

Prepare your answers to the following questions in a plain text file. Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

One way to compare the performance of several hash functions is to simulate the addition of a fixed collection of records to a hash table, using each hash function, and counting the number of hits each slot in the table receives. From that information, we can construct a histogram showing how many slots received one hit, how many received two hits, and so forth. We can also calculate the average number of steps that a search would take.

For instance, given a collection of 1000 words and a hash function that computes a nonnegative integer from a string, we might use a table size of 2000 and discover that:

# of hits	# of slots receiving that # of hits
0	1042
1	932
2	15
3	7
4	3
5	1

Assume that each slot in the hash table uses a linked list to hold the elements that collide in that slot. From that information, we would then calculate that the average number of steps in a search would be about 1.064, since the histogram above implies that:

- 958 words would require 1 step ($932 + 15 + 7 + 3 + 1$)
- 26 words would require 2 steps
- 11 words would require 3 steps
- 4 words would require 4 steps
- 1 words would require 5 steps

Three files of strings are provided on the course website:

```
WordData.txt    111,444 English words (somewhat loosely)
D2.txt          30,573 medium-length strings describing proteins
SQ2.txt         26,381 (mostly) long strings representing amino acid sequences
```

Your task is to compare the performance of four different hash functions on the latter two files:

```
sum()           simply sums the ASCII codes of the characters in the string
elfhash()       classic elfhash function employing complex folding
sfold()         string hasher employing simple folding
sbdm()          string hasher employing more complex folding
```

Java code for each of the four functions is provided in the course notes or on the course website. You will implement driver code to apply each of the hash functions to all of the strings in each of the latter two files and compute the results needed to complete the table given below. The results for the first file are included below to help you validate your logic.

In accordance with common guidelines for achieving good performance, you will use a table size equal to twice the number of strings being hashed into it.

You should not that it is NOT necessary to implement a fully-functional hash table in order to solve this assignment. In fact, it may be counter-productive to do so. You may, and should, take advantage of the standard library.

Hash Function	WordData.txt	D2.txt	SQ2.txt
	111444 strings	30573 strings	26381 strings

sum			
avg # steps	90.494		
worst case	373		
% < log N	17.7%		

elf			
avg # steps	1.321		
worst case	6		
% < log N	100.0%		

sfold			
avg # steps	1.257		
worst case	6		
% < log N	100.0%		

sbdm			
avg # steps	1.251		
worst case	6		
% < log N	100.0%		

The average number of steps is computed in the manner described in the example given on the previous page.

The worst case is the maximum number of steps that would be needed to find a word; it would be 5 for the example.

The last value (% < log N) is the percentage of the given strings that would be found in fewer than log N steps, where N is the number of strings that were hashed.