Bioinformatics Group
Prof. Dr. Rolf Backofen
Florian Eggenhofer
Michael Uhl
Stefan Mautner

Algorithms and Datastructures
WS 2017 / 2018

http://www.bioinf.uni-freiburg.de/Lehre/

UNI FREIBURG

Exercise sheet 5

Deadline: Tuesday, 28.11.2017 11:00 AM

Let $U = \{0, 1, 2, \ldots, N-1\}$ be a universe of N potential keys, p a prime number $\geq \|U\|$ and m the size of the hash table with $p > m$. We have seen in the lecture that the class of hash functions $H = \{h_{a,b} : a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\}$ with

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m \tag{1}$$

is c-universal.

In the following exercises you will write a program which demonstrates that this property holds. The amount of code should be relatively minor this time. Most importantly, you need to understand the concept of universality.

Exercise 1 (5 points)
Write a function *mean_bucket_size* which, given a set of keys S and a hash function h, calculates the *mean bucket size* of h applied on S. The mean bucket size is the mean number of elements that you have to look at in the bucket in order to find the wanted key. Attention: For this you only have to average over the buckets with at least one key (i.e. $\|S\|$ divided by the number of non-empty buckets). Why?

Hints:

- Implement the hash function as a class (*class HashFunction*) with the member variables a, b, p (the prime number), u (universe size), m (hash table size). Also implement a method *apply*(x) (apply hash function specified above to a given key value and return hash table index) and a void method *set_random_parameters()* (set random parameters for a and b, no return value)
- *mean_bucket_size* gets a list of keys and a hash function object and returns the calculated mean bucket size

Exercise 2 (5 points)
Write a function *estimate_c_for_single_set*. Given a set of keys S, the method calculates the mean bucket size for 1000 random hash functions and from this calculates the best possible value

of c and returns c. For the calculation of c use the formula from the lecture $E(\|S_i\|) \leq 1 + c \cdot \frac{\|S\|}{m}$. The function receives a list of keys and the hash function object.

Exercise 3 (5 points)

Write a function *estimate_c_for_multiple_sets* which randomly generates a given number n of key sets with a given size k (no duplicates inside one set of keys!) and calculates for each of the n key sets the best possible c with the function *estimate_c_for_single_set* from exercise 2. Based on these the function then should remember the mean, minimum and maximum c value and finally return the three values. The function receives n, k and the hash function object. Implement an additional function *create_random_universe_subset()* for the random key list generation. It receives k and the universe size u from the hash function and returns the subset list.

Exercise 4 (5 points)
Write a program which calculates the values described in exercise 3 for $\|U\| = 100$, $m = 10$, $p = 101$ and $n = 1000$ randomly chosen key sets of size $k = 20$. Next simulate a non-universal hash function by choosing $p = 10$ and based on this again calculate the values described in exercise 3. Report your calculated values in your *erfahrungen.txt* file. Also shortly explain whether your results make sense and if so why.

Commit
Commit your code into the SVN in a new subdirectory **uebungsblatt_05** and a PDF with the solutions of the theoretical tasks in the same folder. Commit your feedback the text file *erfahrungen.txt* as usual. Please specify: The length of time needed for the exercise. Which tasks have been difficult for you and where did you have problems? How much time did you spend to solve the problems?