ETH zürich

Exercise Session 05 – Hashing

Data Structures and Algorithms

These slides are based on those of the lecture, but were adapted and extended by the teaching assistant Adel Gavranović

Today's Schedule

Intro
Follow-up
Feedback for **code** expert
Learning Objectives
Repetition: Throwing Eggs
Selection
Hashing
Code-Example: Hashtables, Hash-functions and Collisions
Old Exam Question
Tips for **code** expert
Outro

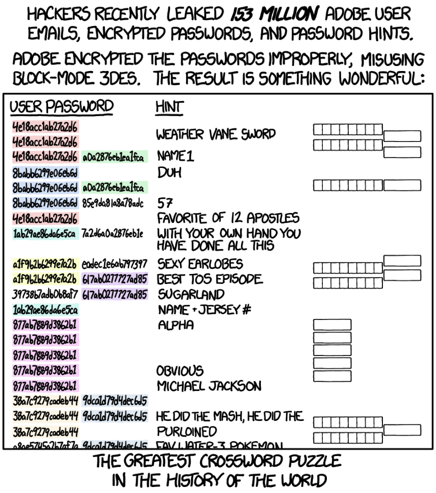
n.ethz.ch/~agavranovic

▸ Exercise Session Material

▸ Adel's Webpage

▸ Mail to Adel

Comic of the Week



xkcd

1. Intro

Intro

Intro

- My voice is a little strained today – Sorry

2. Follow-up

Follow-up from last exercise session

Follow-up from last exercise session

- Regarding last week's in-class coding exercise

Follow-up from last exercise session

- Regarding last week’s in-class coding exercise
  - No worries if you were not able to solve the example exercise during the session
  - It was a rather hard task to get into (no matter how “easy” it was to solve)
- In general: the master solutions will now be published sooner

3. Feedback for **code** expert

General things regarding code expert

General things regarding **code** expert

- If you submit via PDF-upload
    - Make sure to mention it in the submission
    - Make sure its high resolution or a PDF

Task "Prefix Sum in 2D"

Task "Prefix Sum in 2D"

- Don’t use `[]`-accessing but instead use `.at()`
  - It’s safer (because it checks for out-of-bounds access)
  - It might give better error messages as to where the error occurred

Task "Sliding Window"

Task "Sliding Window"

- Most of you only implemented one (out of three) correctly or at all
  - Which is good enough to obtain the XP
  - The phrasing was a little ambiguous

Questions regarding **code** expert from your side?

4. Learning Objectives

Learning Objectives

- ☐ Understand *Hashing*, its components, and related concepts:
  - ☐ Prehashing
  - ☐ Collision
  - ☐ Simple Uniform Hashing
  - ☐ Uniform Hashing
  - ☐ Open Addressing
  - ☐ Closed Hashing
  - ☐ Chaining
- ☐ Be able to apply simple *hashing methods* by hand

5. Repetition: Throwing Eggs

Throwing eggs

- What would be your strategy if you would have an arbitrary number of eggs and n floors?

Throwing eggs

- What would be your strategy if you would have an arbitrary number of eggs and n floors?
  - Binary search. Worst case: $\log_2 n$ tries.

Throwing eggs

- What would be your strategy if you would have an arbitrary number of eggs and n floors?
  - Binary search. Worst case: $\log_2 n$ tries.
- What would you do if you only had one egg?

Throwing eggs

- What would be your strategy if you would have an arbitrary number of eggs and n floors?
  - Binary search. Worst case: $\log_2 n$ tries.
- What would you do if you only had one egg?
  - Start from the bottom. n tries.

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length:

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals:

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials:

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials: $f'(k) = 1 - n/k^2 = 0 \Rightarrow k = \sqrt{n}$.
  $n = 100 \Rightarrow 19$ Trials. $\Theta(\sqrt{n})$

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials: $f'(k) = 1 - n/k^2 = 0 \Rightarrow k = \sqrt{n}$.
  $n = 100 \Rightarrow 19$ Trials. $\Theta(\sqrt{n})$
- Second approach: take first throw trial into account by considering decreasing interval sizes.

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials: $f'(k) = 1 - n/k^2 = 0 \Rightarrow k = \sqrt{n}$.
  $n = 100 \Rightarrow 19$ Trials. $\Theta(\sqrt{n})$
- Second approach: take first throw trial into account by considering decreasing interval sizes. Choose smallest s such that $s + s - 1 + s - 2 + ... + 1 = s(s+1)/2 \geq n$.

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials: $f'(k) = 1 - n/k^2 = 0 \Rightarrow k = \sqrt{n}$.
  $n = 100 \Rightarrow 19$ Trials. $\Theta(\sqrt{n})$
- Second approach: take first throw trial into account by considering decreasing interval sizes. Choose smallest s such that
  $s + s - 1 + s - 2 + ... + 1 = s(s+1)/2 \geq n$. If $n = 100$ then $s = 14$.
  Maximum number of trials: $s \in \Theta(\sqrt{n})$

Throwing eggs

Strategy using two eggs

- First approach: intervals of equal length: partition n into k intervals: maximum number of trials $f(k) = k + n/k - 1$
  Minimize maximum number of trials: $f'(k) = 1 - n/k^2 = 0 \Rightarrow k = \sqrt{n}$.
  $n = 100 \Rightarrow 19$ Trials. $\Theta(\sqrt{n})$
- Second approach: take first throw trial into account by considering decreasing interval sizes. Choose smallest s such that
  $s + s - 1 + s - 2 + ... + 1 = s(s+1)/2 \geq n$. If $n = 100$ then $s = 14$.
  Maximum number of trials: $s \in \Theta(\sqrt{n})$

Asymptotically both approaches are equally good.

6. Selection

Selection algorithm

Selection algorithm

- What happens if many elements are equal when partitioning?

5	5	5	5	5	5	5	5	5	5

Selection algorithm

- What happens if many elements are equal when partitioning?

5	5	5	5	5	5	5	5	5	5

- smaller partition is empty, larger $n - 1$ times 5

left

right

5	5	5	5	5	5	5	5	5	5

Selection algorithm

- What happens if many elements are equal when partitioning?

5	5	5	5	5	5	5	5	5	5

- smaller partition is empty, larger $n-1$ times 5

left

right	5	5	5	5	5	5	5	5	5	5

- degrade runtime to n^2

Selection algorithm

- What happens if many elements are equal when partitioning?

5	5	5	5	5	5	5	5	5	5

- smaller partition is empty, larger $n-1$ times 5

left

right

5	5	5	5	5	5	5	5	5	5

- degrade runtime to n^2
- Solution?

Selection algorithm

- On equality with pivot, alternate between partitions

Selection algorithm

- On equality with pivot, alternate between partitions
- Modify algorithm to return number of elements equal to pivot

Demonstration with pivot 5

Input 5 4 5 5 6 5

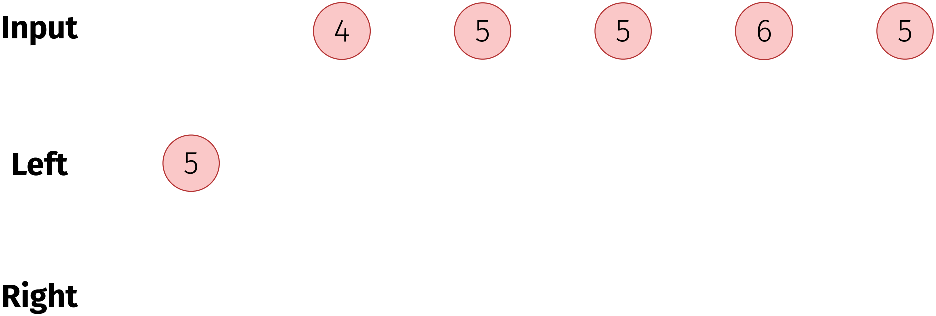
≤ 5

Left 5 4 5 5 5

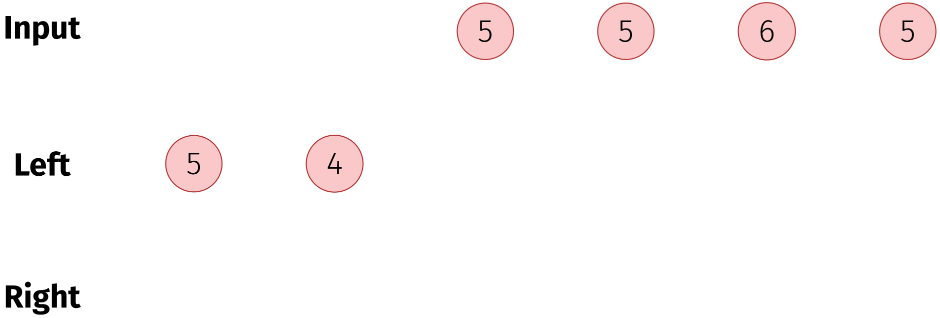
5 <

Right 6 5 5

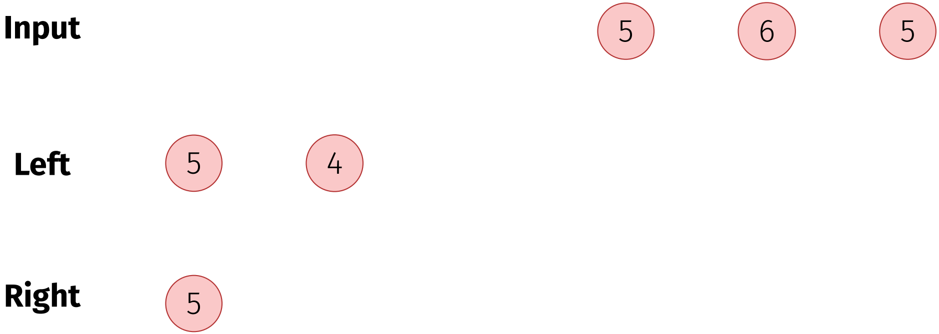
Demonstration with pivot 5



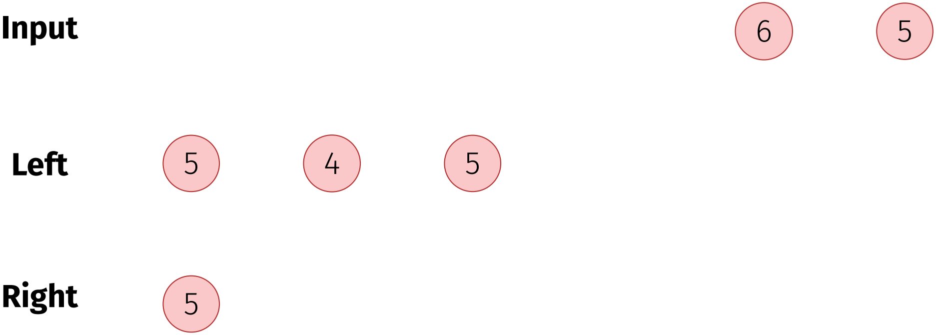
Demonstration with pivot 5



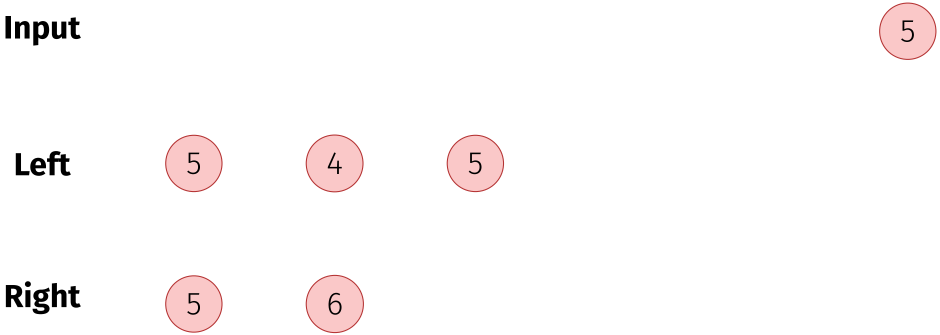
Demonstration with pivot 5



Demonstration with pivot 5



Demonstration with pivot 5



Demonstration with pivot 5

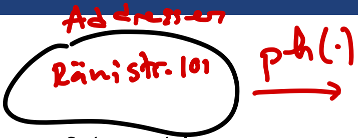
Input

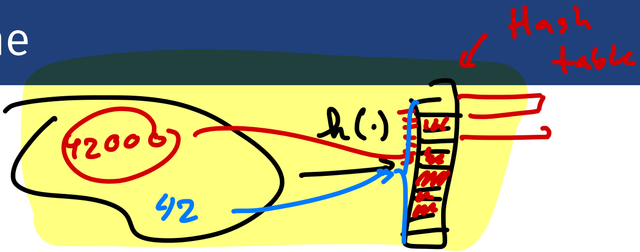
Left 5 4 5

Right 5 6 5

7. Hashing

Hashing well-done





Useful Hashing...

- distributes the keys as uniformly as possible in the hash table.
- avoids probing over long areas of used entries (e.g. primary clustering).
- avoids using the same probing sequence for keys with the same hash value (e.g. secondary clustering).

offset(k, j) "probing method"

offset(k, j) = j

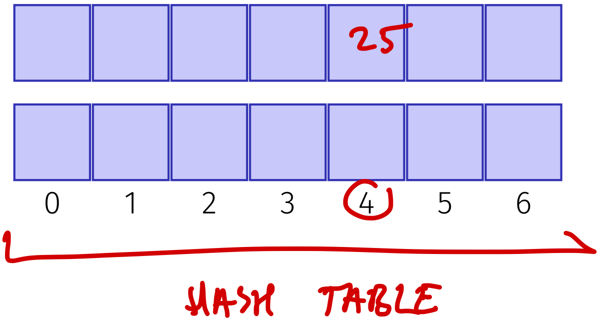
= j + h'(k)

14, 15, ..., 19, 20

Hashing Examples

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $\mathit{offset}(j, k) = j$.
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.



$h(25) = 4$

Hashing Examples

$h(4) = 4 \% 7 = 4$ $\quad$ $h(17) = 3$

Insert the keys ~~25~~, 4, 17, 45 into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $s := \mathit{offset}(j, k) = j$,
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

			17	25	4	

0	1	2	3	4	5	6

$H(j,k) := h(k) + s(j,k)$

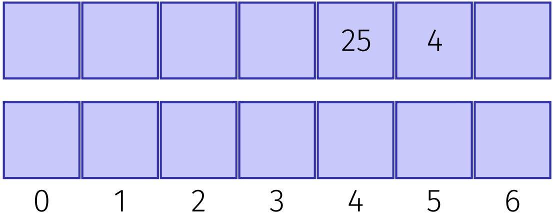
$H(0,4) = 4 + 0$

$H(1,4) = 4 + s(1,4) = 5$, $j = 1$

Hashing Examples

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $\mathit{offset}(j, k) = j$.
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

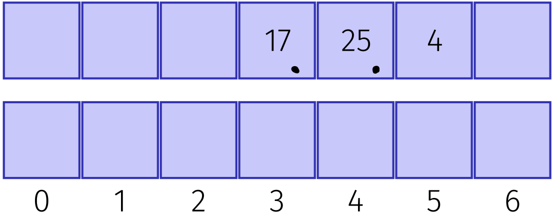


Hashing Examples

$h(45) = 3$ ↯

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \textit{offset}(j, k)$:

- linear probing,
  $\textit{offset}(j, k) = j$.
- Double Hashing,
  $\textit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.



Hashing Examples

j: "how many times have I tried storing the key k already"

Insert the keys 25, 4, 17, 45 into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j,k)$:

- linear probing,
  $\mathit{offset}(j,k) = j$.
- Double Hashing,
  $\mathit{offset}(j,k) = j \cdot (1 + (k \bmod 5))$.

"second hash function"

0	1	2	3	4	5	6
			17	25	4	45
				25		

Hashing Examples

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $\mathit{offset}(j, k) = j$.
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

0	1	2	3	4	5	6
			17	25	4	45
		4		25		

$\mathit{offset}(1,4) = 1 \cdot (1 + (4))$
$= 1 \cdot (5) = 5$

Hashing Examples

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $\mathit{offset}(j, k) = j$.
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

0	1	2	3	4	5	6
			17	25	4	45
		4	17	25		

$h(17) = 4$

$\mathit{offset}(1,17) = 1 \cdot (1 + 17 \% 5) = 3$

Hashing Examples

$h(45) = 45 \bmod 7 = 3$

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing, $\mathit{offset}(j, k) = j$.
- Double Hashing, $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

0	1	2	3	4	5	6
			17	25	4	45

0	1	2	3	4	5	6
		4	17	25	45	

$H(j,k) = h(k) + \mathit{offset}(j,k)$

$= 3 + 0 = 3$

$j = 1$

$H(j,k) = 3 + 1 \cdot (1 + (45 \bmod 5))$

$= h(k) + j \cdot (1) \; 2$

Hashing Examples

Insert the keys $25, 4, 17, 45$ into the hash table, using the function $h(k) = k \bmod 7$ and probing to the right, $h(k) + \mathit{offset}(j, k)$:

- linear probing,
  $\mathit{offset}(j, k) = j$.
- Double Hashing,
  $\mathit{offset}(j, k) = j \cdot (1 + (k \bmod 5))$.

	0	1	2	3	4	5	6
linear probing				17	25	4	45
Double Hashing			4	17	25	45	

Quiz: Hashing

A hash table of length 10 uses closed hashing with hash function $h(k) = k \bmod 10$, and linear probing (probing goes to the right). After inserting five values into an empty hash table, the table is as shown below.

0	1	2	3	4	5	6	7	8	9
		32	52	33	74	96			

Which of the following *choice(s)* give possible order(s) in which the key values could have been inserted in the table?

(A) 32, 33, 52, 96, 74

(B) 32, 52, 33, 74, 96

(C) 32, 52, 74, 96, 33

(D) 96, 32, 52, 33, 74

Quiz: Hashing

A hash table of length 10 uses closed hashing with hash function $h(k) = k \bmod 10$, and linear probing (probing goes to the right). After inserting five values into an empty hash table, the table is as shown below.

0	1	2	3	4	5	6	7	8	9
		32	52	33	74	96			

Which of the following *choice(s)* give possible order(s) in which the key values could have been inserted in the table?

(A) 32, 33, 52, 96, 74

(B) 32, 52, 33, 74, 96

(C) 32, 52, 74, 96, 33

(D) 96, 32, 52, 33, 74

Vocabulary of related concepts

Vocabulary of related concepts

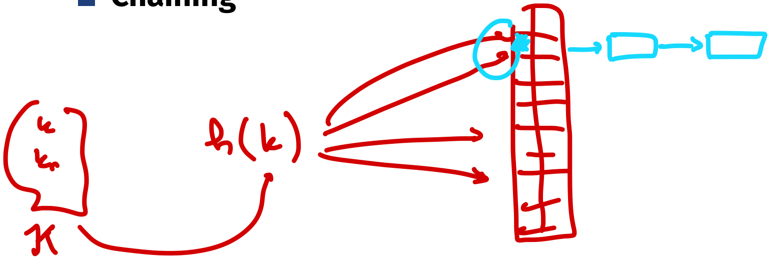
- **Prehashing**

Vocabulary of related concepts

- **Prehashing**
  $ph(k) \rightarrow \mathbb{N}$. i.e. mapping keys onto integers for further use
- **Collision**

Vocabulary of related concepts

- **Prehashing**
  $ph(k) \to \mathbb{N}$. i.e. mapping keys onto integers for further use
- **Collision**
  $h(k_i) = h(k_j)\ i \neq j$. i.e. hash function maps two different keys onto same integer
- **Chaining**



Vocabulary of related concepts

- **Prehashing**
  $ph(k) \to \mathbb{N}$. i.e. mapping keys onto integers for further use
- **Collision**
  $h(k_i) = h(k_j)\, i \neq j$. i.e. hash function maps two different keys onto same integer
- **Chaining**
  Store all $h(k_i) = h(k_j)\, i \neq j$ in one (worst case very long) linked list. Positive: can overcommit (more entries than slots) and easy to remove entries. Negative: Memory consumption of the chains. Alternative: Closed hashing with open addressing
- **Closed Hashing**

Vocabulary of related concepts

- **Prehashing**
  $ph(k) \to \mathbb{N}$. i.e. mapping keys onto integers for further use
- **Collision**
  $h(k_i) = h(k_j)\, i \neq j$. i.e. hash function maps two different keys onto same integer
- **Chaining**
  Store all $h(k_i) = h(k_j)\, i \neq j$ in one (worst case very long) linked list. Positive: can overcommit (more entries than slots) and easy to remove entries. Negative: Memory consumption of the chains. Alternative: Closed hashing with open addressing
- **Closed Hashing** (→ probing)
  Entries stays in table

Vocabulary of related concepts

Vocabulary of related concepts

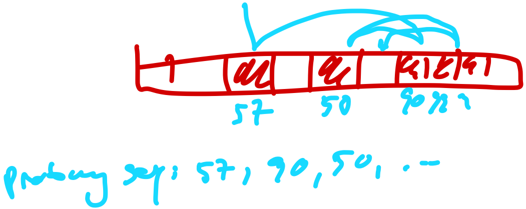
- **Simple Uniform Hashing**

Vocabulary of related concepts

- **Simple Uniform Hashing**
  each key is equally likely to hash to any of the m slots, independently of where any other key has hashed to
- **Uniform Hashing**

Vocabulary of related concepts

- **Simple Uniform Hashing**
  each key is equally likely to hash to any of the m slots, independently of where any other key has hashed to
- **Uniform Hashing**
  the probing sequence of each key is equally likely to be any of the $m!$ permutations of the possible sequences over the hash table of size m
- **Open Addressing**



Vocabulary of related concepts

- **Simple Uniform Hashing**
  each key is equally likely to hash to any of the m slots, independently of where any other key has hashed to
- **Uniform Hashing**
  the probing sequence of each key is equally likely to be any of the $m!$ permutations of the possible sequences over the hash table of size m
- **Open Addressing**
  Position in hash table is not fixed and depends on previous entries

8. Code-Example: Hashtables, Hashfunctions and Collisions

Hands-on example: importance of a well designed hashing strategy-

9. Old Exam Question

Hashing

Eine Hashtabelle mit 10 Einträgen verwendet offene Adressierung mit der Hash-Funktion $h(k) = k \bmod 10$, mit linearer Sondierung (Sondierung geht nach rechts). Nachdem sechs Werte in die initial leere Hashtabelle eingefügt wurden, sieht die Hashtabelle wie folgt aus.

A hash table of length 10 uses open addressing with hash function $h(k) = k \bmod 10$, and linear probing (probing goes to the right). After inserting 6 values into an empty hash table, the table is as shown below.

0	1	2	3	4	5	6	7	8	9
70	9	42	20	10					69

Welche der folgenden Möglichkeiten bezeichnen/bezeichnet jeweils eine Reihenfolge, in der die Schlüssel in die Hashtabelle eingefüllt werden konnten?

Which of the following choice(s) give possible order(s) in which the key values could have been inserted in the table?

(A) 70, 42, 69, 9, 20, 10

(B) 42, 69, 20, 10, 70, 9

(C) 69, 42, 70, 9, 20, 10

(D) 42, 69, 9, 70, 20, 10

Hashing – Solution

Eine Hashtabelle mit 10 Einträgen verwendet offene Adressierung mit der Hash-Funktion $h(k) = k \bmod 10$, mit linearer Sondierung (Sondierung geht nach rechts). Nachdem sechs Werte in die initial leere Hashtabelle eingefügt wurden, sieht die Hashtabelle wie folgt aus.

A hash table of length 10 uses open addressing with hash function $h(k) = k \bmod 10$, and linear probing (probing goes to the right). After inserting 6 values into an empty hash table, the table is as shown below.

0	1	2	3	4	5	6	7	8	9
70	9	42	20	10					69

Welche der folgenden Möglichkeiten bezeichnen/bezeichnet jeweils eine Reihenfolge, in der die Schlüssel in die Hashtabelle eingefüllt werden konnten?

Which of the following choice(s) give possible order(s) in which the key values could have been inserted in the table?

(A) 70, 42, 69, 9, 20, 10

(B) 42, 69, 20, 10, 70, 9

(C) 69, 42, 70, 9, 20, 10

(D) 42, 69, 9, 70, 20, 10

```
(A, C)
```

10. Tips for **code** expert

Finding a Sub-Array

- Given: two integer arrays $A = (a_0, \ldots, a_{n-1})$ and $B = (b_0, \ldots, b_{k-1})$
- Task: Find position of B in A.

Finding a Sub-Array

- Given: two integer arrays $A = (a_0, \ldots, a_{n-1})$ and $B = (b_0, \ldots, b_{k-1})$
- Task: Find position of B in A.
- Naive: Loop through A, check whether the following k entries match B.

Finding a Sub-Array

- Given: two integer arrays $A = (a_0, \ldots, a_{n-1})$ and $B = (b_0, \ldots, b_{k-1})$
- Task: Find position of B in A.
- Naive: Loop through A, check whether the following k entries match B.
  - $O(nk)$ comparison operations

Finding a Sub-Array

- Given: two integer arrays $A = (a_0, \ldots, a_{n-1})$ and $B = (b_0, \ldots, b_{k-1})$
- Task: Find position of B in A.
- Naive: Loop through A, check whether the following k entries match B.
  - $O(nk)$ comparison operations
- Solution using hashing: Calculate hash $h(B)$ and compare it to $h((a_i, a_{i+1}, \ldots, a_{i+k-1}))$.
- Avoid re-computing $h((a_i, a_{i+1}, \ldots, a_{i+k-1})$ for each $i \implies O(n)$ expected

Sliding Window Hash

- Possible hash function: sum of all elements:
  - Can be updated easily: subtract a_i and add a_{i+k}.
  - However: bad hash function

Sliding Window Hash

- Possible hash function: sum of all elements:
  - Can be updated easily: subtract a_i and add a_{i+k}.
  - However: bad hash function
- Better:

$$H_{c,m}((a_i, \cdots, a_{i+k-1})) = \left(\sum_{j=0}^{k-1} a_{i+j} \cdot c^{k-j-1} \right) \bmod m$$

  - $c = 1021$ prime number
  - $m = 2^{15}$ `int`, no overflows at calculations

Sliding Window Hash

Make sure that

- the algorithm computes c^k only once,
- all computations are modulo m for all values in order not to get an overflow (recall the rules of modular arithmetic), and
- the values are always positive (e.g., by adding multiples of m).

Computing with Modulo

$$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$
$$(a - b) \bmod m = ((a \bmod m) - (b \bmod m) + m) \bmod m$$
$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

Exercise: Compute

$$12746357 \bmod 11$$

Computing Modulo

Exercise: Compute

$$12746357 \bmod 11$$

Computing Modulo

Exercise: Compute

$$
\begin{aligned}
&12746357 \bmod 11 \\
&= (7 + 5 \cdot 10 + 3 \cdot 10^2 + 6 \cdot 10^3 + 4 \cdot 10^4 + 7 \cdot 10^5 + 2 \cdot 10^6 + 1 \cdot 10^7) \bmod 11
\end{aligned}
$$

Computing Modulo

Exercise: Compute

$$
\begin{aligned}
&12746357 \bmod 11\\
&= (7 + 5 \cdot 10 + 3 \cdot 10^2 + 6 \cdot 10^3 + 4 \cdot 10^4 + 7 \cdot 10^5 + 2 \cdot 10^6 + 1 \cdot 10^7) \bmod 11\\
&= (7 + 50 + 3 + 60 + 4 + 70 + 2 + 10) \bmod 11
\end{aligned}
$$

For the second equality we used the fact that $10^2 \bmod 11 = 1$.

Computing Modulo

Exercise: Compute

$$\begin{aligned}&12746357 \bmod 11\\&= (7 + 5 \cdot 10 + 3 \cdot 10^2 + 6 \cdot 10^3 + 4 \cdot 10^4 + 7 \cdot 10^5 + 2 \cdot 10^6 + 1 \cdot 10^7) \bmod 11\\&= (7 + 50 + 3 + 60 + 4 + 70 + 2 + 10) \bmod 11\\&= (7 + 6 + 3 + 5 + 4 + 4 + 2 + 10) \bmod 11\end{aligned}$$

For the second equality we used the fact that $10^2 \bmod 11 = 1$.

Computing Modulo

Exercise: Compute

$$
\begin{aligned}
&12746357 \bmod 11 \\
&= (7 + 5 \cdot 10 + 3 \cdot 10^2 + 6 \cdot 10^3 + 4 \cdot 10^4 + 7 \cdot 10^5 + 2 \cdot 10^6 + 1 \cdot 10^7) \bmod 11 \\
&= (7 + 50 + 3 + 60 + 4 + 70 + 2 + 10) \bmod 11 \\
&= (7 + 6 + 3 + 5 + 4 + 4 + 2 + 10) \bmod 11 \\
&= 8 \bmod 11.
\end{aligned}
$$

For the second equality we used the fact that $10^2 \bmod 11 = 1$.

11. Outro

General Questions?

See you next time

Have a nice week!

`[rw::gettogether]` is this Friday!