University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Schneider

Algorithms and Data Structures
Winter Term 2019/2020
Sample Solution Exercise Sheet 3

Remark: For this exercise sheet, watch the fourth and fifth video lecture given on the lecture website.

Exercise 1: Case Based Runtime Analysis

The following algorithm obtains a number $x \in \{0, ..., n\}$. Additionally it obtains an array A of size $n+1$ that contains integers $\{0, ..., n\} \setminus \{x\}$ sorted in *ascending* order, whereas the last entry of A is empty. The algorithm inserts x into its position in A and moves the subsequent elements by one position.

Algorithm 1 INSERT($A[0..n]$, x)

```
$i \leftarrow n$
while $i \geq 0$ and $A[i-1] > x$ do
    Swap $A[i-1]$ and $A[i]$
    $i \leftarrow i - 1$
$A[i] \leftarrow x$
```

Give the runtime of the algorithm *as absolute value and asymptotically* ($\mathcal{O}$-Notation) in the *best, worst and the average case* (compute the average runtime over all possible inputs x).

Remark: To simplify things, assume that one loop cycle takes one time unit, while all other operations have negligible runtime.

Sample Solution

In the best case we have $x = n$ and the while loop terminates immediately. While we have no loop cycles, we still have $\mathcal{O}(1)$ runtime in the asymptotic sense. In the worst case we have $x = 1$ and x must be swapped all the way down to position 1 in A, taking $n-1 \in \mathcal{O}(n)$ loop cycles.
The number of loop cycles is equal to $n - x$. We have $n+1$ possibilities for choosing $x \in \{0, ..., n\}$. Therefore the average runtime (i.e., the average number of loop cycles) is

$$T_{\text{avg}}(n) = \frac{\sum_{x=0}^{n}(n-x)}{n+1} = \frac{\sum_{x=0}^{n} x}{n+1} = \frac{n(n+1)}{2(n+1)} = \frac{n}{2} \in \mathcal{O}(n).$$

Exercise 2: Unsuitable Hash Functions

Let m be the size of a hashtable and let $n \gg m$ be the biggest possible key of any (key,value)-pair. A hash function $h : \{0, \ldots, n\} \to \{0, \ldots, m-1\}$ maps keys to table entries and should meet some criteria to be considered suitable.

The hash function should utilize the whole table, i.e., it should be a surjective function. Furthermore, it should be "chaotic", meaning that it should map similar keys to distinct table entries in order to avoid having lots of collisions in case many similar keys are inserted. A hash function must be deterministic. The following "hash functions" are unsuitable for various reasons. For each hash function quickly explain why this is the case.

(a) $h_1 : k \mapsto k$. [1]

(b) $h_2 : k \mapsto \lfloor \frac{k}{n} \cdot (m-1) \rfloor$.

(c) $h_3 : k \mapsto 2 \cdot \big(k \bmod \lfloor \frac{m}{2} \rfloor\big)$.

(d) $h_4 : k \mapsto \texttt{random}(m)$, $\big(\texttt{random}(m)$ is picked uniformly at random from $\{0, \ldots, m-1\}\big)$.

Sample Solution

(a) The given function is not even a valid hash-function, since it maps to values outside the table range.

(b) This hash function is not seperating similar valus well, that is if k_1, k_2 are close, then $h_2(k_1), h_2(k_2)$ are close as well. Since $n \gg m$ many successive keys are mapped to the same table entry.

(c) The function is not surjective, i.e. it does not use the whole address space (e.g. only even table entries are used, and the entry $m-1$ is never used). For $m = 1$ the function is undefined (since mod 0 is undefined).

(d) A truly randomized function prohibits finding a value efficiently once it has been hashed.

Exercise 3: Not a Universal Family of Hash Functions

Let p be prime and $K := \{0, \ldots, p-1\}$ be a set of keys. Consider the following set of hash functions.

$$\mathbb{H} := \big\{h_{a,b} \mid a, b \in \{0, \ldots, p-1\}\big\} \text{ with}$$
$$h_{a,b}(x) := (ax + b) \bmod m,$$

whereas $p \gg m$.

(a) Show that there exists a subset $S \subseteq K$ of size $\lceil \frac{p}{m} \rceil$ such that $h(x) = h(y)$ for any pair of keys $x, y \in S$ and any function $h \in H$.

(b) Argue that $\mathbb{H}$ is not c-universal for any constant $0 < c < m$.

Sample Solution

(a) Define $S := \{0, m, 2m, \ldots, \lfloor \frac{p}{m} \rfloor m\}$. For the biggest key in S we have $\lfloor \frac{p}{m} \rfloor m \leq m-1$ since p is prime (hence the number in the floor function is not integer). The set S is sufficiently large, since $|S| = \lfloor \frac{p}{m} \rfloor + 1 \geq \lceil \frac{p}{m} \rceil$. Let $x = i \cdot m$ and $y = j \cdot m$ be two keys in S with $0 \leq i, j \leq \lfloor \frac{p}{m} \rfloor$ and $h_{a,b} \in \mathbb{H}$. Then

$$\begin{aligned} h_{a,b}(x) &= ax + b \bmod m = aim + b \bmod m = b \bmod m \\ &= ajm + b \bmod m = ay + b \bmod m = h_{a,b}(y). \end{aligned}$$

(b) The definition of a c-universal family for $c > 0$ is that for any pair of keys $x, y \in K$ with $x \neq y$ and for uniformly random $h \in \mathbb{H}$ we have

$$\mathbb{P}\big(h(x) = h(y)\big) \leq \frac{c}{n}.$$

But as we have seen in part (a), for $x, y \in S$ with $x \neq y$ we have $\mathbb{P}\big(h(x) = h(y)\big) = 1 > \frac{c}{m}$ for any $h \in \mathbb{H}$, since the values in S always collide. Hence $\mathbb{H}$ is not c-universal.

[1]The notation $h : k \mapsto h(k)$ means h maps the value k to the value $h(k)$.