4.2.6 Hash Tables & Dictionaries

Hash Tables

A hash table is a data structure that stores data in a table. Each item is associated with a key, and the key is used to calculate the location where the data should be stored in the table.

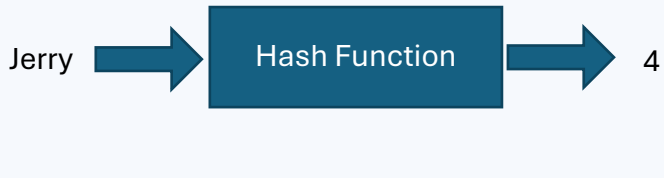
A hashing algorithm is applied to the key and returns a hash value. This hash value is then used as an index to locate where the data item should be stored in the table.

Example:

This hash table contains the names and mobile numbers of a person's contacts.

The data Jerry 07847564736 is being inserted into the hash table.

The key is the person's name "Jerry" and is input into the hash algorithm to produce a hash.



Index	Data
1	
2	Nigel 07943443221
3	
4	Jerry 07847564736
5	
…	

The result of applying the hash algorithm to the key "Jerry" was location 4 so Jerry's information was inserted in location 4 of the hash table.

Pros:

- Very efficient at adding new entries to the table.
- Very efficient at checking for the presence of data in the table.
- O(1) average time complexity

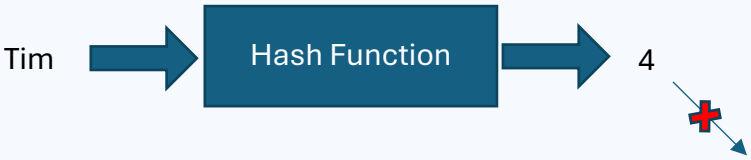
Collisions

A problem that can arise with hash tables is if two or more keys produce the same hash value after being passed through the hash function. This is called a collision.

Hash tables can become less efficient when collisions occur frequently. A poor hash function, or a table that is too full, can cause many keys to map to the same locations, increasing the time needed to find or insert data.

Example:

The key “Tim” has produced a hash for location 4 but Jerry’s data is already contained in this location.



Index	Data
1	
2	Nigel 07943443221
3	
4	Jerry 07847564736
5	
…	

Collision Handling

In the event of a collision, collision handling methods can be applied to decide what should be done with the data that is being entered into the table.

Chaining

If there is a collision, a list is created in that slot and any additional items are appended to the list.

Linear Probing

If there is a collision, check the next location until a free location is found.

Overflow Area

If there is a collision, put the data in a reserved area of memory called the overflow area.

Dictionary

A dictionary is an abstract data type that **stores data as key-value pairs**.

Each key is unique and is used to access its associated value.

Dictionaries are commonly implemented using hash tables, which means searching, inserting and deleting data can be very efficient when the hash function distributes keys well.

Example: A dictionary written in Python

```
student = {
     "name": "Alice",
     "age": 17,
     "grade": "A"
 }
```