Dictionary ADT

Stores a collection of records of the form

key, data

Provides the following operations:

get(key), put(key, data), remove(key)

Dictionary ADT

Stores a collection of records of the form

key, data

Provides the following operations:

get(key), put(key,data), remove(key)

A dictionary can be implemented with different data structures:

get(key)
put(key,data)
remove(key)

Dictionary ADT

Stores a collection of records of the form

key, data

Provides the following operations:

get(key), put(key,data), remove(key)

A dictionary can be implemented with different data structures:

	Linked list
get(key)	O(n)
put(key,data)	O(n)
remove(key)	O(n)

Dictionary ADT

Stores a collection of records of the form

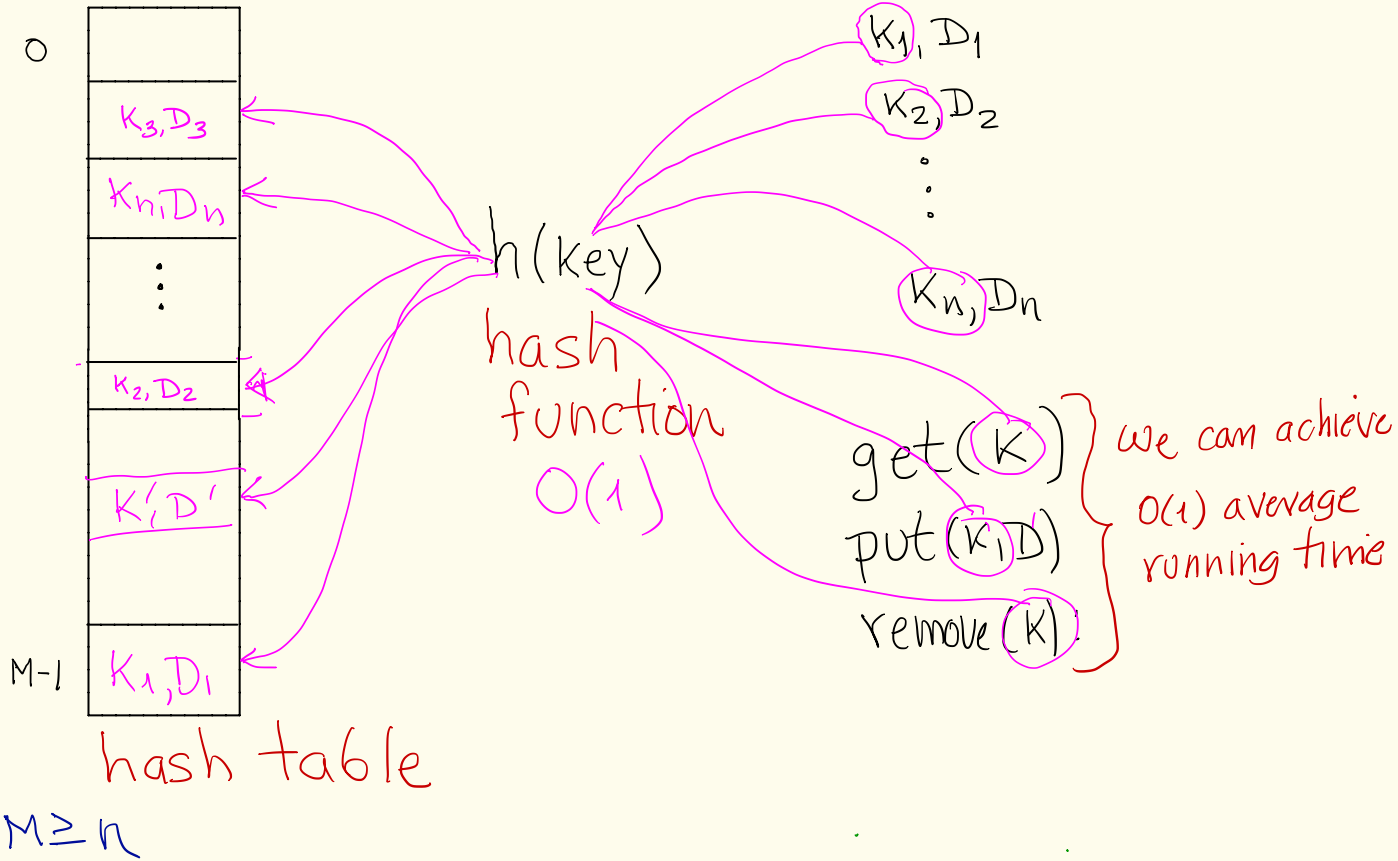
key, data

Provides the following operations:

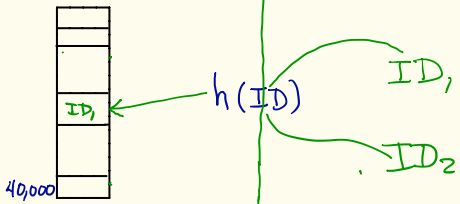
get(key), put(key,data), remove(key)

A dictionary can be implemented with different data structures:

	Linked list	Sorted array
get(key)	O(n)	O(log n)
put(key,data)	O(n)	O(n)
remove(key)	O(n)	O(n)


0
K_3, D_3
K_n, D_n
K_2, D_2
K', D'
M-1
K_1, D_1
hash table
h(key)
hash function
O(1)
K_1, D_1
K_2, D_2
K_n, D_n
get(K)
put(K, D)
remove(K)
we can achieve
O(1) average
running time
$M \geq n$

collision



h must give a position for EVERY ID

Set of all possible UWO ID's:

$d_8 d_7 d_6 \cdots d_1 d_0$

Since each d_i can take 10 different values, this set has 10^9 elements, which is much bigger than the size of the table, so the hash function will produce collisions

Design issues

- Determine size of the table
- Select hash function

Design issues

- Determine size of the table
- Select hash function
  - low time complexity
  - causes few collisions

Design issues

- Determine size of the table
- Select hash function
  - low time complexity
  - causes few collisions
    must spread keys uniformly
    across entire table

Design issues

- Determine size of the table
- Select hash function
  - low time complexity
  - causes few collisions
    must spread keys uniformly across entire table
- How to deal with collisions

Hash function

Keys can be of any type: integer, character, string, array, ...

Hash code

$g(\text{key}) \longrightarrow$ integer

we want a large integer to ensure all entries of the hash table are used

Compression map

$f(\text{integer}) \rightarrow$ position of hash table

Hash function

$$h(\text{key}) = f(g(\text{key}))$$

Hash Code

Converting a string into an integer: First a character can be converted into an integer through casting:

(int) char → integer

Then, apply some function to the integers corresponding to the characters of the string to produce a single integer.

For example:

$$g(\text{“}c_{k-1}\, c_{k-2} \cdots c_1\, c_0\text{”}) = \sum_{i=0}^{k-1} (\text{int}) c_i$$

This is not a good hash code because it produces small integer values.

Polynomial Hash Code

$$s = c_{k-1}\, c_{k-2} \cdots c_1\, c_0$$
$$(int)c_{k-1}\ (int)c_{k-2} \cdots (int)c_1\ (int)c_0$$

Polynomial Hash Code

$$S = \text{"}c_{k-1}\ c_{k-2} \cdots\ c_1\ c_0\text{"}$$
$$(\text{int})c_{k-1}\ (\text{int})c_{k-2} \cdots (\text{int})c_1\ (\text{int})c_0$$

Use these numbers as the coefficients of a polynomial:

$$g(S) = (\text{int})c_{k-1}x^{k-1} + (\text{int})c_{k-2}x^{k-2} + \cdots + (\text{int})c_1x^1 + (\text{int})c_0$$

$x = 33, 37, 39, 41$ good choices for x

Hash Function

$$g(\text{“}c_{k-1}c_{k-2}\cdots c_1 c_0\text{”}) = \left((\text{int})c_{k-1}x^{k-1} + (\text{int})c_{k-2}x^{k-2} + \cdots + (\text{int})c_1 x^1 + (\text{int})c_0\right)$$

Hash Function

$$h(\text{“}c_{k-1}c_{k-2}\cdots c_1 c_0\text{”}) = \left((\text{int})c_{k-1}x^{k-1} + (\text{int})c_{k-2}x^{k-2} + \cdots + (\text{int})c_1 x^1 + (\text{int})c_0\right) \bmod M$$

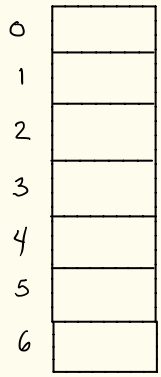
$$= \left(\left(\cdots\left(\left((\text{int}\,c_{k-1})x + (\text{int})c_{k-2}\right)x + \cdots + (\text{int})c_1\right)x + (\text{int})c_0\right) \bmod M$$

Hash Function

$$h(\text{“}c_{k-1}c_{k-2}\cdots c_1 c_0\text{”}) = \left((\text{int})c_{k-1}x^{k-1} + (\text{int})c_{k-2}x^{k-2} + \cdots + (\text{int})c_1 x^1 + (\text{int})c_0\right) \bmod M$$

$$= \left(\left(\cdots\left(\left((\text{int}\,c_{k-1})x + (\text{int})c_{k-2}\right)x + \cdots + (\text{int})c_1\right)x + (\text{int})c_0\right) \bmod M$$

Algorithm hashFunction(s)
In: string $s = \text{“}c_{k-1}c_{k-2}\cdots c_1 c_0\text{”}$
Out: position for s in hash table



T

M = 7

$14, d_1$

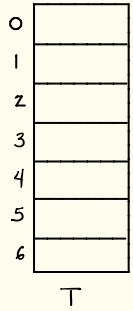
$12, d_2$

$13, d_3$

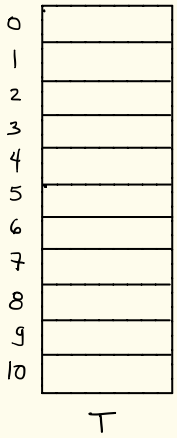
$21, d_4$

$h(k) = k \bmod 7$

$h(k) = k \bmod 7$

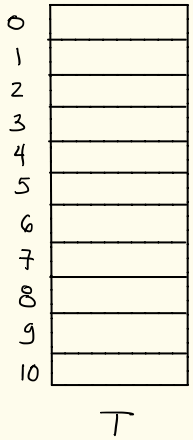


$14, d_1$
$12, d_2$
$13, d_3$
$21, d_4$
$19, d_5$
$2, d_6$



$h(k) = k \bmod 11$

3
14
25
5
28
91



$h(k) = k \bmod 11$