#ASLI ENGINEERING

Internal structure of a Hash Table

SWIPE

BY

ARPIT BHAYANI

Internal Structure of Hash Tables

Hash Tables → One of the most widely used data structures

Every language has its own implementation

- ↳ Python - Dictionary
- ↳ Java - HashMap
- ↳ Javascript - Object
- ↳ Golang - Map

Hash Tables are also used as building block for constructing

- Classes and its members
- Variable lookup Tables

Hash Tables are designed to provide

constant time → insertions, deletion, lookups

key → value

apple	5
banana	15
cat	2
dog	3

Two ideas to construct HashTable

1. application key to hash key [0, N)

   apple → 127621 79

2. hash key to a smaller range [0, m)

   127621 79 → 17

1. Application keys to Hash Keys

We cannot put anything as a key in hash table

It is limited to a specific set of types

eg. string, int, tuple, etc

We can also use custom types as keys if they implement the 'hash' function that spits out an integer for the object

* For some native types, the hash function is internally implemented

Hash keys have a larger range say $[0, 2^{32})$

and a hash function converts the object to this int range

apple ⟶ f ⟶ 12762179

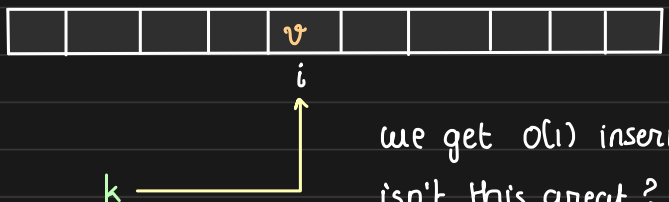
banana ⟶ f ⟶ 51962

cat ⟶ f ⟶ 72

dog ⟶ f ⟶ 1962719

Naive implementation

Because we are already getting int (i) from key (k),
what if we store the value (v) at index i ?



we get O(1) insert, updak, and lookup
isn't this great ?

This approach works well, only when N is small

Space required for holding array when

- N = 10 → 4 x 10 = 40B
- N = 100 → 4 x 100 = 400B
- N = 1000 → 4 x 1000 ≈ 4KB
- N = 1m → 4 x 1m ≈ 4MB

if N = int32 range → 4 x 4 billion ≈ 16 GB

Challenges:

1. Finding this big chunk of memory is tough
2. lot of slots would remain empty

2. Mapping hash key to smaller range

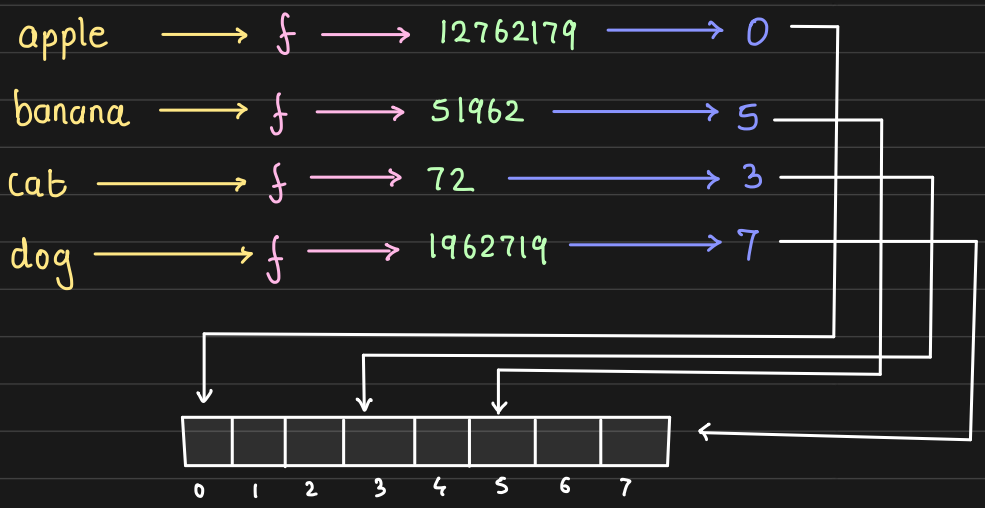
If we are planning to store k keys in the hash table, we have to have array of size m such that

$$m \in O(k)$$

This requires us to have a second step that reduces Hash key from range to smaller bin range

$$[0, N) \longrightarrow [0, m)$$

Say, we are planning to store 4 keys in hash table we can have a bin (array) of length 8



Adding more keys

If we add more keys to our hash table,
the holding array would have to be resized

$m \longrightarrow 2 \times m$

* No need to alter the first hash function → large N
as it hashed key to int range $[0, 2^{32})$

Why are we even hashing string to int ?

The first step is simplifying our problem statement
for the second step, making it easier to
optimize int → int distribution

The first step also allows us to give great abstraction

object → INT32

enabling us to support complex data types as keys