Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Sandro Montanari
Tobias Pröger

13th March 2013

# Datenstrukturen & Algorithmen Programming exercise 4 FS 13

In this exercise we are going to implement open hashing. We are given $n$ keys $k_1, ..., k_n$ to be inserted into an initially empty array of size $m \geq n$, with a prime number $m$. The element $k_i$ is stored in the position

$$h(k_i) = k_i \bmod m$$

if empty. Otherwise we have a collision and $k_i$ has to be inserted in a different position of the array. The new position is determined using probing. After $j$ collisions, we consider the position

$$h(k_i) - s(j, k_i),$$

and we insert $k_i$ there if it is empty. We consider the following probing functions $s(j, k_i)$ introduced at the lecture:

- $s(j, k_i) = j$ (linear probing),
- $s(j, k_i) = \lceil \frac{j}{2} \rceil^2 \cdot (-1)^j$ (quadratic probing),
- $s(j, k_i) = j \cdot h'(k_i)$, where $h'(k_i) = 1 + \big(k_i \bmod (m-2)\big)$ (double hashing).

**Input** The first line of the input contains only the number $t$ of test instances. After that, we have exactly one line per test instance containing the numbers $m, n, k_1, ..., k_n$, where $m \geq n$ is a prime number. We insert each element of the sequence $k_1, ..., k_n$ in this order into three initially empty hash tables of size $m$. The three hash tables use respectively linear probing, quadratic probing and double hashing.

**Output** For every test instance we output three lines. The first line starts with the overall number of collisions caused by the insertion of the sequence in the hash table that uses linear probing and is followed by the content of the hash table after every element has been inserted. If a position in a hash table is empty, we print 0. The second and the third lines are the same as the first, but they contain the collisions and content of the hash tables using quadratic probing and double hashing respectively.

**Example**

*Input:*

```
2
5 3 1 2 3
5 4 1 2 3 8
```

*Output:*

```
0 0 1 2 3 0
0 0 1 2 3 0
0 0 1 2 3 0
3 8 1 2 3 0
1 0 1 2 3 8
1 8 1 2 3 0
```

**Hand-in:** Until Wednesday, 20th March 2013.