

# Strictly-regular number system and data structures

Amr Elmasry<sup>1)</sup> and Claus Jensen<sup>2)</sup>

**Jyrki Katajainen<sup>3)</sup>**

- 1) Max-Planck-Institut für Informatik
- 2) The Royal Library
- 3) University of Copenhagen

These slides are available at <http://www.cphst1.dk>

# What is the problem?

---

In the decimal number system, introduced to the west by Muhammad ibn Mūsā al-Khwārizmī [825], a single increment may incur many digit changes!

$$\begin{array}{r} 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\ \quad \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \quad 9 \\ + \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad 1 \\ \hline 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \end{array}$$

The binary number system has the same problem.

# Why is this a problem?

---

1 1 0 ... <sup>*i*</sup> 1 ... 1 0  
1 1 0 ... 1 ... 1 1



An addition of two bits can  
be a heavy operation!

# Number systems

---

Let  $d$  be a positive integer

**rep( $d$ ):**  $\langle d_0, d_1, \dots, d_{k-1} \rangle$  ( $d_0$  is the least significant digit)

**value( $d$ ):**  $\sum_{i=0}^{k-1} d_i \times w_i$

**b-ary:**  $w_i = b^i$

**Decimal:**  $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ;  $w_i = 10^i$

**Binary:**  $d_i \in \{0, 1\}$ ;  $w_i = 2^i$

**Redundant binary:**  $d_i \in \{0, 1, 2\}$ ;  $w_i = 2^i$

**Regular binary:**  $d_i \in \{0, 1, 2\}$ ;  $w_i = 2^i$ ; Every sequence of digits is of the form  $(0 \mid 1 \mid 01^*2)^*$  [Clancy & Knuth 1977]

**Zeroless regular:**  $d_i \in \{1, 2, 3\}$ ;  $w_i = 2^i$ ; Every sequence of digits is of the form  $(1 \mid 2 \mid 12^*3)^*$  [Brodal 1995]

# What is the most economical system?

---

Develop a number system for which

- $\max \{d_i \mid i \in \{0, 1, \dots, k - 1\}\}$  is as **small** as possible for all  $d$ ;
- an increment at any position  $i$  ( $increment(d, i)$ ) generates as few digit changes as possible in the **worst case**; and
- a decrement at any position  $i$  ( $decrement(d, i)$ ) generates as few digit changes as possible in the **worst case**.

# Strictly-regular system

---

**Digits:**  $d_i \in \{0, 1, 2\}$

**Strict regularity:** The sequence from the least-significant to the most-significant digit is of the form  $(1^+ | 01^*2)^* (\varepsilon | 01^+)$

**Extreme digits:** 0 and 2

a) 1111111 **yes**

b) 11011211101 **yes**

c) 1201 **no**

d) 1110101 **no**

# Increment example

---

**Notation:** Digit  $d_i$  to be increased is displayed in **red**.  $d_a$  is the first extreme digit after  $d_i$ ,  $k$  is a non-negative integer,  $\alpha$  denotes any combination of  $1^+$  and  $01^*2$  blocks, and  $\omega$  any combination of  $1^+$  and  $01^*2$  blocks followed by at most one  $01^+$  block.

**Initial configuration:**  $\alpha 01^* \mathbf{1} 1^* \mathbf{2} 11^k \omega$

**Action:**  $d_i \leftarrow \mathbf{2}$ ;  $d_a \leftarrow d_a - 2$ ;  $d_{a+1} \leftarrow d_{a+1} + 1$

**Final configuration:**  $\alpha 01^* \mathbf{2} 1^* \mathbf{0} 2 11^k \omega$

**Remark:** This is one of 19 cases considered in our correctness proof.

# General algorithm

---

**Subroutine** *fix-carry*( $\mathbf{d}, i$ ): Assert that  $d_i \geq 2$ . Perform  $d_i \leftarrow d_i - 2$  and  $d_{i+1} \leftarrow d_{i+1} + 1$ .

---

**Algorithm** *increment*( $\mathbf{d}, i$ ):

---

- 1:  $++d_i$
  - 2: Let  $d_b$  be the first extreme digit before  $d_i$ ,  $d_b \in \{0, 2, \text{undefined}\}$
  - 3: Let  $d_a$  be the first extreme digit after  $d_i$ ,  $d_a \in \{0, 2, \text{undefined}\}$
  - 4: **if**  $d_i = 3$  **or** ( $d_i = 2$  **and**  $d_b \neq 0$ )
  - 5:     *fix-carry*( $\mathbf{d}, i$ )
  - 6: **else if**  $d_a = 2$
  - 7:     *fix-carry*( $\mathbf{d}, a$ )
-

# Full repertoire of operations

---

*increment*( $\mathbf{d}, i$ ): Assert that  $i \in \{0, 1, \dots, k\}$ . Perform  $++d_i$  resulting in  $\mathbf{d}'$ , i.e.  $value(\mathbf{d}') = value(\mathbf{d}) + w_i$ . Make  $\mathbf{d}'$  valid without changing its value.

*decrement*( $\mathbf{d}, i$ ): Assert that  $i \in \{0, 1, \dots, k - 1\}$ . Perform  $--d_i$  resulting in  $\mathbf{d}'$ , i.e.  $value(\mathbf{d}') = value(\mathbf{d}) - w_i$ . Make  $\mathbf{d}'$  valid without changing its value.

*cut*( $\mathbf{d}, i$ ): Cut  $rep(\mathbf{d})$  into two valid sequences having the same value as the numbers corresponding to  $\langle d_0, d_1, \dots, d_{i-1} \rangle$  and  $\langle d_i, d_{i+1}, \dots, d_{k-1} \rangle$ .

*concatenate*( $\mathbf{d}, \mathbf{d}'$ ): Concatenate  $rep(\mathbf{d})$  and  $rep(\mathbf{d}')$  into one valid sequence that has the same value as  $\langle d_0, d_1, \dots, d_{k-1}, d'_0, d'_1, \dots, d'_{k'-1} \rangle$ .

*add*( $\mathbf{d}, \mathbf{d}'$ ): Construct a valid sequence  $\mathbf{d}''$  such that  $value(\mathbf{d}'') = value(\mathbf{d}) + value(\mathbf{d}')$ .

# Properties

---

- Increments, decrements, catenations, and cuts involve  $O(1)$  digit changes in the worst case
- Addition of two  $k$ -digit numbers involve at most  $k$  carry propagations
- The sum of digits of a  $k$ -digit number is either  $k$  or  $k - 1$  (compactness property)
- The value of a  $k$ -digit number is at least  $\phi^k - 1$  where  $\phi$  is the golden ratio (exponentiality property)

## Related work

---

**Regular system:** Allows increments at any position with  $O(1)$  digit changes [Clancy & Knuth 1977]

**Zeroless regular system:** Allows increments at any position with  $O(1)$  digit changes, and has the exponentiality property [Brodal 1995]

**Two regular systems back to back:**  $d_i \in \{0, 1, 2, 3, 4, 5\}$ ; Allows increments and decrements at any position with  $O(1)$  digit changes [Kaplan & Tarjan 1995; Kaplan & Tarjan 1996; Brodal 1996]

**Extended regular system:**  $d_i \in \{0, 1, 2, 3\}$ ; Every 3 is preceded by at least one  $\{0, 1\}$  before the previous 3 or running out of digits, and every 0 is preceded by at least one  $\{2, 3\}$  before the previous 0 or running out of digits; Allows increments and decrements at any position with  $O(1)$  digit changes [Clancy & Knuth 1977; Kaplan, Shafrir & Tarjan 2002]

# Application: Faster meldable priority queues

---

- **fast meldable priority queues** [Brodal 1995]

<b>operations</b>	<b>worst-case time</b>
<i>find-min, insert, meld</i>	$O(1)$
<i>delete</i>	$O(\lg n)$ ( $n$ current size) $\beta \lg n + O(1)$ element comparisons

Here  $\beta$  is the famous Brodal's constant

[Brodal 1995]:  $\beta = 7$  (proved in this paper)

[Jensen 2009]:  $\beta = 3$  (sketched in this paper)

[this paper]:  $\beta = 2$

[folklore]:  $\beta \geq 1$  (follows from comparison-based sorting)

**My conjecture:**  $\lg n + O(\lg \lg n)$  element comparisons per *delete* possible

# Other applications

---

- **fat heaps** [Kaplan, Shafrir & Tarjan 2002]

operations	worst-case time
<i>find-min, insert, decrease</i>	$O(1)$
<i>meld</i>	$O(\min \{\lg m, \lg n\})$ ( $m, n$ heap sizes)
<i>delete</i>	$O(\lg n)$ $2.53 \lg n + O(1)$ element comparisons

☞ Could be implemented without any number systems

- **two-tier relaxed heaps** [Elmasry, Jensen & Katajainen 2008]

operations	worst-case time
<i>find-min, insert, decrease</i>	$O(1)$
<i>meld</i>	$O(\min \{\lg m, \lg n\})$ ( $m, n$ heap sizes)
<i>delete</i>	$O(\lg n)$ $\lg n + O(\lg \lg n)$ element comparisons

☞ No improvement in constant factors

## Other applications (cont.)

---

- **penultimate meldable priority queues** [Brodal 1996]

<b>operations</b>	<b>worst-case time</b>
<i>find-min, insert, decrease, meld</i>	$O(1)$
<i>delete</i>	$O(\lg n)$ $\beta \lg n + O(1)$ element comparisons

☞ We could not support ternary arithmetic

☞ We could not support increments and decrements at arbitrary position in  $O(1)$  worst-case time

**My conjecture:**  $\beta \leq 20$  provable

## Further reading

---

Elmasry, Jensen, and Katajainen, Strictly-regular number system and data structures, *Proceedings of 12th Scandinavian Symposium and Workshops on Algorithm Theory*, Lecture Notes in Computer Science, Springer-Verlag (2010)

Elmasry, Jensen, and Katajainen, The magic of a number system, *Proceedings of the 5th International Conference on Fun with Algorithms*, Lecture Notes in Computer Science, Springer-Verlag (2010)