CS 240 – Data Structures and Data Management

Module 7E: Hashing - Enriched

T. Biedl E. Kondratovsky M. Petrick O. Veksler

Based on lecture notes by many previous cs240 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2022

version 2022-02-15 17:45

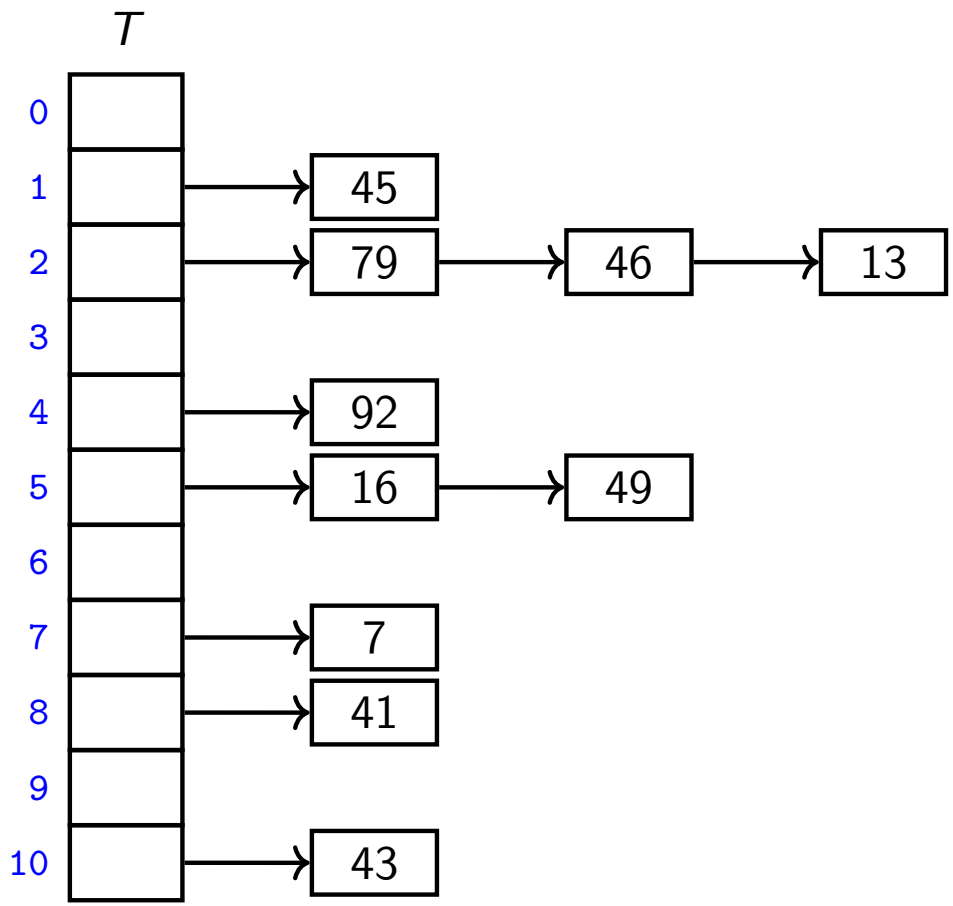
Outline

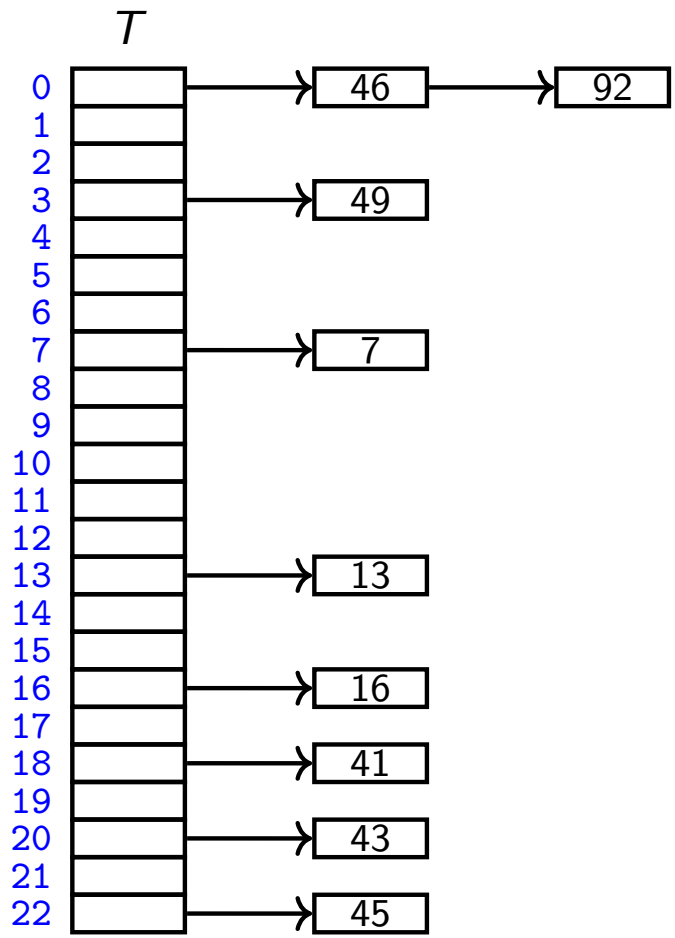
- Rehashing
- Multiplication method
- Randomly chosen hash-functions

Outline

- Rehashing
- Multiplication method
- Randomly chosen hash-functions

Rehashing


T
0
1
2
3
4
5
6
7
8
9
10
45
79
46
13
92
16
49
7
41
43


T
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
46
92
49
7
13
16
41
43
45

Outline

- Rehashing
- Multiplication method
- Randomly chosen hash-functions

Multiplication method

- Pick $A \in (0,1)$ (preferably an irrational, e.g. $A = \frac{\sqrt{5}-1}{2} = \phi$)
- $h(k) = \Big\lfloor M \cdot \big(\underbrace{\underbrace{A \cdot k}_{\text{multiply}} - \underbrace{\lfloor A \cdot k \rfloor}_{\text{integral part}}}_{\text{fractional part, in } [0,1)} \big) \Big\rfloor$ (integer in $[0, M)$)
- Example:

$A = 0.a_1 a_2 a_3 \ldots$
$k = b_1 b_2 \ldots b_6$
(both in base 2)

Should use at least $\log |U| + \log |M|$ bits of A.

$A \cdot k$		(leading bits)						(bits of fractional part)					
	$=$	0	0	0	..	0	0						
	$+a_1 \cdot$	0	b_1	b_2	b_3	..	b_5	b_6					
	$+a_2 \cdot$	0	0	b_1	b_2	b_3	..	b_5	b_6				
	$+a_3 \cdot$	0	0	0	b_1	b_2	b_3	..	b_5	b_6			
	$\vdots$				$\ddots$	$\ddots$		$\ddots$	$\ddots$	$\ddots$	$\ddots$		
	$+a_5 \cdot$	0	0	0	0	0	b_1	b_2	b_3	..	b_5	b_6	
	$+a_6 \cdot$	0	0	0	0	0	0	b_1	b_2	b_3	..	b_5	b_6
	$+a_7 \cdot$	0	0	0	0	0	0	0	b_1	b_2	b_3	..	b_5 b_6
	$+a_8 \cdot$	0	0	0	0	0	0	0	0	b_1	b_2	b_3	.. b_5 b_6
	$\vdots$							$\leftarrow h(k) \rightarrow$			$\ddots$	$\ddots$	$\ddots$ $\ddots$ $\ddots$

Outline

- Rehashing
- Multiplication method
- Randomly chosen hash-functions

Randomly chosen hash-functions

- There are $|U|^M$ many possible hash-functions
- Ideally we would choose randomly among all of them.
- But then we cannot compute hash-value quickly!

- **Idea:** Fix a family $\mathcal{H}$ of hash-functions that are easy to compute. Then choose uniformly among them.
- Example:
  - $U = \mathbb{Z}_5$, $M = 2$
  - $h_b(k) = ((k + b) \bmod 5) \bmod 2$
  - $\mathcal{H} = \{h_b : b \in \mathbb{Z}_5\}$
  - Choose $b \in \mathbb{Z}_5$ randomly to get hash-function

$\mathcal{H}$	keys				
	0	1	2	3	4
h_0	0	1	0	1	0
h_1	1	0	1	0	0
h_2	0	1	0	0	1
h_3	1	0	0	1	0
h_4	0	0	1	0	1

- But how do we measure whether these are "good"?

Universal hash-functions

- For analysis, we needed *uniform hash-values*:,

$$P(h(k) = i) = \frac{1}{M}$$

- But this is *not* good enough.

$\mathcal{H}$	keys				
	0	1	2	4	5
h_0	0	0	0	0	0
h_1	1	1	1	1	1

- $P(h(k) = i) = \frac{1}{2}$ for $i = 0, 1$ and any k
- But these hash-functions are terrible!
- Problem: hash-values not independent

- Also want: Small probability of collisions (**universal hashing**):

$$P\Big(h(k) = h(k')\Big) = \frac{1}{M} \quad \text{for any two keys } k \neq k'$$

- This is enough for analyzing hashing with chaining as before.

Carter-Wegman hash-function

$$h_{a,b}(k) = \Big(\underbrace{a \cdot k + b \bmod p}_{f_{a,b}(k)} \Big) \bmod M$$

(where $k \in \mathbb{Z}_p$, p prime, $a, b \in \mathbb{Z}_p$ chosen randomly, $a \neq 0$), $M < p$

Example: $(p = 5, M = 2)$:

	keys				
	0	1	2	3	4
$f_{1,0}$	0	1	2	3	4
$f_{2,0}$	0	2	4	1	3
$f_{1,2}$	2	3	4	0	1
$f_{2,1}$	1	3	0	2	4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

	keys				
	0	1	2	3	4
$h_{1,0}$	0	1	0	1	0
$h_{2,0}$	0	0	0	1	1
$h_{1,2}$	0	1	0	0	1
$h_{2,1}$	1	1	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Claim: $f_{a,b}$ is a permutation of $\mathbb{Z}_p$.