# Hashing

**Today:** Chapters 11.1-11.4.

# Motivation

The main motivation for Hashing is improving searching time. Below we show how the search time for Hashing compares to the one for other methods:

- Sequential search: $O(N)$
- B Trees and B+ trees: $O(\log_k N)$
- Hashing: $O(1)$

# What is Hashing ?

A **Hash Function** is a function $h(k)$ that transforms a key into an address. An address space is chosen before hand. For example, we may decide the file will have 1,000 available addresses.

If $U$ is the set of all possible keys, the hash function is from $U$ to {0,1,...,999}, that is

$$h : U \longrightarrow \{0, 1, ..., 999\}$$

Example :

| k | **ASCII code for the first 2 letters** | `product` | h(k) = `product` **mod 1,000** |
|---|---|---|---|
| BALL | 66, 65 | 66.65=4,290 | 290 |
| LOWELL | 76, 79 | 76.79=6,004 | 004 |
| TREE | 84, 82 | 84.82=6,888 | 888 |

| RRN | FILE |
|---|---|
| 000 | |
| 001 | |
| ⋮ | ⋮ |
| 004 | LOWELL |
| ⋮ | ⋮ |
| 290 | BALL |
| ⋮ | ⋮ |
| 888 | TREE |
| ⋮ | ⋮ |
| 999 | |

- There is no obvious connection between the key and the location (randomizing).
- Two different keys may be sent to the same address generating a **Collision**

Can you give an example of collision for the hash function in the previous example ?

LOWELL, LOCK, OLIVER, and any word with first two letters L and O will be mapped to the same address:

$$h(\text{LOWELL}) = h(\text{LOCK}) = h(\text{OLIVER}) = 4.$$

These keys are called **synonyms.** The address "4" is said to be the **home address** of any of these keys.

Avoiding collisions is extremely difficult (remember the birthday paradox discussed in class), so we need techniques for dealing with it.

Ways of reducing collisions:

1. **Spread out the records** by choosing a good hash function.
2. **Use extra memory**, i.e. increase the size of the address space (Ex: reserve 5,000 available addresses rather than 1,000).
3. **Put more than one record at a single address** (use of buckets).

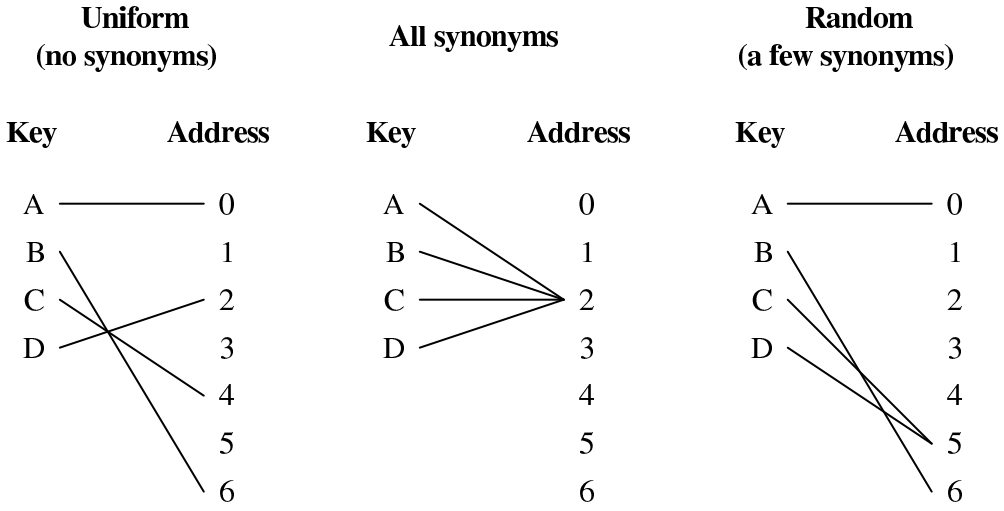
### A Simple Hash Function

To compute this hash function, apply 3 steps :

Step 1 : transform the key into a number.
Step 2 : fold and add (chop off pieces of the number and add them together).
Step 3 : divide by the size of the address space (preferably a prime number).

## Distribution of Records among Addresses

There are 3 possibilities :




Uniform distributions are extremely rare.
Random distributions are acceptable and more easily obtainable.

Trying a **better-than-random** distribution, by preserving natural ordering among the keys :

- Examine keys for patterns.
- Fold parts of the key.
- Use prime number when dividing the key.

When it does not work, try **randomization.** You can using the following Hash functions:

- **Square the key and take the middle**:
  Ex: key = 453 $453^2 = 205209$
  Extract the middle = 52.
- **Radix transformation**:
  Transform the number into another base and then divide by the maximum address.
  Ex: Addresses from 0 to 99
  key = 453 in base 11 : 382
  hash address = 382 mod 99 = 85.