FIT1008 – Intro to Computer Science
Tutorial 11

Semester 1, 2018

Objectives of this tutorial

- To understand Hash Tables.

Exercise 1

Using the following hash function:

```
def hash(input_string):
    return ord(input_string[0]) % 11
```

and linear probing, calculate the hash value of the following names and insert them into a Hash Table of size 11.

```
Eva, Amy, Tim, Ron, Jan, Kim, Dot, Ann, Jim, Jon
```

Note that the ascii value for E is 69, for A 65, for T 84, for R 82, for J 74, for K 75, and for D 68.

Exercise 2

Assume you have completed *Exercise 1* . Illustrate what happens, when you search for the names Jim, Jon and Joe.

Exercise 3

Repeat *Exercises 1 and 2* using Quadratic probing instead of linear probing.

Exercise 4

Using the following function:

```
def hash2(input_string):
    return ord(input_string[0]) % 10 + 1
```

as the second hash function, repeat *Exercise 2* using double hashing instead of linear probing.
Is the second hash function a good choice of function? Discuss in terms of the values provided for keys that are mapped to the same value by the first hash function.

Exercise 5

Consider a HashTable that resolves collisions using linear probing, as follows:

```
from referential_array import build_array

class HashTableLinear:

    prime_list = [25717, 102877, 205759, 411527,
                  823117, 1646237, 3292489, 6584983,
                  13169977]

    def __init__(self):
        self.count = 0
        size = 0
        self. a = 31
        self.table_size = HashTableLinear.prime_list[size]
        self.array = build_array(self.table_size)

    def __setitem__(self, key, value):
        position = self.hash_value(key)
        for _ in range(self.table_size):
            if self.array[position] is None:
                self.array[position] = (key, value)
                self.count += 1
                return
            elif self.array[position][0] == key:
                self.array[position] = (key, value)
                return
            else:
                position = (position+1) % self.table_size
        self.__rehash__()
        self.__setitem__(key, value)
```

Provide an implementation for the method `__rehash__(self)`.

Exercise 6

Discuss the idea behind universal and a perfect hash functions.

Exercise 7

What are advantages and disadvantages of separate chaining over open addressing?