Sergio E. Garcia Tapia
Algorithms by Sedgewick and Wayne (4th edition) [SW11]
November 10th, 2024

3.1: Symbol Tables

Exercise 1. Write a client that creates a symbol table mapping letter grades to numerical scores, as in the table below, then reads from standard input a list of letter grades and computes and prints the GPA (the average of the numbers corresponding to the grades).

A+	A	A-	B+	B	B-	C+	C	C-	D	F
4.33	4.00	3.67	3.33	3.00	2.67	2.33	2.00	1.67	1.00	0.00

Solution. See `com.segarciat.algs4.ch3.sec1.ex01.GPA`

Exercise 2. Develop a symbol-table implementation `ArrayST` that uses an (unordered) array as the underlying data structure to implement our basic symbol-table API.

Solution. See `com.segarciat.algs4.ch3.sec1.ex02.ArrayST`.

Exercise 3. Develop a symbol-table implementation of `OrderedSequentialSearchST` that uses an ordered linked list as the underlying data structure to implement our ordered symbol-table API.

Solution. See `com.segarciat.algs4.ch3.sec1.ex03.OrderedSequentialSearchST`.

Exercise 4. Develop `Time` and `Event` ADTs that allow processing of data as in the example illustrated on page 367.

Solution. See `com.segarciat.algs4.ch3.sec1.ex04.Time`. The class is immutable, and it it implements `Comparable<Time>`, so that it has a natural order. I was unclear about what an `Event` ADT would include, so I did not provide an implementation for this ADT.

Exercise 5. Implement `size()`, `delete()`, and `keys()` for `SequentialSearchST`.

Solution. See `com.segarciat.algs4.ch3.sec1.ex05.SequentialSearchST`.

Exercise 6. Give the number of calls to `put()` and `get()` issued by `FrequencyCounter`, as a function of the number W of words and the number D of distinct words in the input.

Solution. The following assumes that the minimum length accepted for a word is 1,.

During the first phase, the program builds the symbol tables by processing all W words. For each word, there is a call to `contains()`, for a total of W such calls. Since a call to `put()` is made regardless of the result, there are W calls to `put()` during this phase. Each result of `false` from the call to `contains()` corresponds to a distinct word, so there are D such outcomes. Thus, there are $W - D$ direct calls to `get()` in the branch of the `if-else`, where `get()` is used to retrieve the count of a previously-seen word. Note also that each call to `contains()` leads to a call to `get()`, accounting for W more calls.

In the second phase, one addition call to `put()` is made, which enters the empty string, so that there are now $D+1$ keys in the symbol table. In the loop, 2 calls to `get()` are made in each iteration, for a total of $2(D+1)$ calls. A final call to `get()` is made after the loop.

Thus, if f is the number of calls made to `put()`, and g is the number of calls made to `get()`, then

$$\begin{aligned} f(W,D) &= W+1 \\ g(W,D) &= W+(W-D)+2(D+1)+1 \\ &= 2W+D+3 \end{aligned}$$

Exercise 7. What is the average number of distinct keys that `FrequencyCounter` will find among N random nonnegative integers less than 1,000, for $N = 10, 10^2, 10^3, 10^4, 10^5$, and 10^6?

Solution. Consider the random experiment of picking N integers at random, where each integer is between 1 and 1,000, and is chosen independently of the other. Then each outcome is an N-tuple, where each component is an integer between 1 and 1,000. Let X be a random variable that counts the number of distinct keys in an N-tuple. Then X is a discrete random variables, whose values range from 1 through $\min\{N, 1000\}$.

Consider the number of outcomes with $X = k$ distinct integers. If k is not an integer, or $k > \min\{N, 1000\}$, or $k \leq 0$, then there are 0 such outcomes. Otherwise, we can count in two steps:

(i) Choose k distinct integers. There are $\binom{1000}{k}$ ways of doing this, for $1 \leq k \leq 1000$, and 0 for $k > 1000$.

(ii) Of the k integers chosen, choose $N-k$ integers to fill in the remaining spots. There are k^{N-k} ways to do this.

(iii) Permute the N chosen integers, in $N!$ ways.

By the multiplication principle of counting, we find that there are $\binom{1000}{k} \cdot k^{N-K}$ ways to choose k distinct integers when choosing a total of N integers. There are a total of 1000^N outcomes. Since every outcome is equally likely,

References

[SW11] Robert Sedgewick and Kevin Wayne. *Algorithms*. 4th ed. Addison-Wesley, 2011. ISBN: 9780321573513.