Hash Tables

Data Structures and Algorithms

Andrei Bulatov

The Dictionary Problem

We have a dictionary

Each entry of the dictionary consists of a key, a word, and an article explaining the word

We to perform several simple operations with dictionary:

- search
- insert
- delete

How should store the dictionary to perform those operations most efficiently?

Goal: perform search in O(1) time, at least on average

Straightforward Approaches

We can try lists:

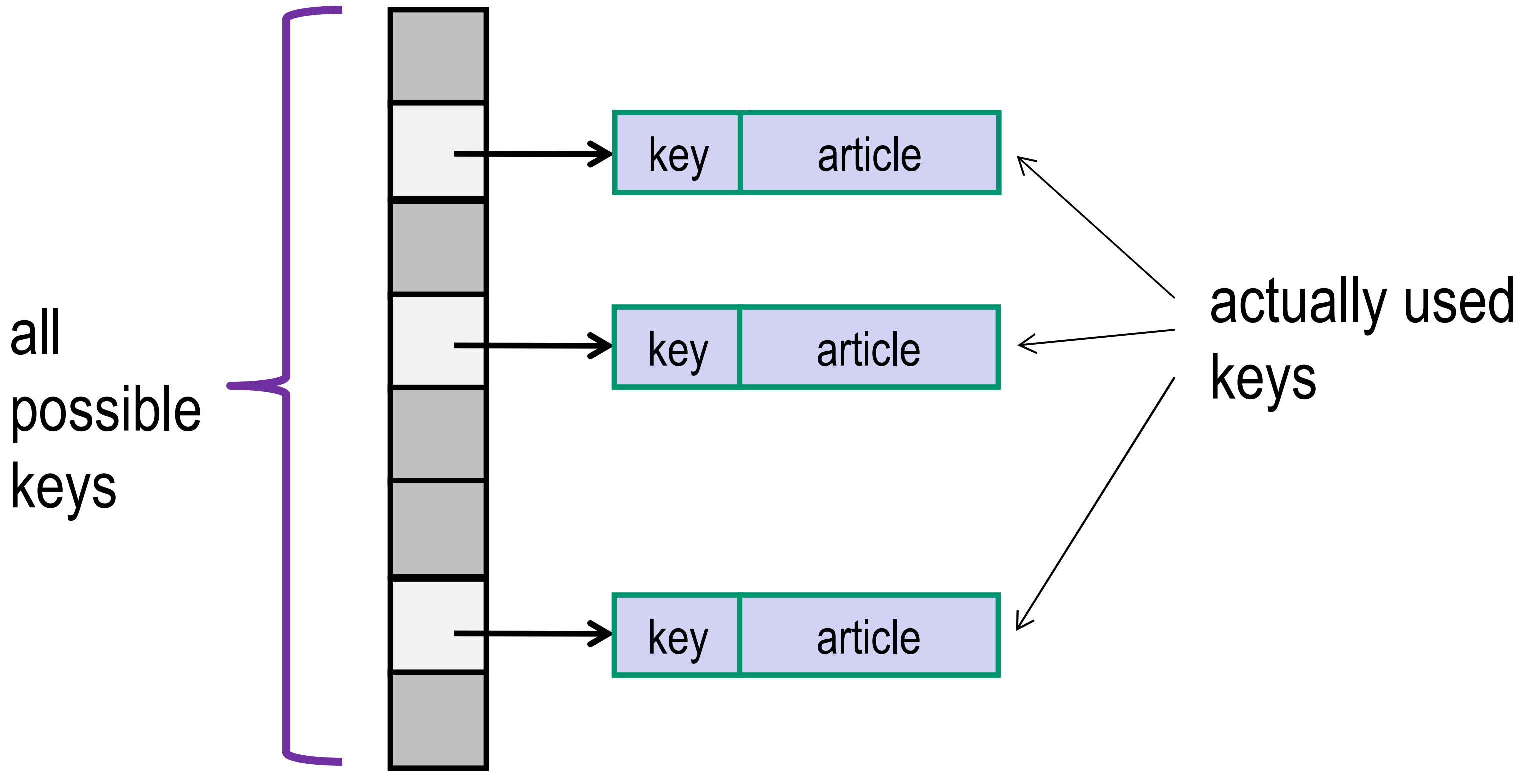
Then search takes O(n) time

Binary trees, heaps:

Search takes O(log n) time

Direct Access Tables

Create an array with entries indexed by all possible keys



Word *Antidisestablishmentarianism* contains 28 letters

Hashing

The idea is

- to have a smaller, more realistic table
- to address an entry use not the key k itself, but a certain function $h(k)$ computed using k

The function used is called hash function, and the table --- hash table

Let U be the set (universe) of all possible keys, and m is the desired size of the table

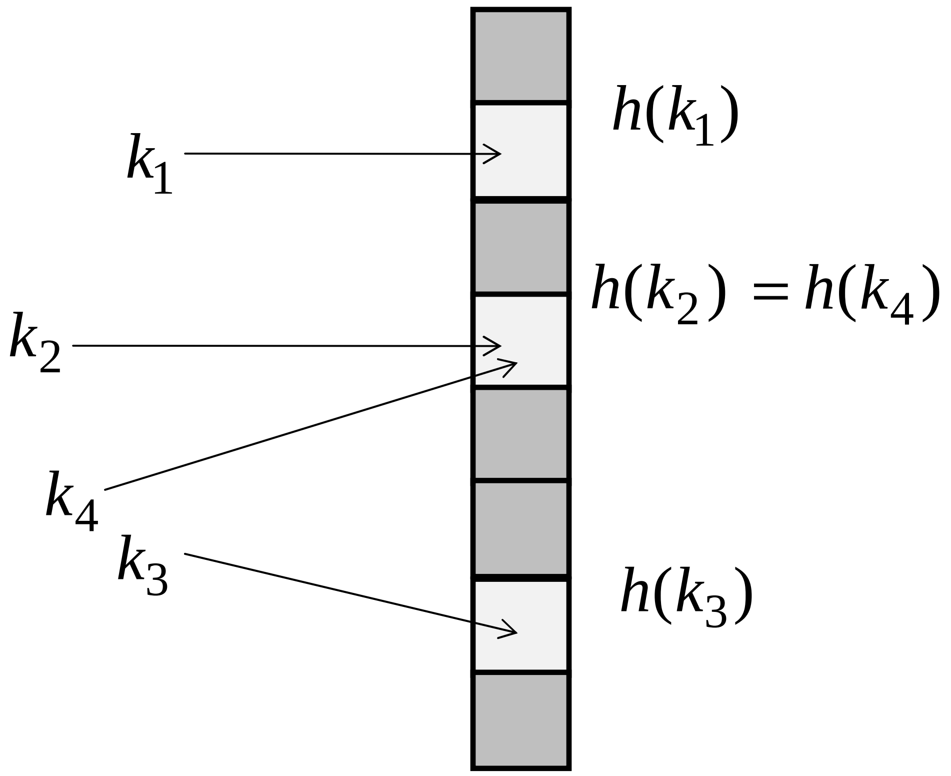
Then $h: U \to \{0, \dots, m - 1\}$

Key k (or the whole element) hashes to $h(k)$

$h(k)$ is the hash value of k

Collisions

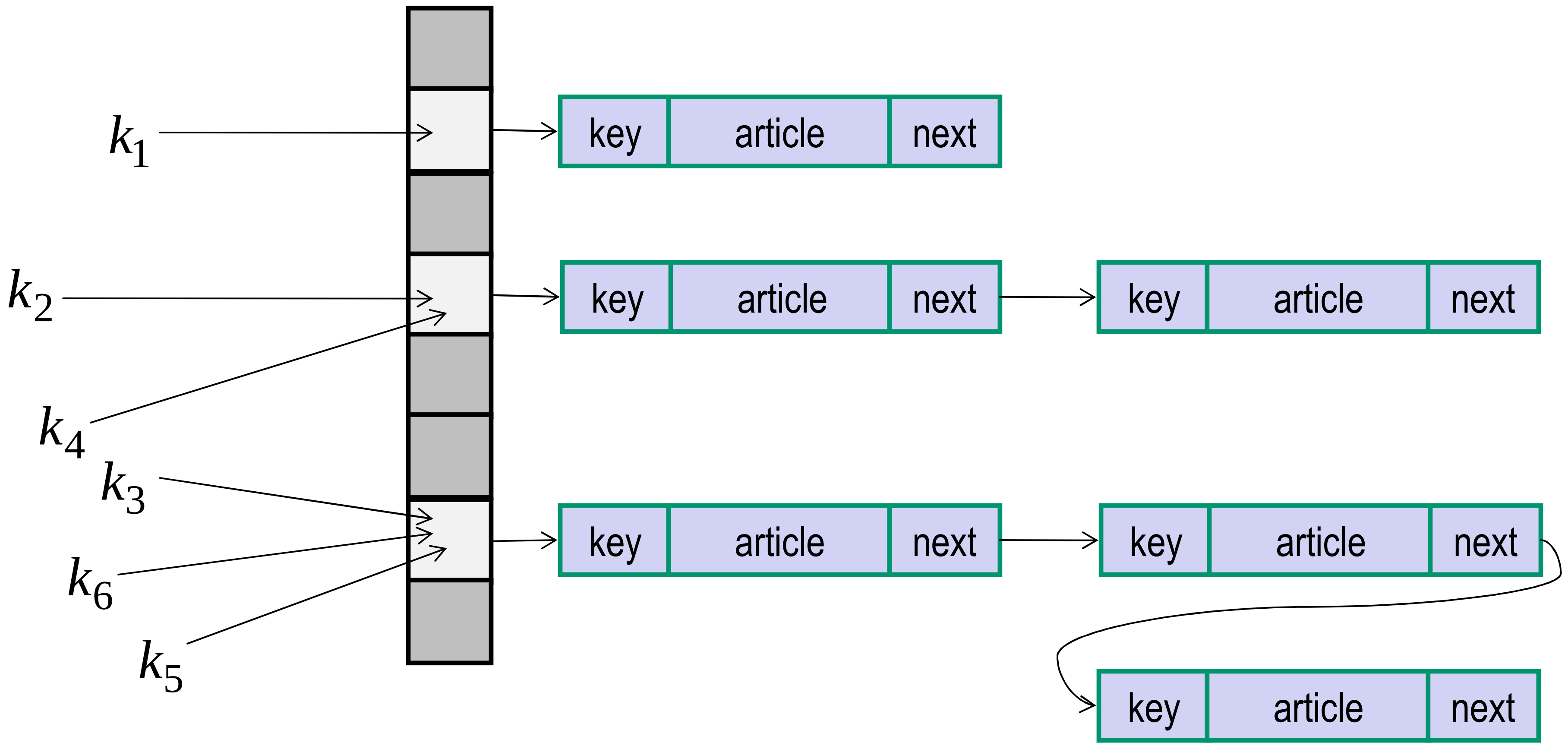
Everything is perfect as long as the hash values of all keys are different



This is called a collision

Chaining

In case of collision create a list of elements with the same hash value



Analysis

We estimate time needed for searching in a hash table

Due to collisions the worst case is very bad, O(n)

So we need some assumptions to do average case analysis

Simple uniform hashing: every key hashes to any of the m slots with equal probability

Also suppose we store n elements in m slots

The parameter $\alpha = \frac{n}{m}$ is called the load factor

For $j \in \{0,1, \ldots, m-1\}$ let n_j denote the number of keys with hash value j

Clearly, $n = n_0 + n_1 + \cdots + n_m$

and the average value of n_j is $E[n_j] = \alpha = \frac{n}{m}$

Analysis: Unsuccessful Search

We assume that computing h(k) takes O(1) time

Theorem

In a hash table in which collisions are resolved by chaining, an unsuccessful search takes $\Theta(1 + \alpha)$, on the average, under the assumption of simple uniform hashing

Proof

Any key k not stored in the table is equally likely to hash to any slot

The expected time to search is the expected length of the list in the slot h(k)

The expected length is $E[n_j] = \alpha$

Thus time needed $\Theta(1 + \alpha)$

Analysi: Successful Search

For successful search the probability that a list is searched is proportional to its length

Theorem

In a hash table in which collisions are resolved by chaining, a successful search takes $\Theta(1 + \alpha)$, on the average, under the assumption of simple uniform hashing

Proof

The element being searched is equally likely to be any of the n elements in the table

The number of elements examined when searching for an element x is the number of elements before x in the list $+ 1$

Analysis (cntd)

These elements were placed into the table after x

To find the expected number of elements examined we take the average over all elements in the table of the number of elements added to x's list after x

Let x_i be the i-th element added to the list and k_i its key

For keys k_i and k_j introduce the indicator random variable X_{ij} that equals 1 if $h(k_i) = h(k_j)$ and 0 otherwise

By assumption of simple uniform hashing, $\Pr[h(k_i) = h(k_j)] = 1/m$ and so $E[X_{ij}] = 1/m$

Analysis (cntd)

Then

$$E\left[\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}X_{ij}\right)\right]=\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}E[X_{ij}]\right)$$

$$=\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right)=1+\frac{1}{nm}\sum_{i=1}^{n}(n-i)$$

$$=1+\frac{1}{nm}\left(\sum_{i=1}^{n}n-\sum_{i=1}^{n}i\right)=1+\frac{1}{nm}\left(n^2-\frac{n(n+1)}{2}\right)$$

$$=1+\frac{n-1}{2m}=1+\frac{\alpha}{2}-\frac{\alpha}{2n}=O(1+\alpha)$$

Good Hash Functions

Good hash functions are those that are as close to simple uniform hashing as possible

It is difficult to achieve, since we do not know the distribution of keys

Note, there are two types of hash functions with absolutely different requirements:

- hash functions to support data structures
- cryptographic hash functions

Assumption:

All keys are natural numbers

The Division Method

Choose m

Then h(k) = k mod m

Should be careful with some values of m

Say, no powers of 2, or powers of 10, or …

Primes is a good choice, as long as they are not close to a power of 2

The Multiplication Method

Choose m

Choose A with 0 < A < 1

x mod 1 denotes the fractional part of x, that is $x - \lfloor x \rfloor$

Then $h(k) = \lfloor m\,(kA \bmod 1) \rfloor$

$m = 2^p$ is a convenient value

If the size of a computer word is w, choose A to be a fraction like $\frac{s}{2^w}$ for a integer s

To compute h(k), multiply k by 2^w

The result is a 2w-bit value $r_1 2^w + r_0$

Then h(k) is then the p most significant bits of r_0

Universal Hashing

To guarantee hashing even closer to simple uniform, a natural idea is to choose hash function also at random, independent of the keys being hashed

We use universal collection of hash functions

A collection H of hash functions is called universal, if for each pair of distinct keys k and l, the number of hash functions $h \in H$ such that $h(k) = h(l)$ is no more than $|H|/m$

To construct a hash table we first select $h \in H$ (randomly!), and then use it

Universal Hashing (cntd)

Lemma

Suppose a hash function is chosen at random from a universal collection and is used to hash n keys into a table of size m.

If key k is not in the table, then the expected length $E[n_{h(k)}]$ of the list that k hashes to is at most α = n/m.

If k is in the table, then the expected length $E[n_{h(k)}]$ of the list containing k is at most 1 + α

Corollary

Using universal hashing and collision resolution by chaining in a table with m slots, it takes expected time Θ(n) to handle any sequence of n table operations.

Constructing a Universal Hashing Collection

Choose a prime p such that all possible keys are in the range {0, …, p – 1}

Let $Z_p = \{0, \ldots, p-1\}$ and $Z_p^* = \{1, \ldots, p-1\}$

For $a \in Z_p^*$ and $b \in Z_p$ let

$$h_{a,b}(k) = ((ak+b) \bmod p) \bmod m$$

and $H_{p,m} = \{h_{a,b} : a \in Z_p^*, b \in Z_p\}$

Theorem

The class $H_{p,m}$ of hash functions is universal

Homework

Suggest how to organize a direct access table in which not all keys are different. All operations must run in $O(1)$ time

Show that if $|U| > mn$ (U denotes the set of all possible keys), there is a subset of U of size n consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is $\Theta(n)$