University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithms and Datastructures
Winter Term 2022
Grading Guidelines Exercise Sheet 4

Due: Wednesday, November 16th, 2pm

Exercise 1: Hashing with Open Addressing *(10 Points)*

We consider hash tables with open addressing and two different methods for collision resolution: *linear probing* and *double hashing*. Let m be the size of the hash table where m is prime. Let $h_1(x) := 53 \cdot x$ and $h_2(x) := 1 + (x \bmod (m-1))$. We define the following hash functions for collision resolution according to the lecture:

- linear probing: $h_\ell(x,i) := (h_1(x) + i) \mod m$.
- double hashing: $h_d(x,i) := (h_1(x) + i \cdot h_2(x)) \mod m$.

(a) Implement a hash table with operations `insert` and `find` using the mentioned strategies for collision resolution[1]. You may use the template `HashTable.py`. *(5 Points)*

(b) Create a hash table of size $m > 1000$ (m prime) and measure the average time for inserting k keys for $k \in \{\lfloor \frac{m \cdot i}{50} \rfloor \mid i = 1, \dots, 49\}$ in four variations: Using linear probing / double hashing; inserting k random keys[2] / the set of keys $\{m \cdot i \mid i = 1, \dots, k\}$. Create a plot showing the four different average runtimes. Discuss your results. *(5 Points)*

Exercise 2: Application of Hashtables *(10 Points)*

Consider the following algorithm:

Algorithm 1 `algorithm` ▷ Input: Array A of length n with integer entries

```
for i = 1 to n − 1 do
    for j = 0 to i − 1 do
        for k = 0 to n − 1 do
            if |A[i] − A[j]| = A[k] then
                return true
return false
```

(a) Describe what `algorithm` computes and analyse its asymptotical runtime. *(3 Points)*
Hint: The difference $|A[i] - A[j]|$ may become arbitrarily large.

(b) Describe a different algorithm $\mathcal{B}$ for this problem (i.e., $\mathcal{B}(A) =$ `algorithm`(A) for each input A) which uses hashing and takes time $\mathcal{O}(n^2)$ (with proof). *(3 Points)*

Hint: You may assume that inserting and finding keys in a hash table needs $\mathcal{O}(1)$ if $\alpha = \mathcal{O}(1)$ (α is the load of the table).

(c) Describe another algorithm for this problem without using hashing which takes time $\mathcal{O}(n^2 \log n)$ (with proof). *(4 Points)*

[1] You can assume that no more than m elements will be inserted to the hash table.
[2] Unique random values from $\{0, \dots, z\}$ with $z \gg m$, e.g., with `random.sample(range(z+1), k)`.