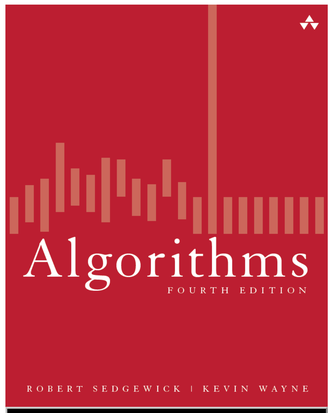

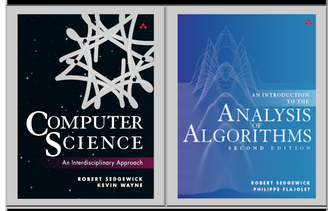

3. SEARCHING

Overview. Modern computing and the internet have made accessible a vast amount of information. The ability to efficiently search through this information is fundamental to computation. This chapter describes classical *searching* algorithms that have proven to be effective in numerous applications for decades. We use the term *symbol table* to describe an abstract mechanism where we save information (a *value*) that we can later search for and retrieve by specifying a *key*.

- *3.1 Elementary Symbol Tables* includes unordered and ordered implementations, using arrays or linked lists.
- *3.2 Binary Search Trees* describes binary search trees.
- *3.3 Balanced Search Trees* describes red-black BSTs, a data structure that guarantees logarithmic performance per symbol table operation.
- *3.4 Hash Tables* describes two classic hashing algorithms: separate chaining and linear probing.
- *3.5 Applications* introduces the set data type and includes numerous applications of symbol tables and sets.

Java programs in this chapter. Below is a list of Java programs in this chapter. Click on the program name to access the Java code; click on the reference number for a brief description; read the textbook for a full discussion.

REF	PROGRAM	DESCRIPTION / JAVADOC
-	FrequencyCounter.java	frequency counter
3.1	SequentialSearchST.java	sequential search
3.2	BinarySearchST.java	binary search
3.3	BST.java	binary search tree
3.4	RedBlackBST.java	red-black tree
3.5	SeparateChainingHashST.java	separate chaining hash table
3.6	LinearProbingHashST.java	linear probing hash table
-	ST.java	ordered symbol table
-	SET.java	ordered set
-	DeDup.java	remove duplicates
-	AllowFilter.java	allowlist filter
-	BlockFilter.java	blocklist filter
-	LookupCSV.java	dictionary lookup
-	LookupIndex.java	index and inverted index
-	FileIndex.java	file indexing
-	SparseVector.java	sparse vector

Last modified on August 26, 2016.