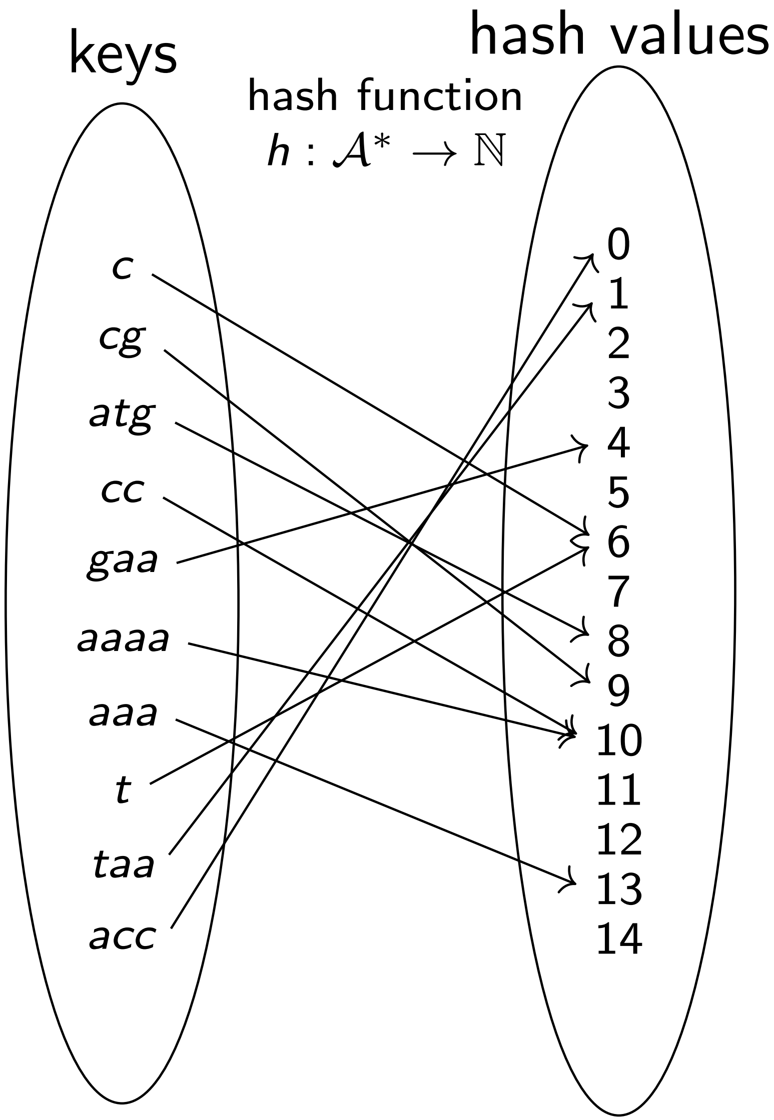
Excursion to hash functions

- one of the very basic tasks in computer science is to efficiently store values associated with keys
- simple solution:
  - store key/value pairs in list and use linear search to find value for given key in $O(n)$ time and space for n key value pairs
  - sort the keys and use binary search to find the value for a given key in $O(\log n)$ time and $O(n)$ space
- in many cases one wants to have constant time access for any key
- 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 non-negative 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, 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('acgt')` ⇒ 2 786 942 770 732 621 960
- can be applied to any hashable object (e.g. strings, numbers, functions)

Examples of hash functions for strings

$$\mathrm{js}(s) = h_1(s, |s|) \text{ where}$$

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

$$\mathrm{sdbm}(s) = h_2(s, |s|) \text{ where}$$

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

$$bp(s) = h_3(|s|) \text{ where}$$

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

ord maps characters to integers; ^ stands for exclusive or