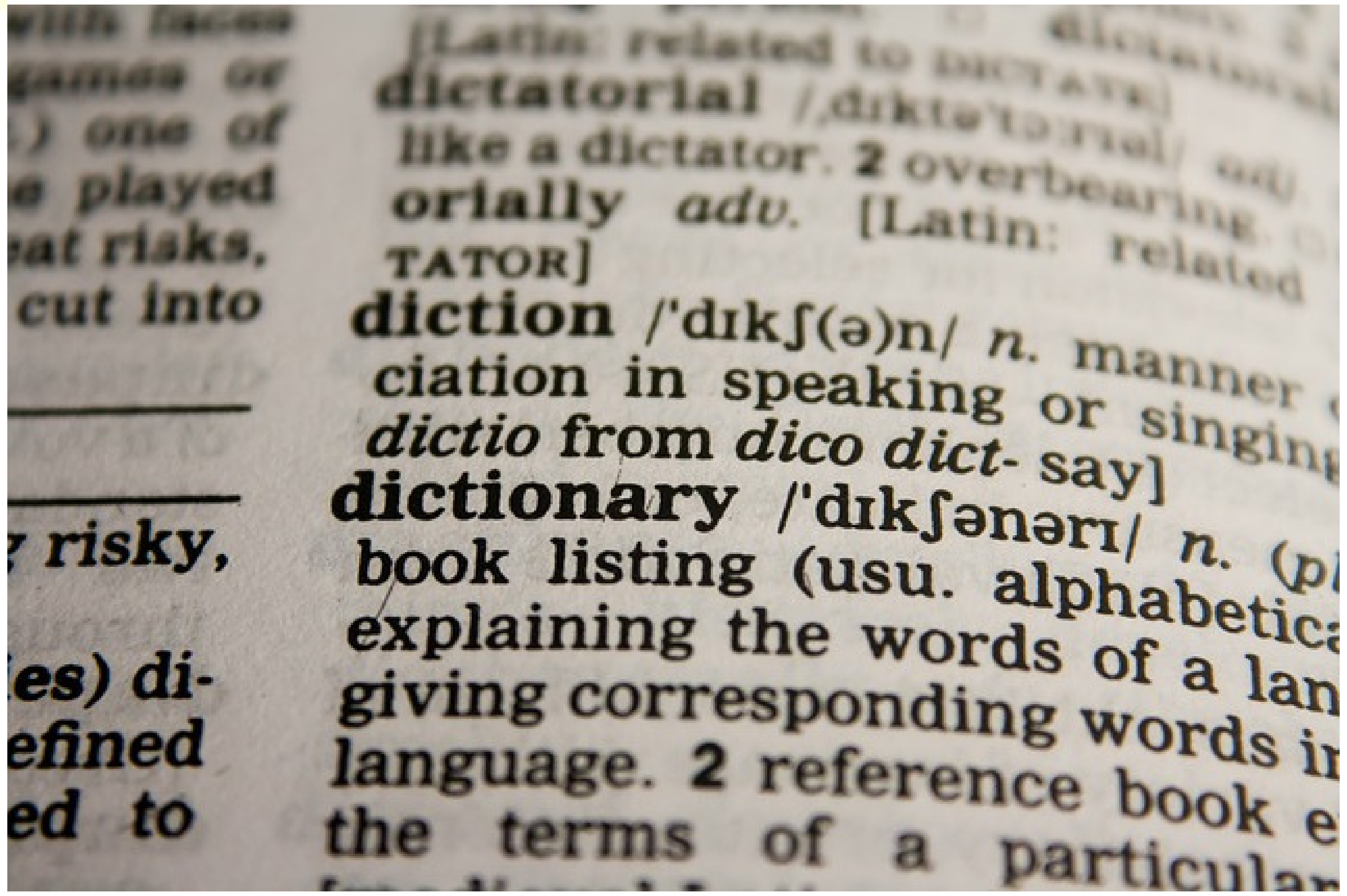
Hash Table

Tel
Fax
E-mail
Mobile
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

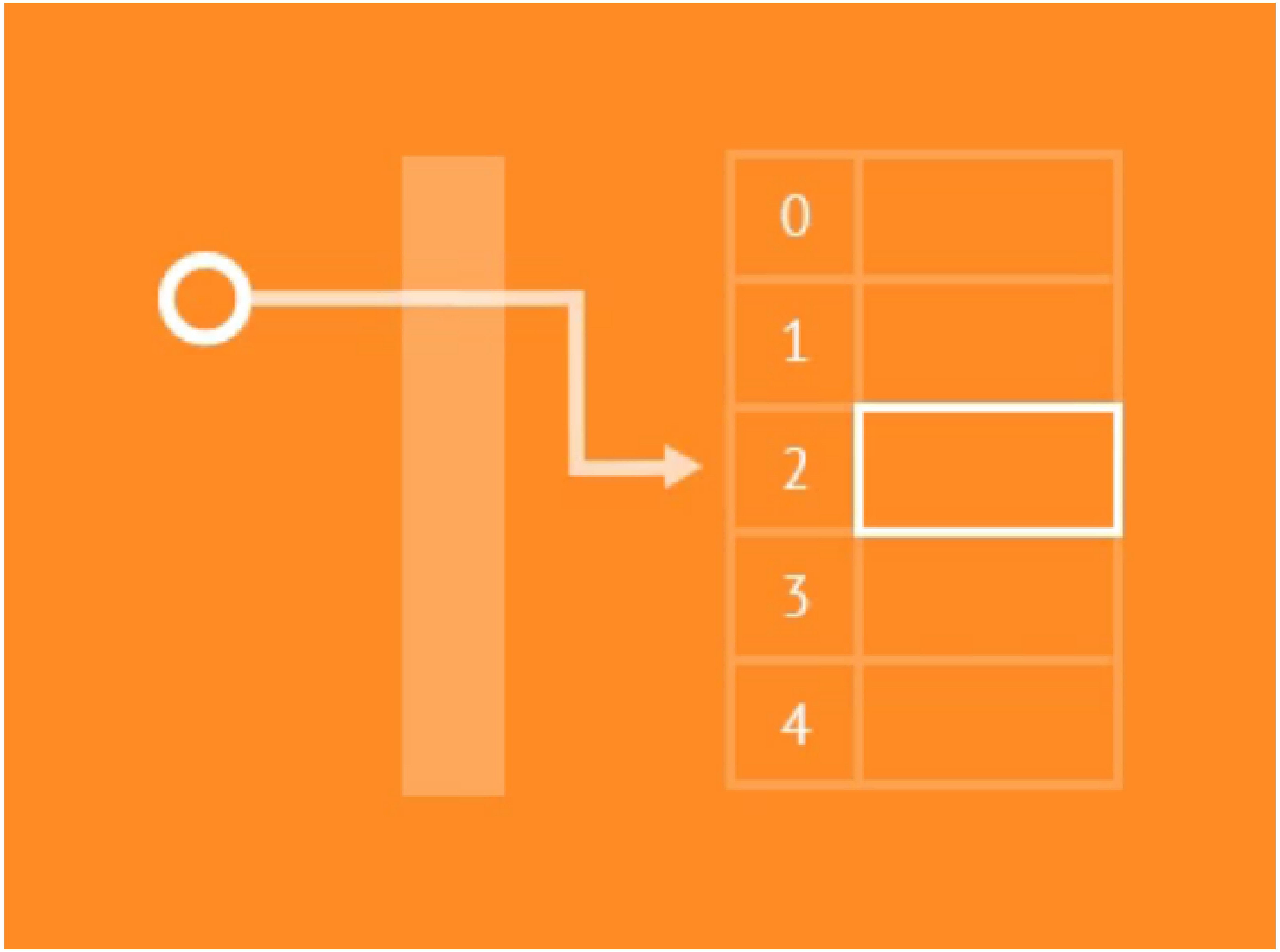


with faces
games or
) one of
e played
at risks,
cut into

risky,

ies) di-
efined
ed to

[Latin: related to DICTATE]
dictatorial /ˌdɪktəˈtɔːrɪəl/ *adj.*
like a dictator. **2** overbearing.
orially *adv.* [Latin: related
TATOR]
diction /ˈdɪkʃ(ə)n/ *n.* manner
ciation in speaking or singing
dictio from *dico dict-* say]
dictionary /ˈdɪkʃənərɪ/ *n.* (*pl*
book listing (usu. alphabetica
explaining the words of a lan
giving corresponding words in
language. **2** reference book e
the terms of a particular


0
1
2
3
4

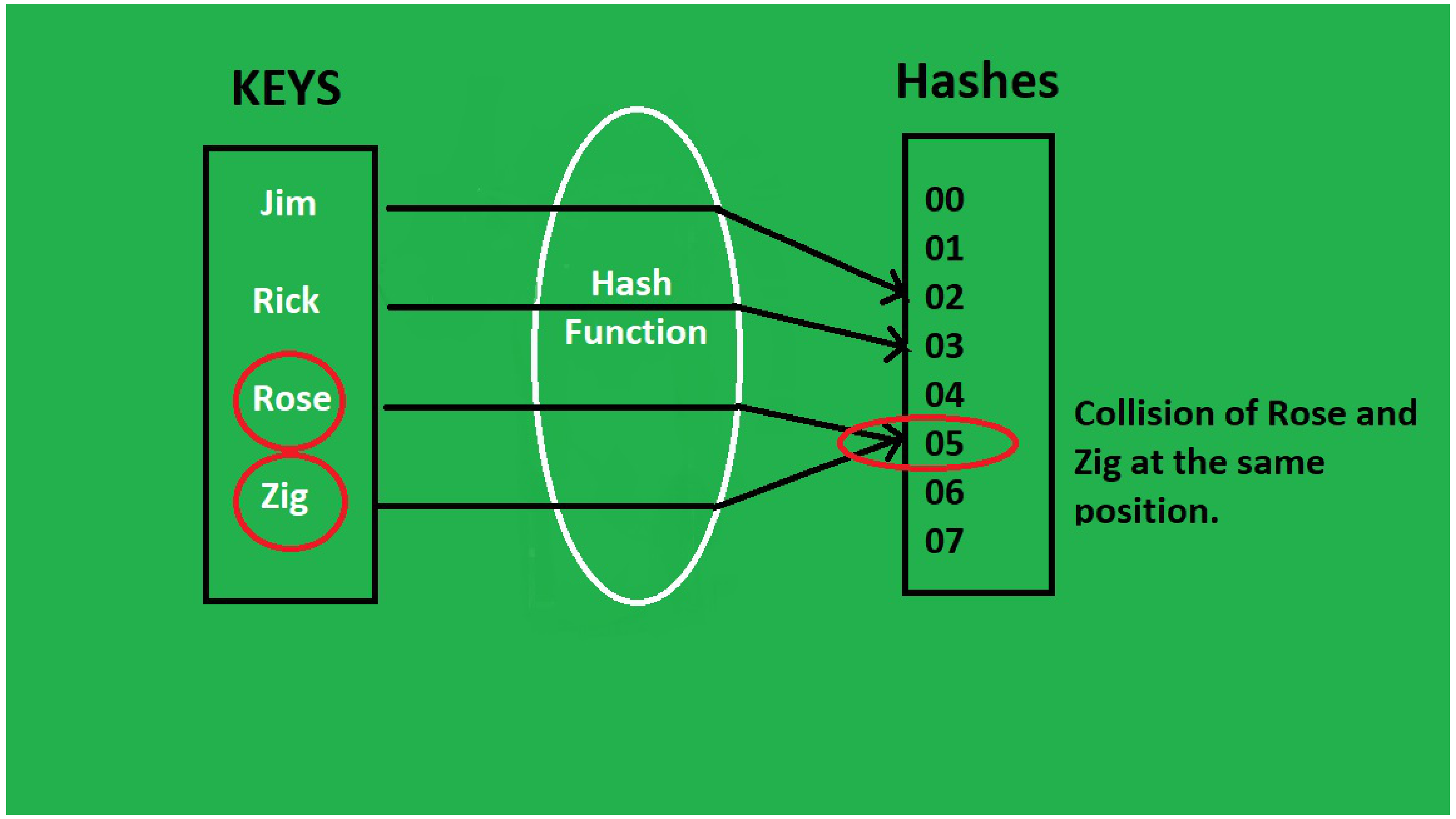
Hash function

A function that converts a given big input key to a small practical integer value. The mapped integer value is used as an index in the hash table. A good hash function should have the following properties.

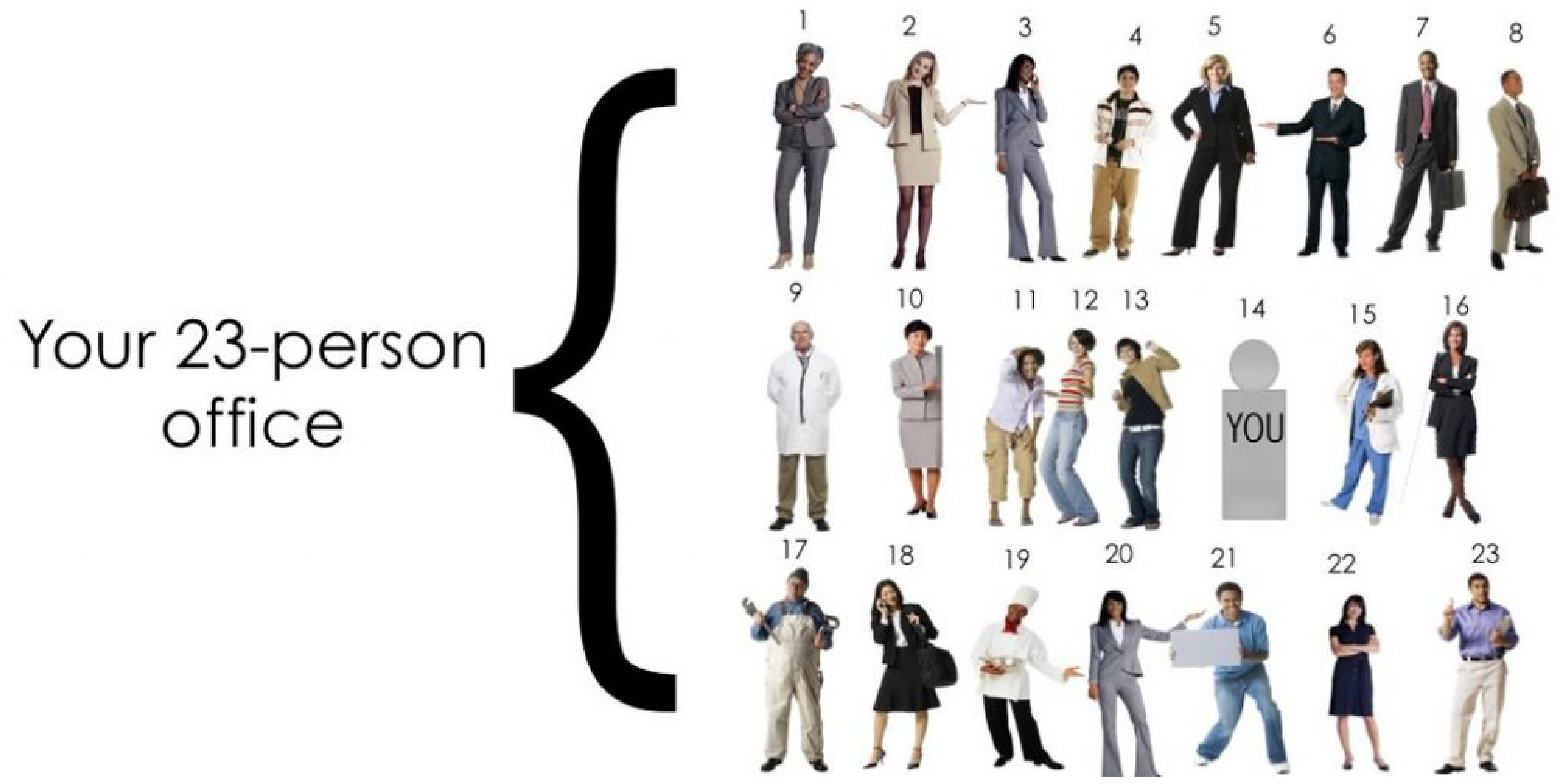
- **Efficiently** computable.
- Should **uniformly** distribute the keys (Each table position equally likely for each key)
    - For example, for phone numbers, a bad hash function is to take the first three digits. A better function is to consider the last three digits. Please note that this may not be the best hash function. There may be better ways.
- A hash procedure must be **deterministic**—meaning that for a given input value it must always generate the same hash value

Collision

There is a possibility that two keys result in the same value. The situation where a newly inserted key maps to an already occupied slot in the hash table is called **collision**.



Collisions are very likely even if we have big table to store keys.



An important observation is Birthday Paradox. With only 23 persons, the probability that two people have the same birthday is 50%.

Collision Handling

- Separate chaining
- Open spacing

For details:
https://www.cse.cuhk.edu.hk/irwin.king/_media/teaching/csc2100b/tu6.pdf

Hash table size

Choice of hash table size **depends** in part on the choice of hash function and collision resolution strategy

But a good general “rule of thumb” is:

- The hash table should be an array with a length about 1.3 times the maximum number of keys that will actually be in the table
- The size of the hash table array should be a prime number

So, the next prime larger than 1.3 times is a good choice

If you **underestimate** the number of keys

  - you may have to create a larger table and rehash the entries when it gets too full

if you **overestimate** the number of keys

  - you will be wasting some space

Use Cases

- Literally storing anything where order does not matter but access speed does.
- An inverted index for search engines and information retrieval where you map each token to a list of documents/websites that contain that token.
- Sets can be represented through hash tables. Finding if an item is present or not is a lot faster than having to look through a list of 1,000,000 items.
- An example of a really common application is counting frequencies of certain words where the key is that word and the value is the frequency.
- One way to represent a graph where each node maps to a list of its neighbors.