Algorithms Theory

05 - Hashing

Dr. Alexander Souza

Dictionary Problem

Randomization

Choose a hash function randomly from a family of functions

Overview

- Introduction
- Universal hashing "few collisions"
- Perfect hashing "no collisions"

The dictionary problem

Given: Universe U = [0…N-1] , where N is a natural number.

Goal: Maintain set $S \subseteq U$ under the following operations.

- Search(x,S): Is $x \in S$?
- Insert(x,S): Insert x into S if not already in S.
- Delete(x,S): Delete x from S.

Advantage of hashing:

$O(1)$ time per operation

Disadvantage of hashing:

- Does not directly support extended dictionary operations. Just search insert, delete.
- Maybe we have space overhead.

Trivial implementation

Array A[0…N-1] where $A[i] = 1 \Leftrightarrow i \in S$

Each operation takes time O(1) but the required memory space is $\Theta(N)$.

A	0	0	1	1	0	1	1	1	0	0	1
	0										N–1

Goal: Space requirement $O(|S|)$ and expected time O(1) per operation.

Idea of hashing

Use an array of length $O(|S|)$.

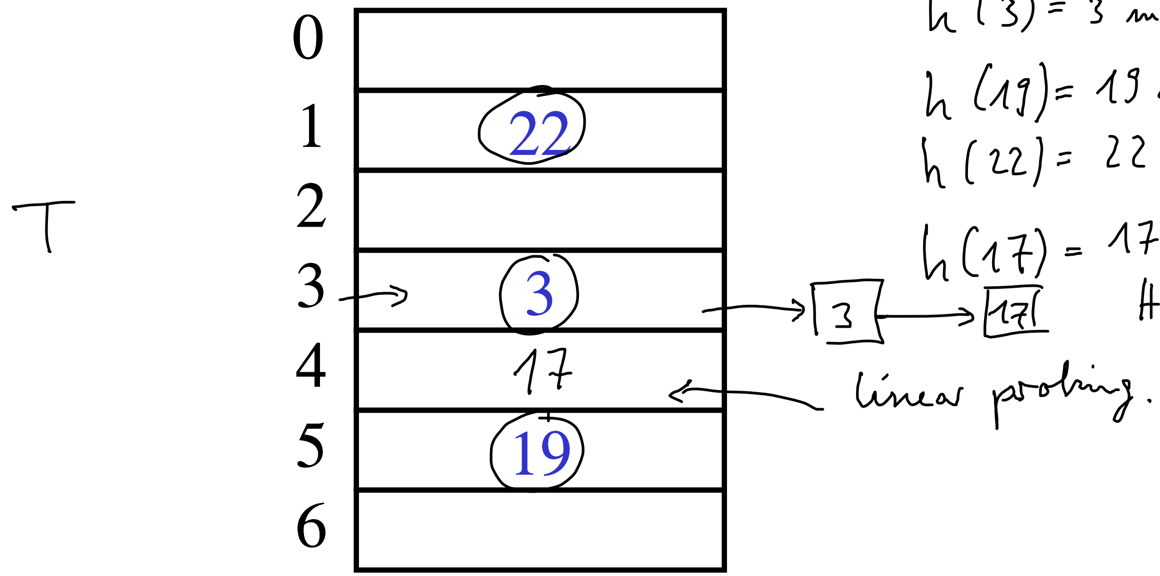
Compute the position where to store an element using a function defined on the keys.

Universe	$U = [0\ldots N\text{-}1]$	$m = O(\|S\|)$
Hash table	Array $T[0\ldots m\text{-}1]$	
Hash function	$h: U \rightarrow [0\ldots m\text{-}1]$	map keys to table positions

Element $x \in S$ is stored in $T[h(x)]$.

Example

$N = 100$; $U = [0\ldots99]$; $m = 7$; $h(x) = x \bmod 7$; $S = \{3, 19, 22\}$



$h(3) = 3 \bmod 7 = 3$

$h(19) = 19 \bmod 7 = 5$

$h(22) = 22 \bmod 7 = 1$

$h(17) = 17 \bmod 7 = 3$

Hashing with chaining

If 17 is inserted next, a collision arises because $h(17) = 3$.

Possible collision resolutions

- Hashing with chaining: $T[i]$ contains a list of elements.
- Hashing with open addressing: Instead of one address for an element there are m many that are probed sequentially.
- Universal hashing: Choose a hash function such that only few collisions occur. Collisions are resolved by chaining.
- Perfect hashing: Choose a hash function such that no collisions occur.

  Set S is known in advance

Universal hashing

Idea: Use a class H of hash functions. The hash function $h \in H$ actually used is chosen uniformly at random from H.

Goal: For each $S \subseteq U$, the expected time of each operation is $O(1 + \beta)$, where $\beta = |S|/m$ is the load factor of the table.

Running time: O(longest chain)

this could be $O(|S|)$

Expected running time: $O\left(\frac{|S|}{m}\right)$

Property of *H*: For two arbitrary elements $x,y \in U$, only "few" $h \in H$ lead to a collision ($h(x) = h(y)$).

collision

Universal hashing

$\mathcal{U} = [0, \ldots, N-1]$ $T[0, \ldots, m-1]$

Definition: Let N and m be natural numbers. A class $H \subseteq \{ h : [0 \ldots N\text{-}1] \rightarrow [0 \ldots m\text{-}1] \}$ is universal if for all $x, y \in U = [0 \ldots N\text{-}1]$, $x \neq y$:

x y

set of functions that lead to a collision for x and y

$$\frac{|\{h \in H : h(x) = h(y)\}|}{|H|} \leq \frac{1}{m}$$

"small" prob. of collision.

all functions in the class H.

Intuitively: An h chosen uniformly at random is as good as if the table positions of the elements are chosen uniformly at random.

A universal class of functions

Let N, m be natural numbers, where N is prime.
For numbers $a \in \{1, \ldots, N\text{-}1\}$ and $b \in \{0, \ldots, N\text{-}1\}$, let
$h_{a,b} : U = [0 \ldots N\text{-}1] \rightarrow \{0, \ldots, m\text{-}1\}$ be defined as:

$$h_{a,b}(x) = ((ax + b) \bmod N) \bmod m$$

Theorem: $H = \{h_{a,b}(x) \mid 1 \leq a < N \text{ and } 0 \leq b < N\}$ is a universal class of hash functions.