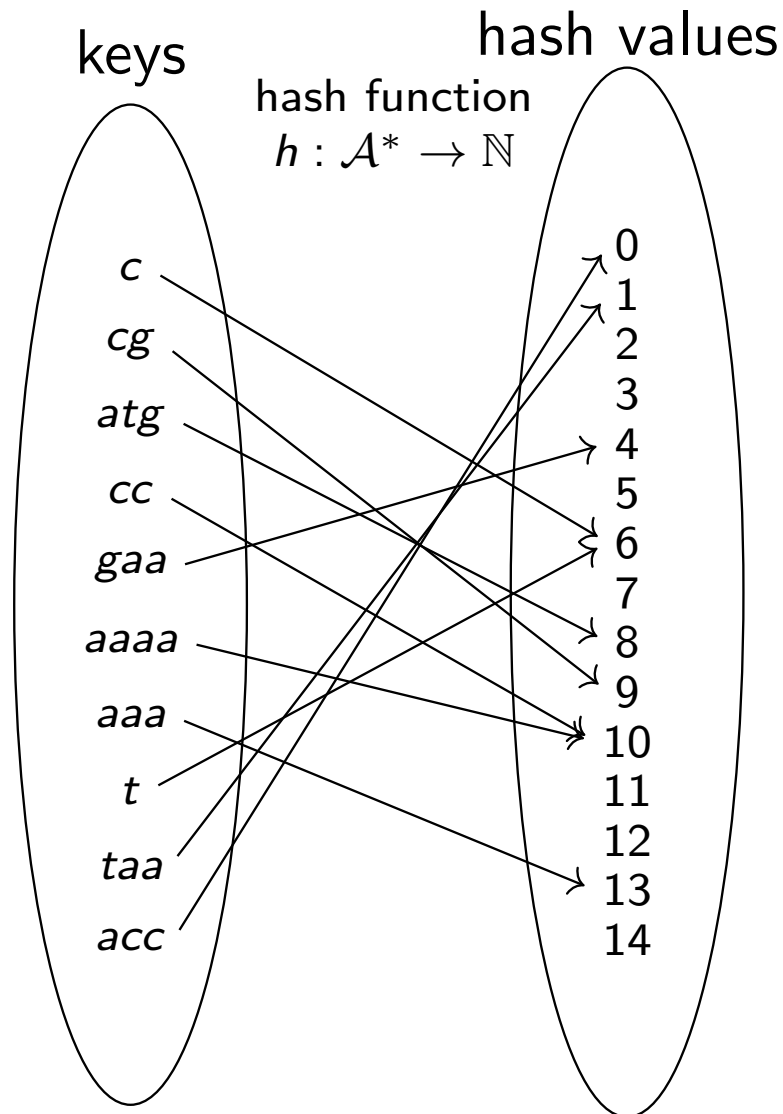


Excursion to hash functions

- one of the very basic tasks in computer science is to efficiently store values associated with keys
- one usually wants to access the value for any key as fast as possible
- a very common way to achieve this is to uniquely associate a key with an index of an array where to store the value
- this association is established by a hash function:
a hash function maps any kind of (hashable) object to a unique integer
- see example for string-keys below

Excursion to hash functions



- collision when $h(w) = h(w')$ for $w \neq w'$, as in $h(cc) = 10 = h(aaaa)$ or $h(c) = 6 = h(t)$
- strategies to solve such conflicts: hashing with chaining, double hashing, open addressing, cuckoo hashing ...
- a hash function is used in a Python-dictionary or a Ruby-Hash or a map in the C++-standard template library
- it is hidden from the user
- Python: obtain hash-value via method `hash`, e.g. `hash('atcg') ⇒ 1 231 534 521 241 347 127`
- can be applied to any hashable object (e.g. strings, numbers, functions)

Examples of hash functions for strings

$js(s) = h_1(s, |s|)$ where

$$h_1(s, i) = \begin{cases} 0 & \text{if } i = 0 \\ (ord(s[i]) + h_1(s, i-1) \cdot 2^5 + h_1(s, i-1)/4) \wedge h_1(s, i-1) & \text{otherwise} \end{cases}$$

$sdbm(s) = h_2(s, |s|)$ where

$$h_2(s, i) = \begin{cases} 0 & \text{if } i = 0 \\ ord(s[i]) + h_2(s, i-1) \cdot (2^6 + 2^{16} - 1) & \text{otherwise} \end{cases}$$

$bp(s) = h_3(s, |s|)$ where

$$h_3(s, i) = \begin{cases} 0 & \text{if } i = 0 \\ ord(s[i]) \wedge (h_3(s, i-1) \cdot 2^7) & \text{otherwise} \end{cases}$$

- *ord* maps characters to integers
- \wedge stands for exclusive or