This thesis presents some results on *Modern Hashing*, an associated project within the DFG Priority Program 1307 *Algorithm Engineering*.

Funding: German Research Foundation (DFG)

Summary

We study randomized algorithms that take as input a set S of n keys from some large universe U or a set of n key-value pairs, associating each key from S with a specific value, and build a data structure that solves one of the following tasks. On calling the operation `lookup` for a key $x \in U$:

- ▷ Decide set membership with respect to S (membership tester).
- ▷ If $x \in S$, then return the value associated with x. If $x \in U \setminus S$, then return either some specific value $\perp$ interpreted as "$x \notin S$" (dictionary), or some arbitrary value (retrieval data structure).
- ▷ If $x \in S$, then return a natural number associated with x, from a range with size close to n, where for all elements of S the numbers are pairwise distinct. The numbers for elements $x \in U \setminus S$ are arbitrary (perfect hash function).

The data structures that we cover are variations of *cuckoo hashing* and *Bloomier filter* which have the same simple structure. They consist of a table with m cells, each capable of holding entries of size r bits, as well as a constant number of hash functions, which are used to map the elements from U to a constant number of table cells. Assuming *fully random hash functions*, we will discuss how the data structures can be constructed in time linear in n, and what load n/m or space consumption $m \cdot r$ can be achieved in trade-off with the time for `lookup`.
This leads to the question whether a random bipartite graph with n nodes (keys) on the left, m nodes (cells) on the right, and edges determined by the hash functions, asymptotically almost surely has a *matching* of a certain type, and furthermore, how such a matching can be calculated efficiently.
The focus of this thesis concerns:

- ▷ optimizing the lookup time for dictionary and membership tester (based on cuckoo hashing) with respect to:
  - (i) a bound on the average number of cell probes,
  - (ii) the expected number of page accesses in a setting where the table (memory) is subdivided into pages of the same size.
- ▷ minimizing the space usage of the retrieval data structure and perfect hash function (based on Bloomier filter), when for each key the number of cell probes for `lookup` is upper bounded by a constant.
- ▷ an efficient simulation of fully random hash functions as needed by the data structures.

Thresholds for Matchings in Random Bipartite Graphs with Applications to Hashing-Based Data Structures

Michael Rink