, be a straight line parallel to the x axis through $y = 1$.

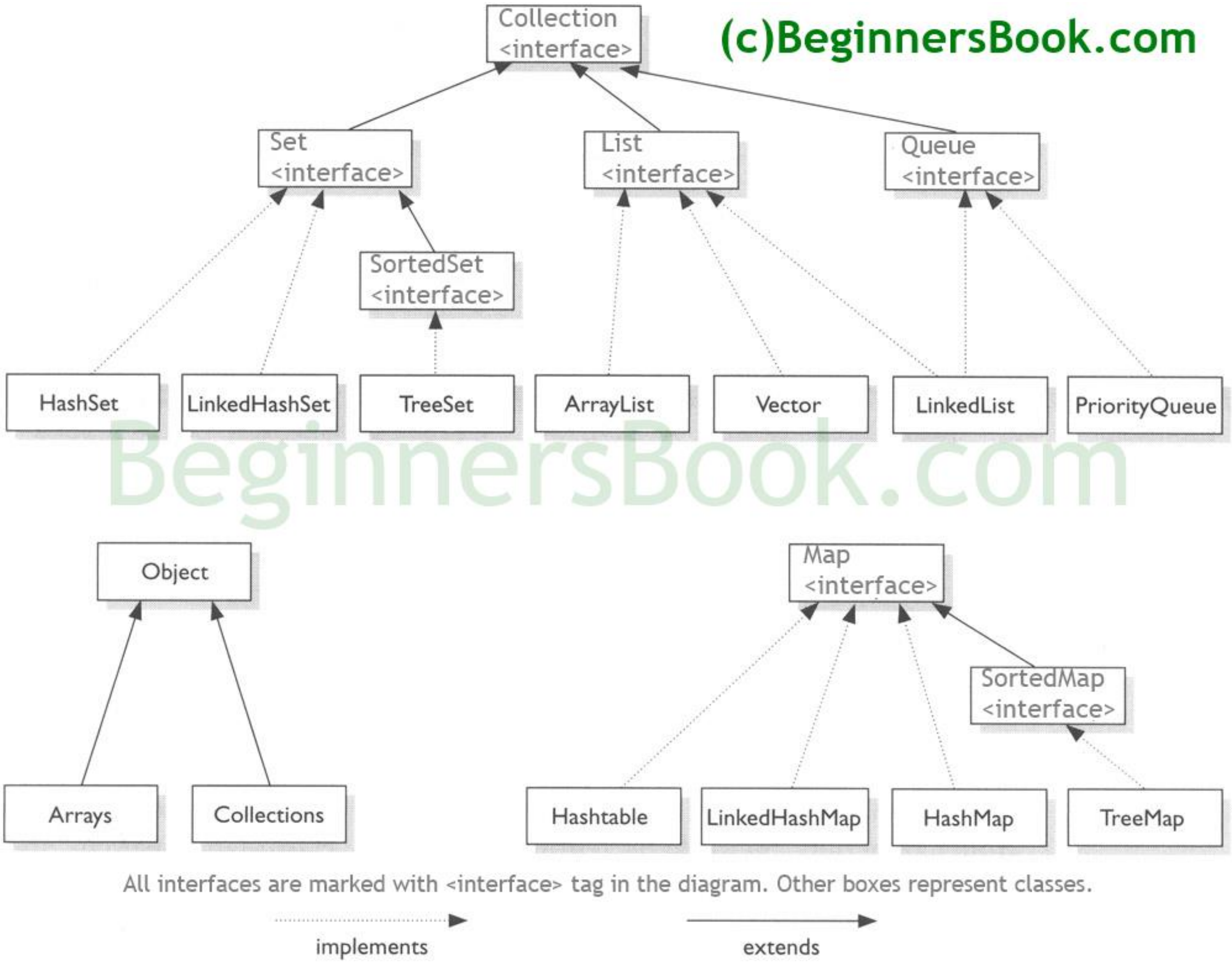


The table demonstrates the relative speed at which the values of the functions grow. (Figure 10-3b represents the growth-rate functions graphically.[3])

(a)

	n					
Function	10	100	1,000	10,000	100,000	1,000,000
1	1	1	1	1	1	1
$\log_2 n$	3	6	9	13	16	19
n	10	10^2	10^3	10^4	10^5	10^6
$n * \log_2 n$	30	664	9,965	10^5	10^6	10^7
n^2	10^2	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{301}	$10^{3,010}$	$10^{30,103}$	$10^{301,030}$

Java Collection Framework


Collection <interface>
(c)BeginnersBook.com
Set <interface>
List <interface>
Queue <interface>
SortedSet <interface>
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue
BeginnersBook.com
Object
Map <interface>
SortedMap <interface>
Arrays
Collections
Hashtable
LinkedHashMap
HashMap
TreeMap
All interfaces are marked with <interface> tag in the diagram. Other boxes represent classes.
implements
extends