Scuola universitaria professionale
della Svizzera italiana
SUPSI

Università
della
Svizzera
italiana

Istituto Dalle Molle di studi sull'intelligenza artificiale

Algorithms and Data Structures

Hash Tables

Matteo Salani
matteo.salani@idsia.ch

Motivation

How many words in a dictionary?

- In italian there are 21 letters, the longest word is about 30 characteres.
- In principle there can be all combinations of 21 characters to form words from 1 to 30 characters

$$21 + 21^2 + 21^3 + \cdots + 21^{30} \approx 10^{42}$$

- If we put 200 words per page, in a 2000 pages volume we would need $2.5 \cdot 10^{36}$ volumes
- 4 cubic decimeters per volume: $6.5 \cdot 10^{32}\ m^3$
- Volume of planet earth is about $10^{12}\ m^3$

Motivation

How many words in a dictionary?

- Korean $\approx$ 1.100.000 words
- English $\approx$ 470.000 words
- Portuguese $\approx$ 442.000 words
- German $\approx$ 330.000 words
- Italian $\approx$ 260.000 words
- French $\approx$ 135.000 words

There are much less used words than all the possible combinations
Source Wikipedia
https://en.wikipedia.org/wiki/List_of_dictionaries_by_number_of_words

Hash table

- Dynamic set that supports only the dictionary operations INSERT, SEARCH, and DELETE
- A hash table is an effective data structure for implementing dictionaries

Searching for an element in a hash table can take as long as searching for an element in a linked list, in the worst case, in practice, hashing performs extremely well.

Hash table complexity

Under reasonable assumptions, the average search time is O(1)

Hash function

- Is a NON one-to-one function that maps the keys in a set in a set of indexes of a given size.
- Two key can be mapped into the same index. We speak about **collision**
- We aim for a function that minimizes the number of collisions
- Collisions must be managed

Hash function

U elements in the dictionary, m elements in the table

$$h : U \to \{0, 1, \cdots, m-1\}$$

Hash function

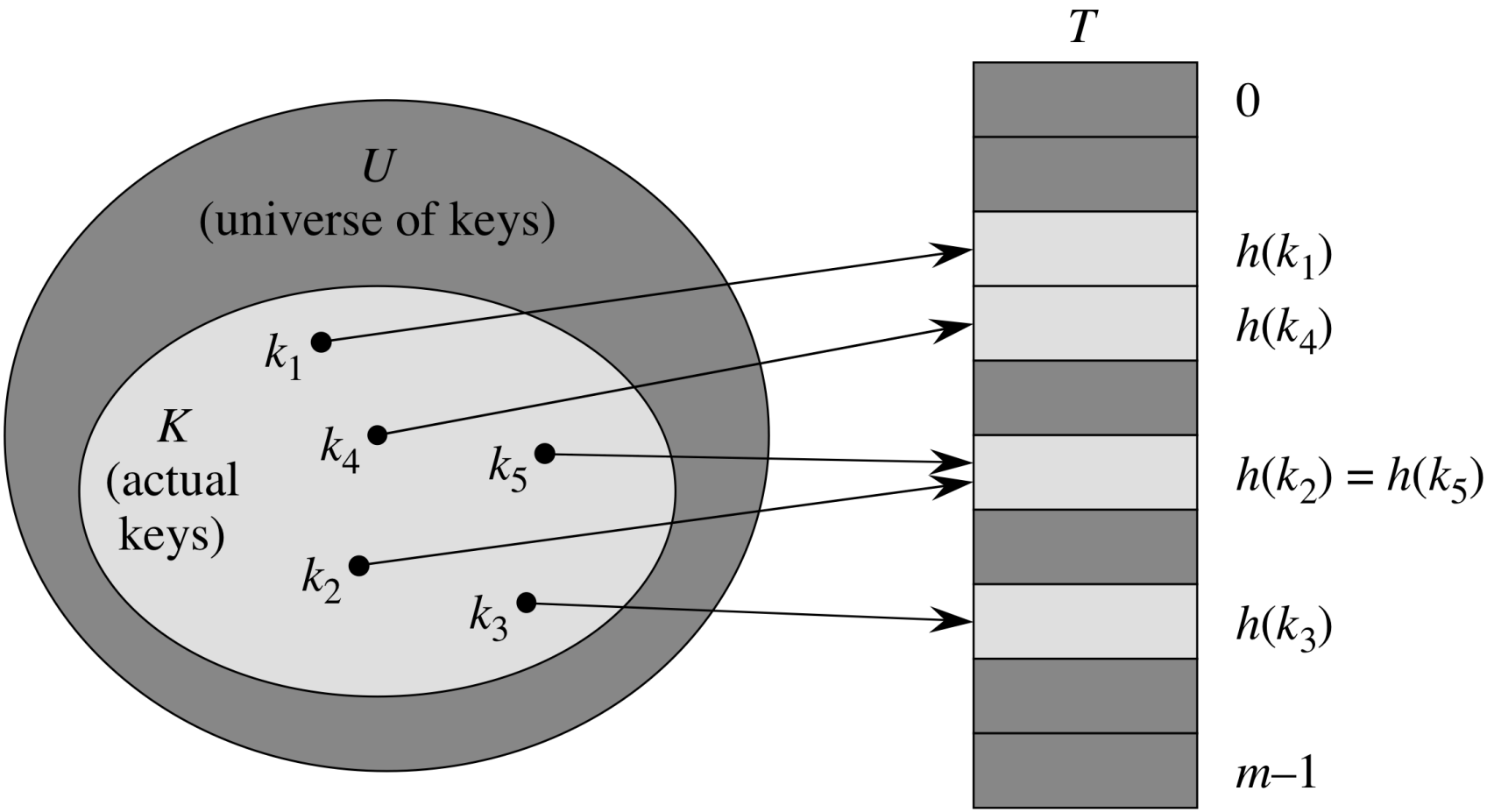

Figure: Hash function with m elements, k_2 and k_5 map in to the same element

Chaining

In case of collisions, one technique is simply to store a list of values. The technique is called **chaining**. The hash table is therefore called **open** hash table.

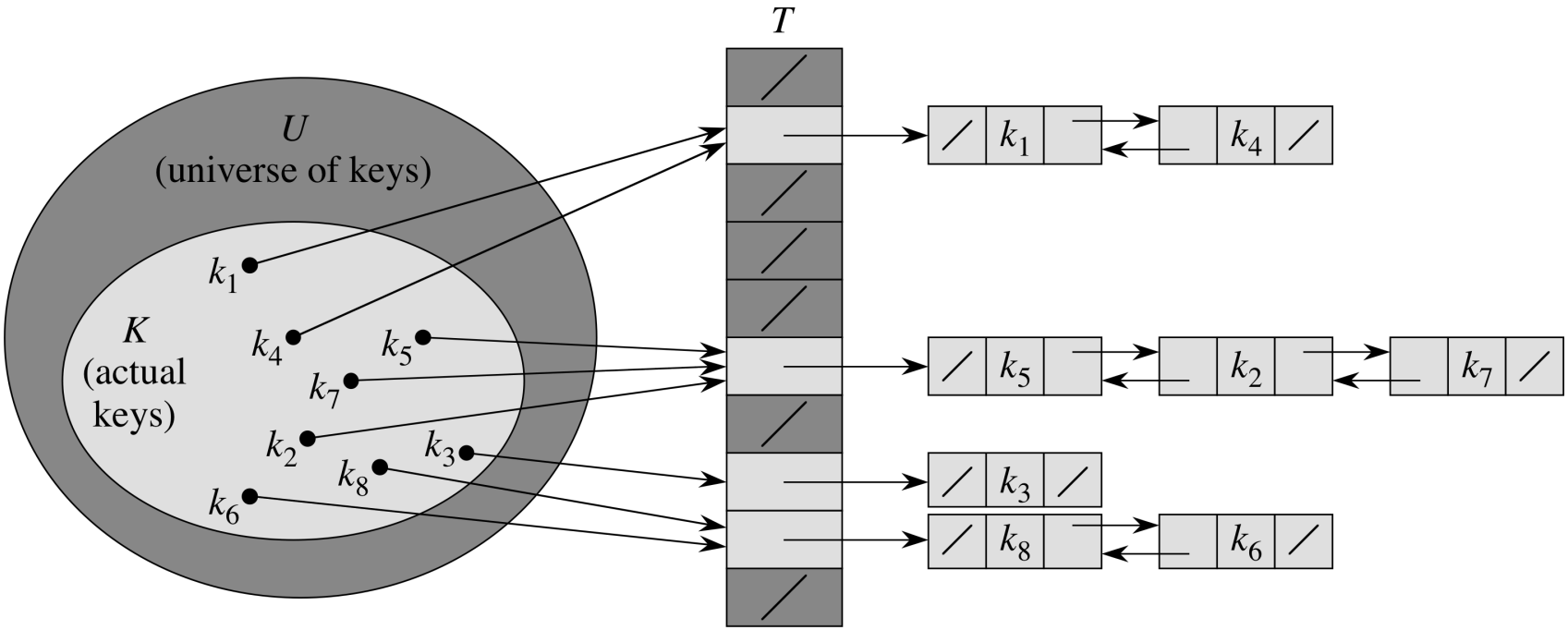

Figure: Multiple values stored with the same key

Chaining

Unfortunately, the worst case scenario for hash tables with chaining is that all keys get the same hash value.

The search time is thus the same of a regular list $O(n)$.

Instead, we are interested in the average case.
It depends on how well the hash function h distributes the set of keys to be stored among the m slots, on the average.

The ratio between keys stored in the hash table and the size of the table is called **load factor**.

Hash functions for integers

Let t be the size of the hash table.

The **division method**

$$h(k) = k \bmod t$$

is a very fast hash function.

Prime numbers not too close to powers of 2 are good candidates for t.

For example, we want to hold 2000 keys and accept an average of 3 elements in an unsuccessful search, so we allocate a hash table of size $m = 701$.
701 is a prime near to 2000/3 but not near any power of 2

Hash functions for integers

The **multiplication method**

$$h(k) = \lfloor t \cdot (k \cdot A \bmod 1) \rfloor$$

Function not sensitive to t.

Choosing t as a power of 2 makes computation very fast.

Value A must be between 0 and 1.
$(\sqrt{5} - 1)/2$ seems a good value in general.

Hash functions for integers

The **Mid-square method**

Compute k^2 and consider the $r = \log_2 t$ central bits

$h(k)$ is therefore in the range $0, \ldots, 2^r - 1$

Here it is reasonable to consider t in powers of 2

Hash functions for strings

One possible hash function for strings s is obtained with the following procedure:

- Divide s in blocks of k characters.
- For each block i composed of characters $c_0^i, c_1^i, \ldots, c_{k-1}^i$ compute

$$p_i = 256^{k-1} \cdot POS(c_{k-1}^i) + 256^{k-2} \cdot POS(c_{k-2}^i) + \ldots$$

$$\ldots + 256^1 \cdot POS(c_1^i) + 256^0 \cdot POS(c_0^i)$$

- Sum the values

$$z = \sum_{i=1}^{\lceil len(s)/k \rceil} p_i$$

- Apply one of the hash functions for integers to value z

Other chaining methods

In case of collision we may apply other hash functions to find a free slot in the dictionary.
In this case the hash table is called **closed** hash table
Linear probing:

$$h_1(k, p) = (h(k) + p) \bmod t \qquad p = 1, 2, \ldots$$

- Iterate until a free slot is found
- p can vary too, e.g. $p_i = p_{i-1} + 1$
- The method may cluster keys around value $h(k)$

Other chaining methods

Quadratic probing:

$$h_1(k,p) = (h(k) + c_1 \cdot p + c_2 \cdot p^2) \bmod t \qquad p = 1, 2, \ldots$$

Pseudo-random probing:

$$h_1(k,p) = (h(k) + p) \bmod t \qquad p = random(1, t-1)$$

The same random number generator (with same seed!) must be used.
Double hashing:

$$h(k,i) = (h_1(k) + i \cdot h_2(k)) \bmod t$$

h_1 and h_2 are auxiliary hash functions.