Number systems and de-amortization [Knuth 1970s]

→ make guarantees worst-case instead of amortized

binary counter

b_k	...	b_2	b_1	b_0 (LSB)
1	...	0	1	0
1	...	0	1	1
1	...	1	0	0

increment → constant amortized time

goal: make increment constant actual ~~time~~

problem:

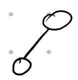
00 1 1 1 1 1 1 1 1
01 0 0 0 0 0 0 0 0

Example application to data structures

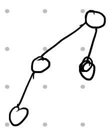
binomial tree of rank k

B_0 (rank 0) :

B_1 :



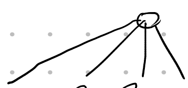
B_2 :



$|B_k| = 2^k$

...

B_k :



$B_{k-1} \cdots B_2 \; B_1 \; B_0$

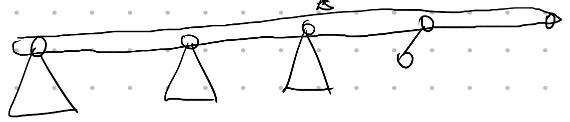
$\rightsquigarrow \; |B_k| = 1 + \sum_{i=0}^{k-1} |B_i|$

Binomial heap

collection of binomial trees, each tree has unique rank

e.g.

B_5 $\quad B_3$ $\quad B_2$ min B_1 $\quad B_0$



- each node stores a key
- trees are in heap-order

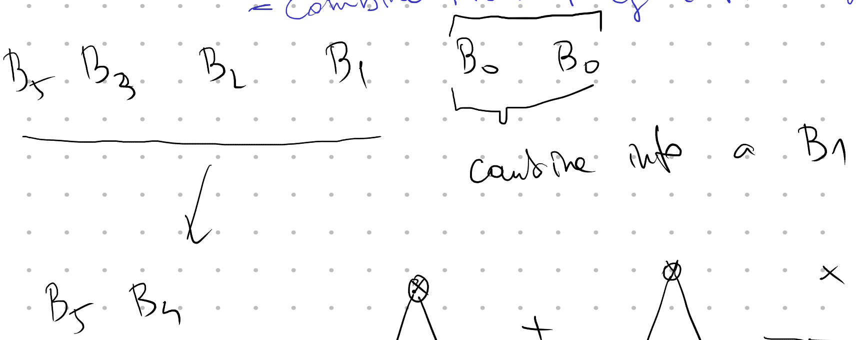
size $|B_5| = 2^5$ $\quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$

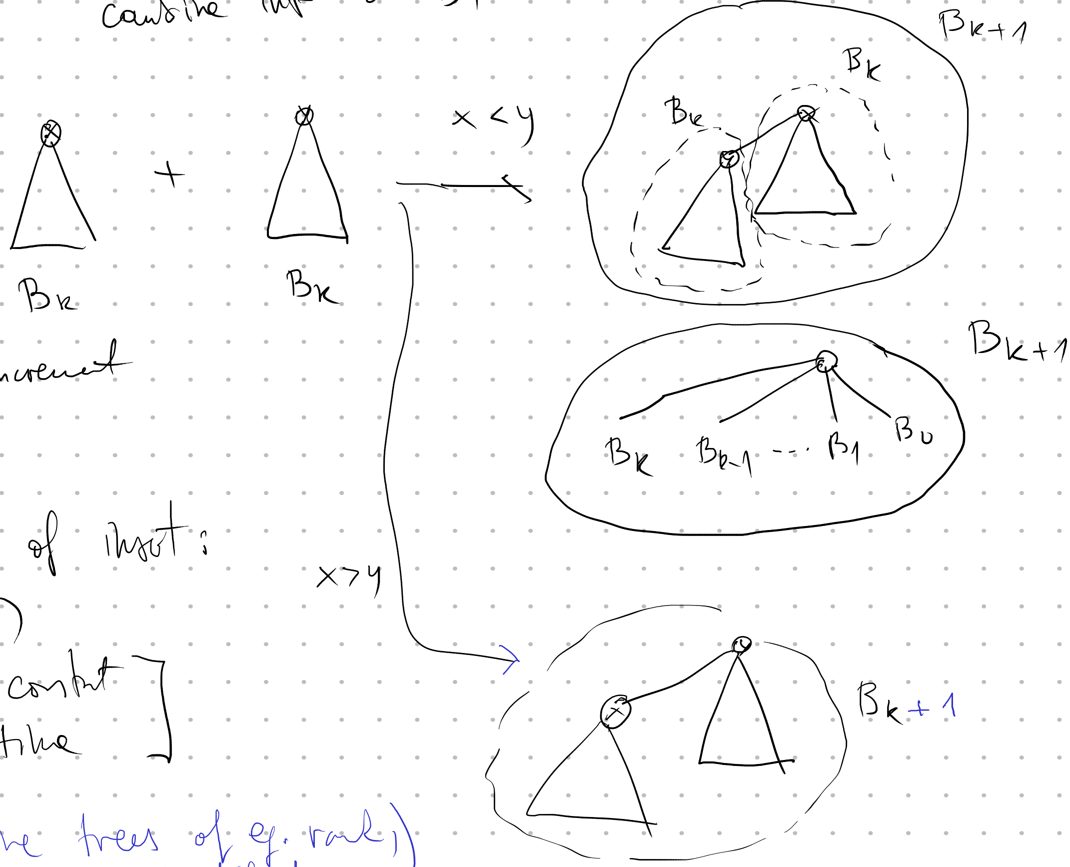
elements in binomial heap $= n$

→ view as representation of n in binary

5	4	3	2	1	0	
1	0	1	1	1	1	$= (n)_{(2)}$

- find min ✓ $O(1)$ time
- insert(x): ✓
  - create a new tree B_0 x
  - add this tree to collection of trees
  - combine trees of equal rank until done





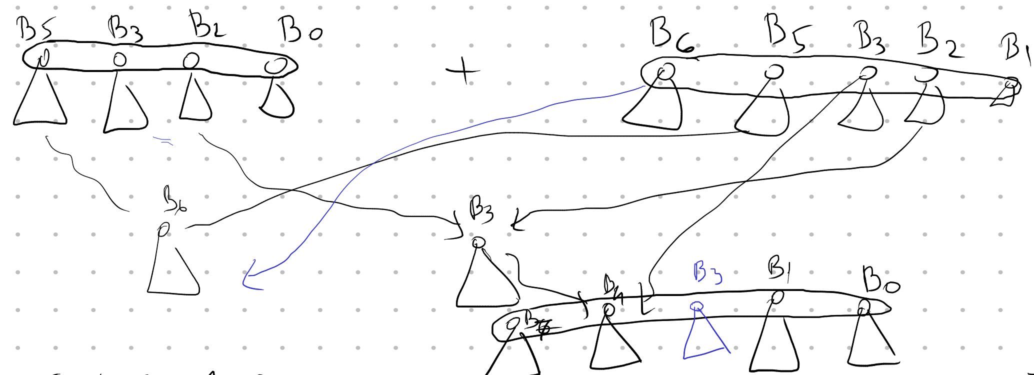
€ € € € €
1 0 1 1 1 1
increment
€ €
1 1 0 0 0 0

amortized cost of insert:

$O(1)$

[would like actual constant time]

- merge ✓ (combine trees of eq. rank until done)



$\overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{0}\ \overset{0}{1} + \overset{6}{1}\ \overset{5}{1}\ \overset{4}{0}\ \overset{3}{1}\ \overset{2}{1}\ \overset{1}{1}\ \overset{0}{0} \longrightarrow \overset{7}{1}\ \overset{6}{0}\ \overset{5}{0}\ \overset{4}{1}\ \overset{3}{1}\ \overset{2}{0}\ \overset{1}{1}\ \overset{0}{1}$

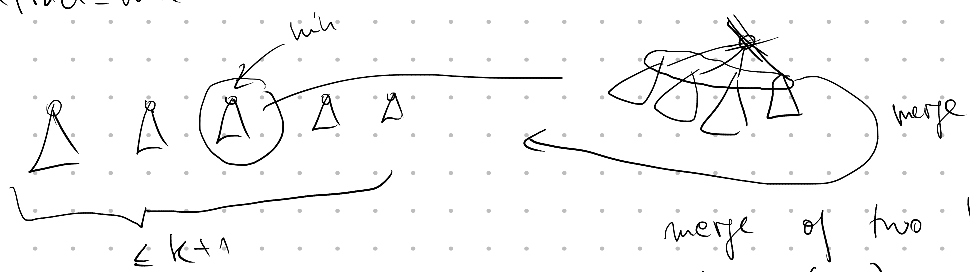
time: $O(k) = O(\log n)$

$|B_k| = 2^k$

$|B_0| + |B_1| + \dots + |B_k| = 2^{k+1} - 1 \geq n \geq 2^k$ — k here is rank of largest tree

$k = O(\log n)$

- extract_min



- cut out root with min key
- merge children-list with rest of heap

merge of two lists of binomial trees

$O(k) = O(\log n)$

- find-min $O(1)$
- insert $O(1)$ amortized
- merge $O(\log n)$
- extract-min $O(\log n)$

de-amortize?
actual $O(1)$ time

Binary counter

$b_k \; \ldots \; b_2 \; b_1 \; b_0$
$1 \qquad\quad 0 \;\; 1 \;\; 1$

$n = \text{value} = \sum_{i=0}^{k} b_i \cdot 2^i \le 2^{k+1} - 1$

increment → insert $O(1)$ amortized } time

add → merge $O(\log n)$ actual

$\ldots 0\,1111111$
$\ldots 1\,0000000$

<u>Idea</u>: Use digits $\{0,1,2\}$

$0\,1111111$ +1
$0\,1111112$ +1
$0\,1111121$ +1
…
02222222 +1 → just a diff. representation for 11111110
11111111 → still need to flip many digits

$t_k \; \ldots \; t_2 \; t_1 \; t_0$

$n = \sum_{i=0}^{k} t_i \, 2^i \le \sum 2^{i+1} \le 2^{k+2} - 1$

$k = O(\log n)$

<u>Idea</u>: keep 2s separated (cannot be neighbors)

Rules for numbering system

$(t_k \; t_{k-1} \; \dots \; t_2 \; t_1 \; t_0)$

(R_1) Digits $\{0,1,2\}$

(R_2) Between every two 2's there is at least one 0
(and arbitrary # of 1's)

(R_3) $t_0 \neq 2$ (last digit not 2)

e.g. $t_6 \; t_5 \; t_4 \; t_3 \; t_2 \; t_1 \; t_0$
$2 \; 0 \; 1 \; 0 \; 1 \; 2 \; 0$

Numbering system is redundant

e.g. $t_2 \; t_1 \; t_0$
$1 \; 2 \; 0 = 0\cdot 2^0 + 2\cdot 2^1 + 1\cdot 2^2 = 8$

$t_2 \; t_1 \; t_0$
$1 \; 1 \; 2 = 2\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 = 8$

How to increment counter?

increment

$t_0 \leftarrow t_0 + 1$
if $(t_0 = 2)$
  $t_0 = 0$
  Fix

$O(1)$ worst-case time

$t_6 \; t_5 \; t_4 \; t_3 \; t_2 \; t_1 \; t_0$
$\dots \boxed{2} \; 0 \; 1 \; 0 \; 1 \; \boxed{2} \; 0$
$\dots 2 \; 0 \; 1 \; 0 \; 1 \; 2 \; 1$

1
6
...
Stack S

Fix — Store indices of all $\boxed{2}$-digits

if $|S| = 0$ (no 2-digit yet, just increment t_0)
  $t_1 \leftarrow t_1 + 1$
  if $(t_1 = 2)$ push$(1, S)$

$t_1 \; t_0$
$\dots \; x \; 1$
$(x+1) \; 0 \; 0$

value = value $-1+2$ = value $+1$

$\dots \boxed{2} \dots \boxed{2} \dots 1$
0

else
  if $(t_1 = 0)$ (easy case, no 2-digit created)
    $t_1 \leftarrow 1$

$\dots \boxed{2} \dots \boxed{2} \dots 0 \; 1$
$\dots 1 \; 0$

  elsif $(t_1 = 1)$ (created new 2-digit, must fix previous)
    $t_1 \leftarrow 2$
    last $\leftarrow$ pop(S)
    $t_{last} \leftarrow 0$
    $t_{last+1} \leftarrow t_{last+1} + 1$
    if $(t_{last+1} = 2)$ push$(last+1, S)$
    push$(1, S)$

$(x+1) \; 0$ last

$\dots \boxed{2} \dots 0 \dots x \; \boxed{2} \dots 1 \; 1$
$\boxed{2} \; 0$

last+1 — at least one 0

- value unchanged
- x either 0, 1 — R_1 ✓, R_3 ✓

$x=0$ ✓
$x=1$ ✓ — R_2 ✓

elseif $(t_1 = 2)$ → 1 will be popped

pop(S)

$t_1 \leftarrow 1$

$t_2 \leftarrow t_2 + 1$

if $(t_2 = 2)$ push$(2, S)$

$t_2\ t_1\ t_0$

$\boxed{2} \ldots ? \ldots x\ \boxed{2}\ 1$

$\boxed{2}\ 0\ \boxed{2}(x+1)\ 1\ 0$

decrease by 3

$+4$

$+1$

R_1 ✓

$X = 0$

$X = 1$

R_2 ✓

---

after fixing, stack S again correct, storing locations of $\boxed{2}$ - digits

$t_{k+1}\ t_k$

$x\ 2$

$(x+1)\ 0$

value unchanged

$+2^{k+1} - 2 \cdot 2^k = 0$

Back to binomial heaps

e.g.

$B_5 \quad B_3 \quad B_2 \quad B_0$

e.g.

$B_5 \quad B_5 \quad B_3 \quad B_2 \quad B_1 \quad B_0 \quad B_0$

t_5	t_4	t_3	t_2	t_1	t_0
2	0	1	1	~~0~~ 1	~~2~~ 0

New version:

Each val appears at most ~~once~~ twice

- Rules 1–3 have natural interpretation
- Data structure insert can follow increment/Fix from before:

$t_{k+1}\ t_k$

$x\ 2$

$(x+1)\ 0$

combining two B_k trees

Result

$O(1)$ actual time insert

$O(\lg n)$ time extract-min?

merge?

We need to add two counters, such that Rules 1–3 are true for the result.

exercise: Show how to do add

→ 0/1/2 Number system vs binomial heap

a showcase of de-amortization.

Technique / ideas more generally applicable.