

# Two number systems and one application

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

**Jyrki Katajainen<sup>1)</sup>**

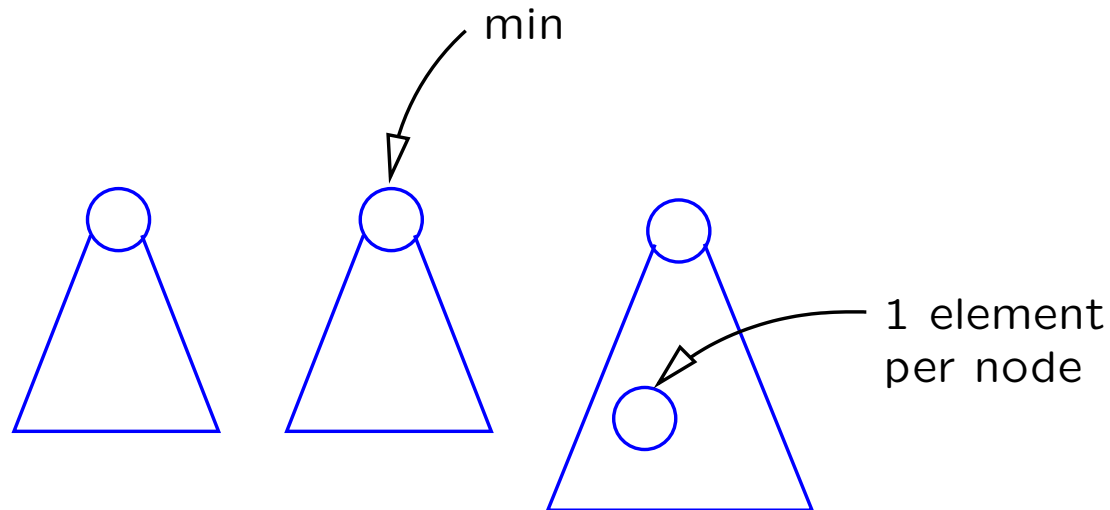
1) University of Copenhagen

2) The Royal Library

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

# Application

---



**Representation of a priority queue:** An ordered collection of pointer-based **perfect** binary heaps,  $2^{i+1} - 1$  elements for  $i = 0, 1, \dots$

**Operations:** *find-min*, *insert*, *borrow* (What is this?), *delete* (Is *delete-min* missing?), *meld*

Credit: [Williams 1964]

# Number system

---

**Digit set:**  $d_i \in \{0, 1, \dots\}$

**Representation of a number:**  $\langle d_0, d_1, \dots, d_{\ell-1} \rangle$  ( $d_0$  least significant,  $d_{\ell-1} \neq 0$ )

**Weight set:**  $\{w_i \mid i \in \{0, 1, \dots\}\}$

**Decimal value:**  $\sum_{i=0}^{\ell-1} d_i \times w_i$

**Operations:** *increment, decrement, addition, cut, catenation*

# Some number systems

---

**Decimal:**  $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ;  $w_i = 10^i$  [al-Khwārizmī 825]

**Unary:**  $d_i \in \{1\}$ ;  $w_i = 1$

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

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

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

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

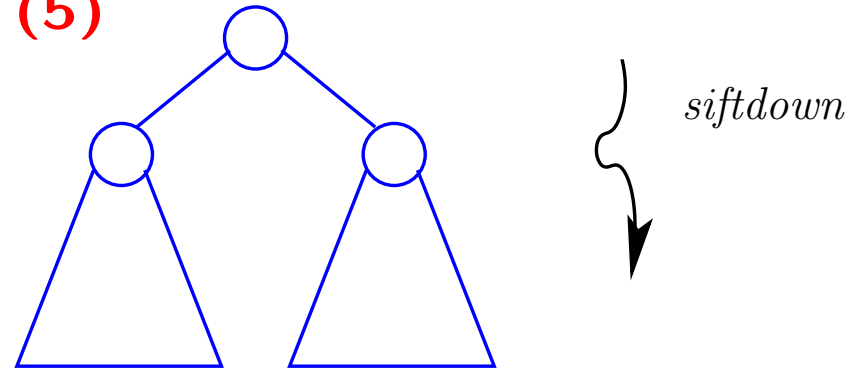
**Zeroless regular:**  $d_i \in \{1, 2, 3\}$ ;  $w_i = 2^i$ ; Every string of digits is of the form  $(1 \mid 2 \mid 12^*3)^*$  [Brodal 1995]

# Link

- (1) digit  $d_i$  at position  $i$
- (2) *increment*
- (3) *decrement*
- (4) *addition*
- (5) digit transfer  $2w_i + 1 = w_{i+1}$

- (6)  $d_i = O(1) \forall i$

- (1)  $d_i$  heaps of size  $w_i$
- (2) *insert*
- (3) *borrow*
- (4) *meld*
- (5)



- (6)  $O(\lg n)$  heaps;  $n$ : #elements

Credit: [Vuillemin 1978]

# Canonical skew

---

A positive integer  $n$  is represented as a string  $\langle d_0, d_1, \dots, d_{\ell-1} \rangle$  of digits, least-significant digit first, such that

- $d_i \in \{0, 1, 2\} \forall i \in \{0, 1, \dots, \ell - 1\}$ , and  $d_{\ell-1} \neq 0$ ,
- if  $d_j = 2$ , then  $d_i = 0 \forall i \in \{0, 1, \dots, j - 1\}$ ,
- $w_i = 2^{i+1} - 1 \forall i \in \{0, 1, \dots, \ell - 1\}$ , and
- the decimal value of  $n$  is  $\sum_{i=0}^{\ell-1} d_i w_i$ .

**Example:** Unique representation of integer  $52_{10}$

position $i$	0	1	2	3	4
digit $d_i$	0	2	0	1	1
weight $w_i$	1	3	7	15	31

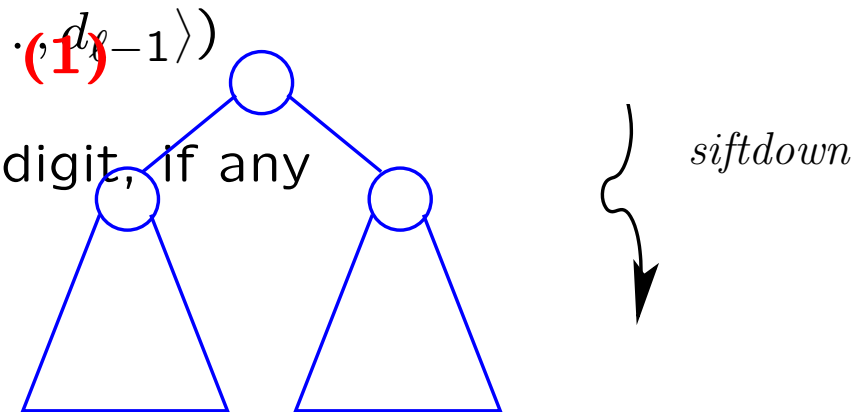
Credit: [Myers 1983]


# Increment/insert

**Algorithm** *increment*( $\langle d_0, d_1, \dots, d_{k-1} \rangle$ )

- 1: let  $d_j$  be the first non-zero digit, if any
- 2: **if**  $d_j$  exists **and**  $d_j = 2$
- 3:      $d_j \leftarrow 0$
- 4:     increase  $d_{j+1}$  by 1 **(1)**
- 5: **else**
- 6:     increase  $d_0$  by 1 **(2)**

**(3)** at most 2 digit changes



**(2)** Add a new node  to the collection

**Extra:** Update the min pointer, if necessary

**(3)**  $O(\lg n)$  worst-case time;  
 $n$ : #elements

Credit: [Bansal et al. 2003]

# Decrement/borrow

**Algorithm** *decrement*( $\langle d_0, d_1, \dots, d_{\ell-1} \rangle$ )

1: **assert**  $\langle d_0, d_1, \dots, d_{\ell-1} \rangle$  is not empty

2: let  $d_j$  be the first non-zero digit

3: decrease  $d_j$  by 1 **(1)**

4: **if**  $j \neq 0$

5:      $d_{j-1} \leftarrow 2$  **(2)**

**(3)** at most 2 digit changes

**(1)** Remove the root of the smallest heap

**(2)** Add its subtrees, if any, to the collection

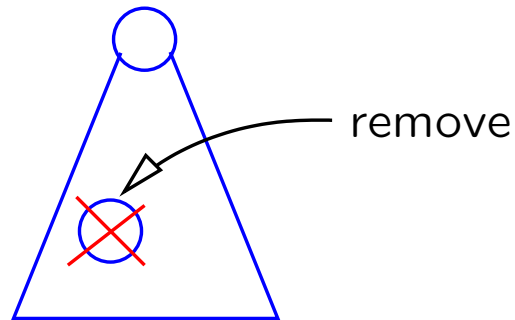
**(3)**  $O(1)$  worst-case time

**Problem:** The min pointer may be invalidated!

**Solution:** Swap the root with the next root, or with its own left child before the removal

# Delete

---



- (1) Borrow a node ○
- (2) Replace ~~○~~ with ○
- (3) Perform *siftdown* or *siftup* for ○
- (4) Update the min pointer, if necessary
- (5)  $O(\lg n)$  worst-case time;  $n$ : #elements

# Summary

---

<b>Structure</b> <b>Operation</b>	Array-based binary heap	Canonical skew	Regular skew
<i>find-min</i>	$O(1)$	$O(1)$	$O(1)$
<i>insert</i>	$O(\lg n)$	$O(\lg n)$	$O(1)$
<i>borrow</i>	$O(1)$	$O(1)$	$O(1)$
<i>delete</i>	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
<i>meld</i>	$O(n)$	$O((\lg n)^2)$	$O((\lg m)^2)$

worst-case performance;  $m \leq n$

**Open:** Is faster *decrease* or *meld* possible? (Amortized bounds matching those achievable for Fibonacci heaps can be obtained, except for *decrease*; this is not shown in our paper though.)

# Regular skew

---

**Cost of a digit change:**  $O(j)$  at position  $j$

**Discretization:** Initially,  $j$  bricks at position  $j$ , i.e.  $b_j = j$

**Digit set:**  $d_i \in \{0, 1, 2\} \forall i$ ; when  $b_k > 0$ ,  $d_k$  is said to form a wall ( $\boxed{1}$  or  $\boxed{2}$ ) of  $b_k$  bricks

**Incremental digit changes:** Remove some bricks from some walls in addition to the normal actions; do not transfer digits across any walls

**Representation of a number:** Otherwise an integer is represented as in any skew system

Credit: [Carlsson et al. 1988]

# Regularity conditions

---

**Preceding 0:** 02 or 0 $\boxed{1}$  or 0 $\boxed{2}$  ( $d_0$  can be a 2)

**Absorbing 0:** 21\*0 or  $\boxed{2}$ 1\*0, this 0 should not be preceding ( $d_{\ell-1}$  can be a 2 or  $\boxed{2}$ )

**Critical 0:** If  $d_k$  forms a wall, then

(i)  $\exists j < k - 1$  such that  $d_j = 0$ , and  $b_k \leq j + 1$ , or

(ii)  $\exists j < k - 1$  such that  $d_j = 0$ ,  $d_j$  is absorbing, and  $b_k \leq j$

**Example:** One possible representation of integer  $143_{10}$

position $i$	0	1	2	3	4	5
critical 0				$d_1$		$d_2$
digit $d_i$	2	0	0	$\boxed{1}$	0	$\boxed{2}$
variable $b_i$				1		3

# Increment/insert

**Algorithm**  $transfer(\langle d_0, d_1, \dots, d_{\ell-1} \rangle, j)$

- 1: **assert**  $d_j = 2$
- 2:  $d_j \leftarrow 0$
- 3: increase  $d_{j+1}$  by 1 **(1)**
- 4:  $b_{j+1} \leftarrow j + 1$  **(2)**

**(1)** Add a heap to the collection

**(2)** Initiate *siftdown*

**Algorithm**  $increment(\langle d_0, d_1, \dots, d_{\ell-1} \rangle)$

- 1: let  $d_k$  be the first wall, if any
- 2: let  $d_j$  be the first 2 for which  $b_j = 0$ , if any
- 3: **if**  $d_j$  exists **and** ( $d_k$  does not exist **or**  $d_j < d_k$ ) **then** **update** the min pointer, **Extra: or update**
- 4:  $transfer(\langle d_0, d_1, \dots, d_{\ell-1} \rangle, j)$ , if necessary
- 5: reduce  $b_{j+1}$  by 1 **(3)**
- 6: **else**
- 7: increase  $d_0$  by 1 **(1)**

**(3)** Advance *siftdown* one level downwards (or do nothing if

# Proof of correctness

---

**Case 1 of 12:**  $d_j$  is the first digit that equals 2, is not a wall, precedes any wall,  $j \neq 0$ ,  $d_{j+1} = 0$ , and  $d_{j+1}$  is critical. In such a case, a transfer is initiated at  $d_j$ . Before the operation,  $d_{j-1} = 0$ . After the operation,  $d_j = 0$  and  $d_{j+1}$  becomes a wall with  $b_{j+1} = j$ . At this point,  $d_{j-1}$  is the critical 0 for the wall  $d_{j+1}$ . . . .

**Case 2 of 12:**  $d_j$  is the first digit that equals 2, is not a wall, precedes any wall,  $j \neq 0$ ,  $d_{j+1} = 0$ , and  $d_{j+1}$  is not critical. . . .

**Case 3 of 12:**  $d_j$  is the first digit that equals 2, is not a wall, precedes any wall,  $j \neq 0$ , and  $d_{j+1} = 1$ . . . .

**Case 4 of 12:**  $d_0 = 2$ ,  $d_1 = 0$ , and  $d_1$  is critical. For such a case, the created wall  $d_1$  is immediately dismantled. . . .

**Case 5 of 12:** . . .

# Addition/meld

---

**Algorithm**  $addition(\langle d_0, \dots, d_{k-1} \rangle, \langle e_0, \dots, e_{\ell-1} \rangle)$

1: **assert**  $k \leq \ell$

2: **for**  $i \in \{0, 1, \dots, k-1\}$

3:     **repeat**  $d_i$  times

4:         **if**  $b_i > 0$

5:             reduce  $b_i$  to 0 **(1)**

6:             *arbitrary-increment*( $\langle e_0, \dots, e_{\ell-1} \rangle, i$ ) **(2)**

7: **return**  $\langle e_0, e_1, \dots, e_{\ell-1} \rangle$

**(1)** Finish *siftdown*, if any

**(2)** Add a heap to the collection

**(3)**  $O(k)$  digit changes

**(3)**  $O((\lg m)^2)$  worst-case time;

$m, n$ : #elements,  $m \leq n$

## Further reading

---

Elmasry, Jensen, and Katajainen, Two skew number systems and one application, *Theory of Computing Systems*, (invited)

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)