# Informatik II

## Übung 7

FS 2020

# Program Today

1. Repetition Lectures
2. String-Hashing and Computing with Modulo
3. In-Class-Exercises: Sliding Window

# Hashing well-done

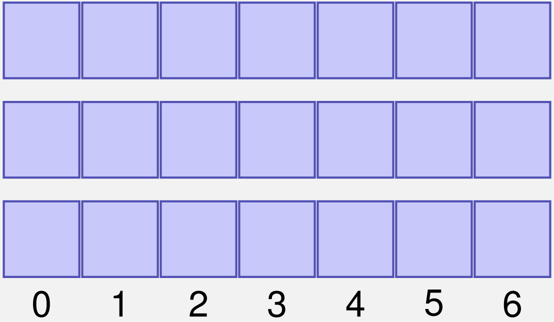
Useful Hashing. . .

- 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).

# 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, $h(k) + s(j, k)$:

- linear probing,
  $s(j, k) = j$.
- quadratic probing,
  $s(j, k) = (-1)^{j+1} \lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j, k) = j \cdot (1 + (k \bmod 5))$.



# 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, $h(k) + s(j, k)$:

- linear probing,
  $s(j, k) = j$.
- quadratic probing,
  $s(j, k) = (-1)^{j+1} \lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j, k) = j \cdot (1 + (k \bmod 5))$.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | 25 | | |
| | | | | | | |
| | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | 25 | 4 | |
| | | | | | | |
| | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | 17 | 25 | 4 | |
| | | | | | | |
| | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1} \lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1} \lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j, k)$:

- linear probing,
  $s(j, k) = j$.
- quadratic probing,
  $s(j, k) = (-1)^{j+1} \lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j, k) = j \cdot (1 + (k \bmod 5))$.

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

# 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, $h(k) + s(j,k)$:

- linear probing, $s(j,k) = j$.
- quadratic probing, $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing, $s(j,k) = j \cdot (1 + (k \bmod 5))$.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| linear probing | | | | 17 | 25 | 4 | 45 |
| quadratic probing | | | 45 | 17 | 25 | 4 | |
| Double Hashing | | | 4 | 17 | 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, $h(k) + s(j,k)$:

- linear probing,
  $s(j,k) = j$.
- quadratic probing,
  $s(j,k) = (-1)^{j+1}\lceil j/2 \rceil^2$.
- Double Hashing,
  $s(j,k) = j \cdot (1 + (k \bmod 5))$.

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

# Computing with Modulo

$$
\begin{aligned}
(a+b) \bmod m &= ((a \bmod m) + (b \bmod m)) \bmod m \\
(a-b) \bmod m &= ((a \bmod m) - (b \bmod m) + m) \bmod m \\
(a \cdot b) \bmod m &= ((a \bmod m) \cdot (b \bmod m)) \bmod m
\end{aligned}
$$

**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 $10^2 \bmod 11 = 1$.

# Implementation Hash(String) in Java

$$h_{c,m}(s) = \left( \sum_{i=0}^{k-1} s_{k-1-i} \cdot c^i \right) \bmod m$$

```
int ComputeHash(int C, int M, String s) {
  int hash = 0;
  for (int i = 0; i < s.length(); ++i){
    hash = (C * hash % M + s.charAt(i)) % M;
  }
  return hash;
}
```

# In-Class-Exercises: Sliding Window

Code-Expert $\rightarrow$ code examples (7)

Given a String `text` of length $n$, we want to find the shortest substring $\texttt{text}[l, r]$, which contains each of the characters ‘a’, ‘b’ and ‘c’ at least once.

# In-Class-Exercises: Sliding Window

Code-Expert $\rightarrow$ code examples (7)

Given a String `text` of length $n$, we want to find the shortest substring $\texttt{text}[l, r]$, which contains each of the characters ‘a’, ‘b’ and ‘c’ at least once.

- Brute-Force: Testing all $\frac{n\cdot(n-1)}{2}$ substrings needs time $\mathcal{O}(n^3)$.

# In-Class-Exercises: Sliding Window

Code-Expert $\rightarrow$ code examples (7)

Given a String `text` of length $n$, we want to find the shortest substring $\texttt{text}[l, r]$, which contains each of the characters 'a', 'b' and 'c' at least once.

- Brute-Force: Testing all $\frac{n \cdot (n-1)}{2}$ substrings needs time $\mathcal{O}(n^3)$.
- Idea: Consider a fixed substring $\texttt{text}[l, r]$:
  - If it is missing some characters $\rightarrow$ increase substring length.
  - If it contains all 3 characters $\rightarrow$ decrease substring length.

# In-Class-Exercises: Sliding Window

Code-Expert $\rightarrow$ code examples (7)

Given a String `text` of length $n$, we want to find the shortest substring $\texttt{text}[l, r]$, which contains each of the characters 'a', 'b' and 'c' at least once.

- Brute-Force: Testing all $\frac{n \cdot (n-1)}{2}$ substrings needs time $\mathcal{O}(n^3)$.
- Idea: Consider a fixed substring $\texttt{text}[l, r]$:
  - If it is missing some characters $\rightarrow$ increase substring length.
  - If it contains all 3 characters $\rightarrow$ decrease substring length.
- Sliding Window Approach.

# In-Class-Exercises: Sliding Window

Sliding Window Approach:

# In-Class-Exercises: Sliding Window

Sliding Window Approach:

**Time:** $\mathcal{O}(n)$.

- In each step we enlarge the sliding window to the right or decrease it on the left. Hence there can be at most $2n$ steps.
- We hash a constant number of characters, hence HashMap operations will take time $\mathcal{O}(1)$.

# Comparison to Rabin-Karp Exercise

**Rabin-Karp:** We are looking for a specific Substring “abc”, and not just its individual Characters ‘a’, ‘b’, ‘c’!

- Easier, since our Sliding Window always has the same length!
- But at the same time more difficult, since the order of the characters matters!

# Questions / Suggestions?