



Algorithms and Datastructures

Exercise Sheet 4

Exercise 1: Hashing with Open Addressing (5 Points)

Let $\mathcal{H}$ be a hash table of size $m = 13$ and let $h_1, h_2, h_3 : \mathbb{N}_0 \mapsto \{0, \dots, m-1\}$ be hash functions defined as follows¹:

- $h_1(k) := \overline{k} \bmod m$
- $h_2(k) := 3 \cdot k \bmod m$
- $h_3(k) := k + 1 \bmod m$

Add the keys 23, 12, 75, 945, 30, 99, 345 (in that order) into the initially empty hash table $\mathcal{H}$. Solve conflicts as follows:

- a) Linear Probing using hash function h_1 . (2 Points)
- b) Use Double Hashing using hash functions h_2 and h_3 . (3 Points)

Write down every intermediate step!

Exercise 2: Hashing with Chaining (5 Points)

Given a Hash Table of size m and an arbitrary hash function $h : S \mapsto \{0, \dots, m-1\}$. Let S be a set of at least $y \cdot m$ elements, so $|S| \geq y \cdot m$.

- a) Show that S has a subset Y of at least y elements (hence $|Y| \geq y$) such that $h(x_1) = h(x_2)$ for all $x_1, x_2 \in Y$. (4 Points)
- b) What does the result of a) tell us about the Worst-Case runtime of "find" in a hash table with Chaining (if the table is filled with all the elements of S before we call "find")? (1 Point)

Exercise 3: Application of Hashtables (10 Points)

Consider the following algorithm:

Algorithm 1 algorithm	▷ Input: Array A of length n with integer entries
1: for $i = 1$ to $n - 1$ do	
2: for $j = 0$ to $i - 1$ do	
3: for $k = 0$ to $n - 1$ do	
4: if $ A[i] - A[j] = A[k]$ then	
5: return true	
6: return false	

¹We define the digit sum of k by $\overline{k}$.

- (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., $\mathcal{B}(A) = \text{algorithm}(A)$ 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)