# $\epsilon$-Approximate Selection

Based on

Approximate Medians and other Quantiles in One Pass and with Limited Memory

by G. Manku, S. Rajagopalan, and B. Lindsay

Proc. 1998 ACM SIGMOD, Vol 27, No 2.

February 11, 2004

# Background: Median

Given $(a_1, \ldots, a_N)$ from an ordered set:

## Median Problem:

- Find element that evenly splits elements.
- Useful statistic.
- Applies to non-numerical data.

## Current activity:

- Focused on minimal number of comparisons.

# Generalization of Median

Two forms of the same problem:

1. Quantile Problem:

   Find element at $\lceil \phi N \rceil$.

2. Selection Problem:

   Find element at location $k \leq N$.

Used for:

- Order Statistics.
- Equal depth histograms.
- Other?

Solutions:

- $O(N)$ probabilistic: (Modified quicksort). By Tony Hoare, 1961
- $O(N)$ deterministic. By Blum, Floyd, Pratt, Rivest, and Tarjan, 1973

# <u>The Problem</u>

**Def. 1 ( $\epsilon$-Approximate Selection )**

*Given* $P \leq N$*,*

*Return:* $V$ *within* $\lceil \epsilon N \rceil$ *in the sorted order.*

**One-Pass $\epsilon$-Approximate Selection Problem**

We are given

- $\epsilon$, the approximation accuracy,
- $N$, the size of the data sequence,
- $S := (a_1, a_2, \ldots, a_N)$.

Allowed one pass over the data

to form a summary data object,

for $\epsilon$-approximating selection queries.

The goal is use a "small amount of memory" while forming this summary data structure.

# A Solution?

Can solve problem exactly

- using $O(N)$ space,
- in $N \ln(N)$ time,
- for $O(1)$ queries.

Just

sorting the data,

placing the results in an array.

For $\epsilon$-approximation:

Select evenly spaced elements

separated by $\epsilon N$

yields summary data object of size $1/\epsilon$

# A Solution

Problem: $N \gg$ available memory

**Goal:** We will cover a method that yields an $\epsilon$-approximation using $O(\epsilon^{-1}[\ln(\epsilon N)]^2)$ space.

# Construction Operators

Build summary structure on incoming data.

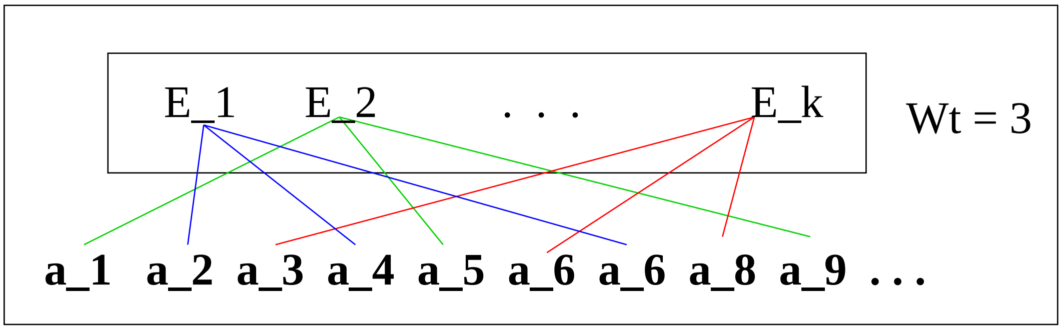
They manage

**Buffers:** $B$ buffers, each of size $K$.

Active buffer $Y$ contains $K$ elements.

Each element in $Y$ "summarizes" or

represent $Y$.wt elements from sequence.




**Toggle:** This has a technical purpose.

# $\underline{\texttt{New()}}$

- fills empty buffer $Y$ from sequence.
- sorts the elements,
- assign $Y$ a weight of one: $Y$.wt $:= 1$,
- sets $Y$.offset $:= 1$,
- $Y$.used $:=$ $\texttt{True}$.

$$
\begin{array}{cccccccccc}
 & a & q & c & w & n & p & b & s & \ldots \\
 & & & \downarrow & & & & & & \\
| & a & q & c & w & n & | & & & \\
 & & & \downarrow & & & & & & \\
| & a & c & n & q & w & | & & &
\end{array}
$$

If necessary pad $N$ to multiple of $K$, with

$-\infty$ in the first buffer.

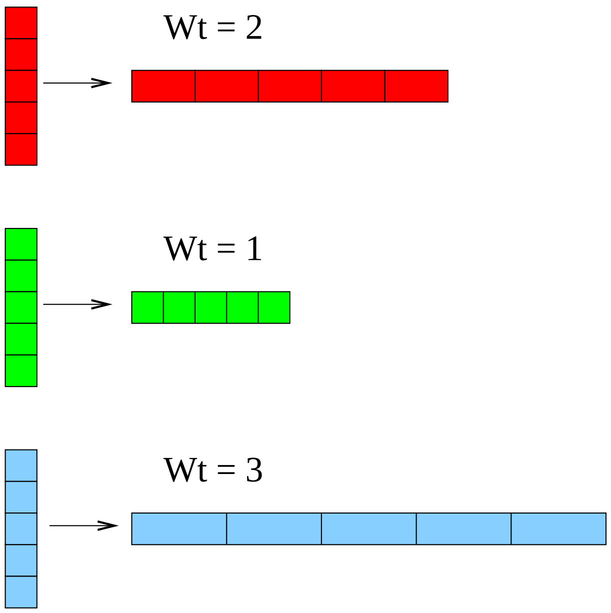
$+\infty$ in the last buffer

# Collapse()-1

$Y_1, \ldots, Y_k$, $k \geq 2$ full buffers $\rightarrow$ empty $Y_0$:

Virtually:

**1.** Replicate elements of $Y_i$ by $Y_i$.wt,



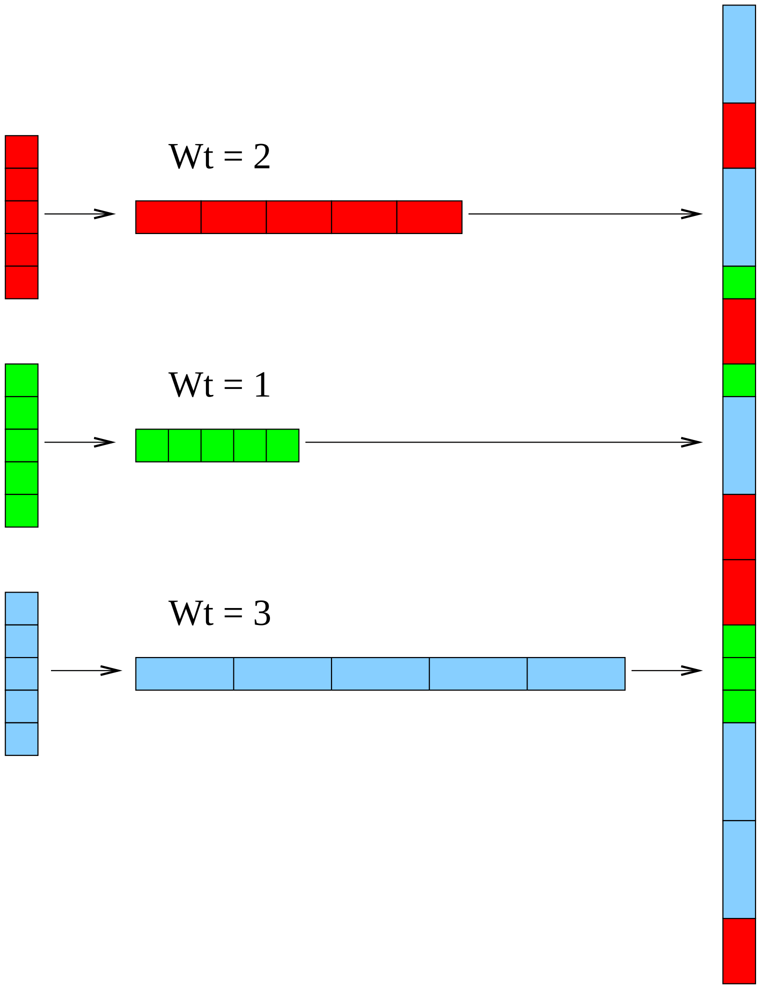
# Collapse()-2

$Y_1, \ldots, Y_k$, $k \geq 2$ full buffers $\rightarrow$ empty $Y_0$:

Virtually:

**1.** Replicate elements of $Y_i$ by $Y_i$.wt,

**2.** Merge sorted buffers.

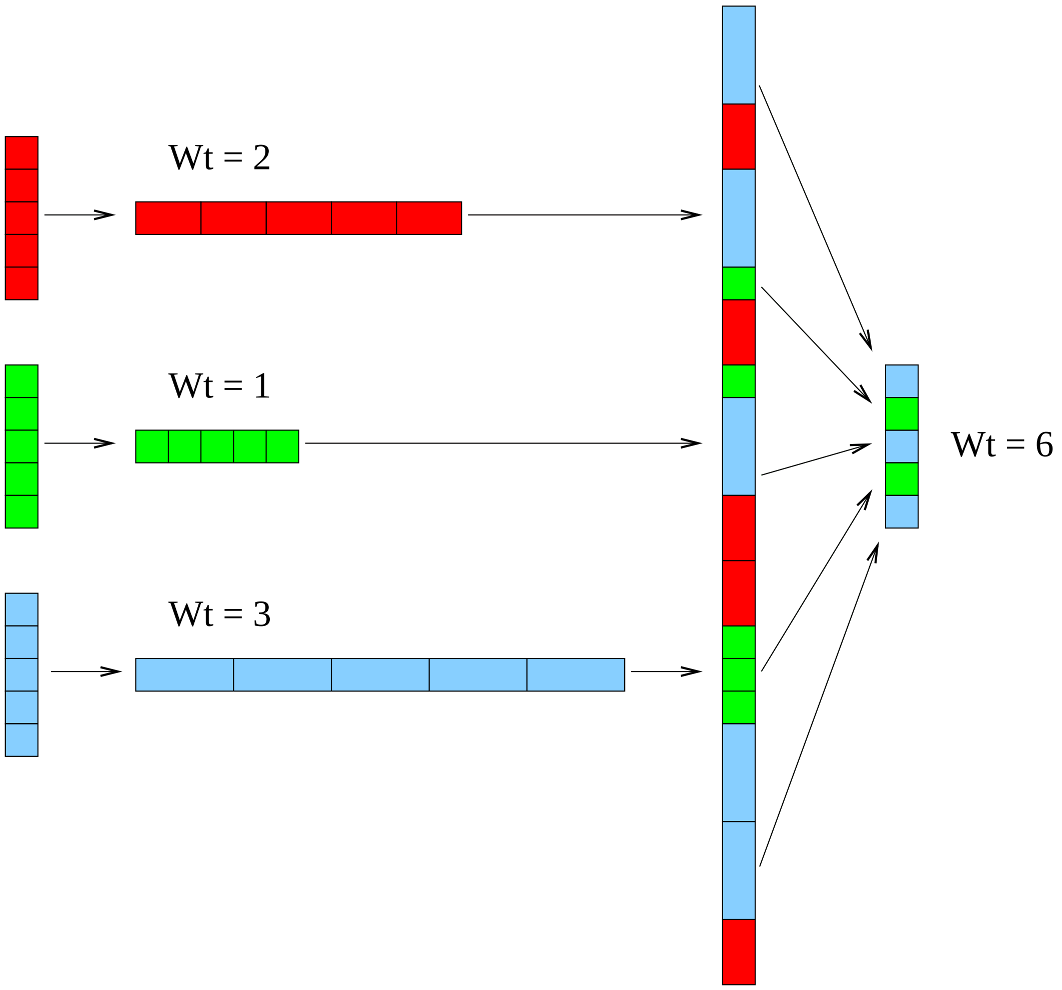


# Collapse()-3

$Y_1, \ldots, Y_k$, $k \geq 2$ full buffers $\rightarrow$ empty $Y_0$:

Virtually:

1. Replicate elements of $Y_i$ by $Y_i$.wt,
2. Merge sorted buffers.
3. Select $K$ evenly spaced elements



# Collapse()-Weighted Merge

1. $Y_0.\mathsf{wt} := Y_1.\mathsf{wt} + \cdots + Y_k.\mathsf{wt}$.

2. $Y_0.\mathsf{offset} :=$

$$\begin{cases} \dfrac{Y_0.\mathsf{wt}+1}{2} & \text{if } Y_0.\mathsf{wt} \text{ is odd,} \\ \dfrac{Y_0.\mathsf{wt}}{2} + \texttt{Toggle} & \text{if } Y_0.\mathsf{wt} \text{ is even.} \\ & \texttt{Toggle := 1 - Toggle}. \end{cases}$$

3. Select $K$ elements located:
   $Y_0.\mathsf{offset} + jY_0.\mathsf{wt}$, for $j = 0, \ldots, (K-1)$.

**Initialize:**

priority queue with
$(Y_1[1], 1), \ldots, (Y_k[1], k)$;

virtual_element_count := 0;

**While** priority queue not empty:

Dequeue least element $(e, i)$;

Enqueue next from $Y_i$;

virtual_element_count $+=$ $Y_i.\mathsf{wt}$;

If passes selection point,
  add $e$ to $Y_0$;

# `Collapse()`

$Y_1, \ldots, Y_k$, $k \geq 2$ full buffers $\rightarrow$ empty $Y_0$:

Last, set the status of the buffers:

**4.** $Y_0$.used := `True`.

**5.** Mark $Y_1$.used$, \ldots, Y_k$.used to `False`.

Note, could collapse into one of the old buffers.

# Completion/Query Operators

Finalize(): Convert buffers to query format.

Merge sort active buffers.

Convert ordered sequence of `(value, weight)`,

to ordered array of `(value, offset)`.

$$(a, 8), (b, 2), (c, 7), (d, 2), (e, 8), \cdots$$

$$\downarrow$$

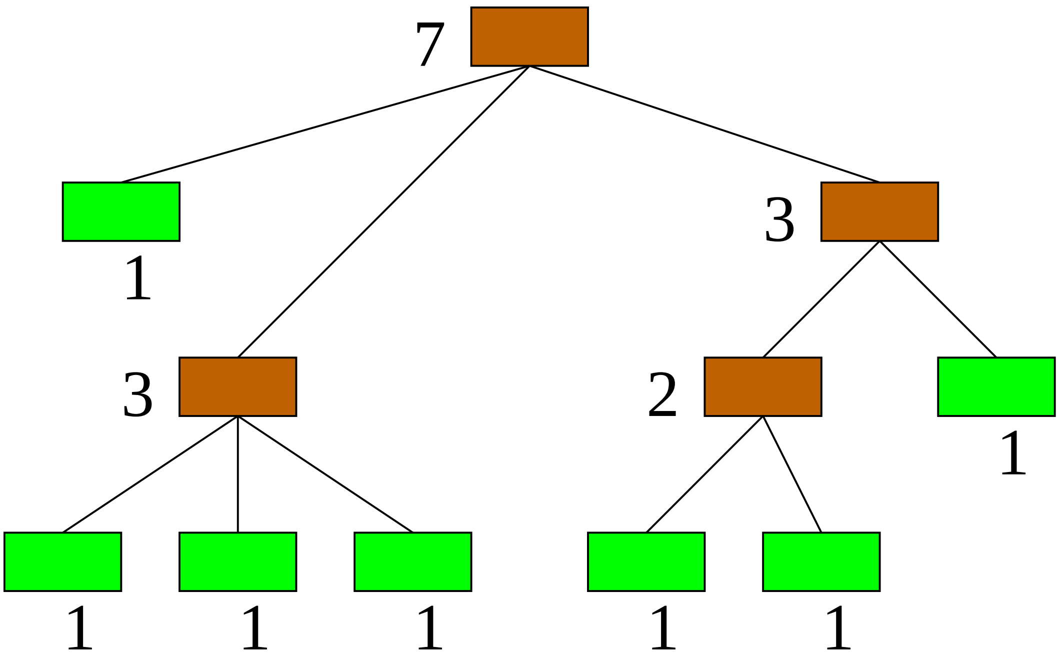
$$(a, 8), (b, 10), (c, 17), (d, 19), (e, 27), \cdots$$

Select(): Given $P \leq N$,
use bisection to seek candidate $V$.

# History Forest

- Various algorithms built from operators.
- Differ only in use of New() and Collapse().
- Policy reflected in the evolution of buffers.
- Can view buffer evolution as forest.
  - New() creates the leaves.
  - Collapse() produces internal nodes.

A Tree in a History Forest:



# History Forest-2

Note

- Roots are active buffers.
- Internal nodes have weight equal to number of leaf decedents.

**Lemma 2** *Mirror of a history tree is a history tree.*

**P:** Let $\{a_1, \ldots, a_N\} \in (O, \leq)$, we

- Invert order of sequence: $(i \mapsto N + 1 - i)$,
- Use the reverse order on $O$,
- Invert uses of $-\infty, +\infty$.
- Possibly start `toggle` with different value.

The $P$-th selection of the original corresponds to the $N+1-P$-th selection in the mirror. $\square$

# Some Analysis

Given $P \leq N$, Select() returns $V$.

How far is $V$ from $P$?

**Def. 3** *We define some variables:*

$P_V$ *Real location of $V$ in sorted sequence.*

$C$ *The number of Collapse() operations, or internal nodes in history forest.*

$W_{max}$ *Weight of heaviest root (active buffer).*

$W_{\Sigma}$ *Sum of the weights of internal nodes.*

# Some Analysis-2

Assume no repetitions in data (worst case).

We will show

$$P_V > P - \left(\frac{W_\Sigma - C - 1}{2} + W_{max}\right).$$

Mirror tree has same values for:
$C$, $W_{max}$, and $W_\Sigma$.

Symmetry will imply

$$P_V < P + \left(\frac{W_\Sigma - C - 1}{2} + W_{max}\right).$$

Establishing localization bound

$$|P_V - P| < \frac{W_\Sigma - C - 1}{2} + W_{max}.$$

# Special Elements

**Def. 4 ( Special Elements )**
*For any root $X$,*

$$X.\text{special} := \{\, v \in X \mid v < V \,\}.$$

$\equiv$ *elements in final array less than $V$.*

*Once* $Y.\text{special}$ *determined, let*
$Y.\text{max\_special} :=$

$$\begin{cases} \max Y.\text{special} & \textit{if } Y.\text{special} \neq \emptyset, \\ -\infty & \textit{otherwise.} \end{cases}$$

*If $Y$ has children, proceed recursively*

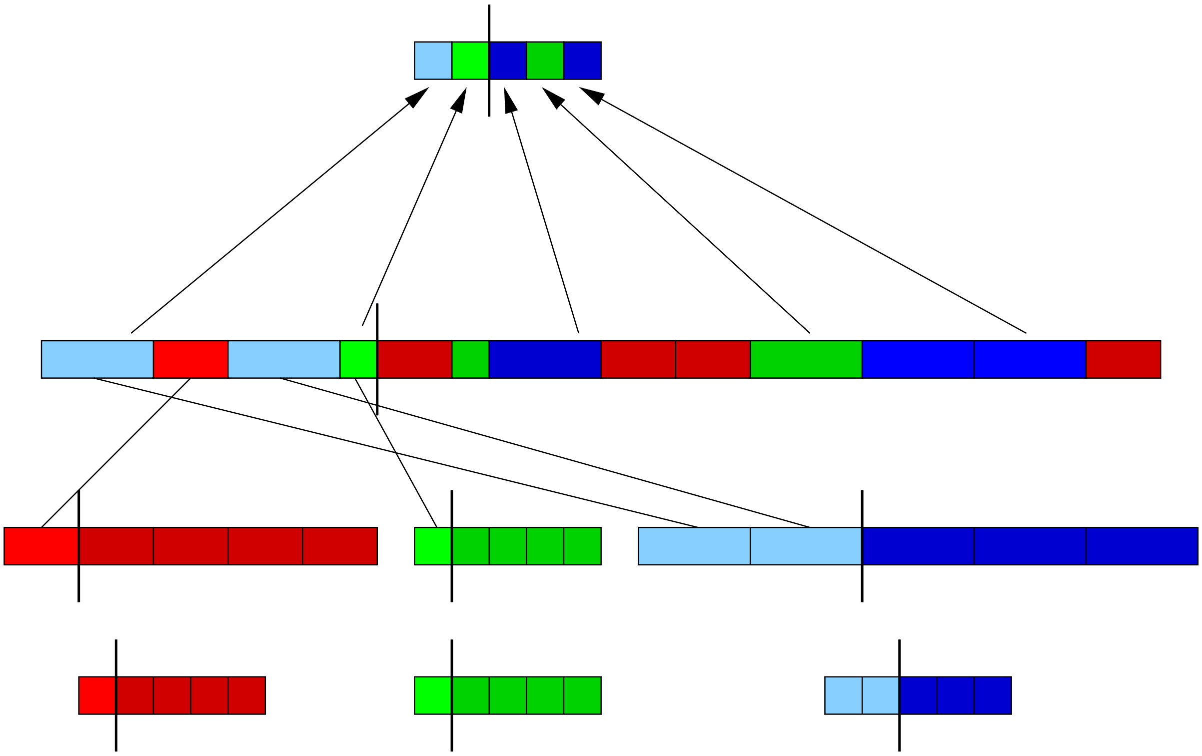
*For* $Z \in Y.\text{children}$

$$Z.\text{special} := \{\, v \in Z \mid v \leq Y.\text{max\_special} \,\}.$$

# Special Elements

For $Z \in Y$.children

$$Z.\text{special} := \{ v \in Z \mid v \leq Y.\text{max\_special} \}.$$



# Weighted Count of Specials

**Def. 5** *We set*

$$Y.\text{wt\_special} := Y.\text{wt} * |Y.\text{special}|$$

*weighted count of special elements in $Y$.*

*Let*

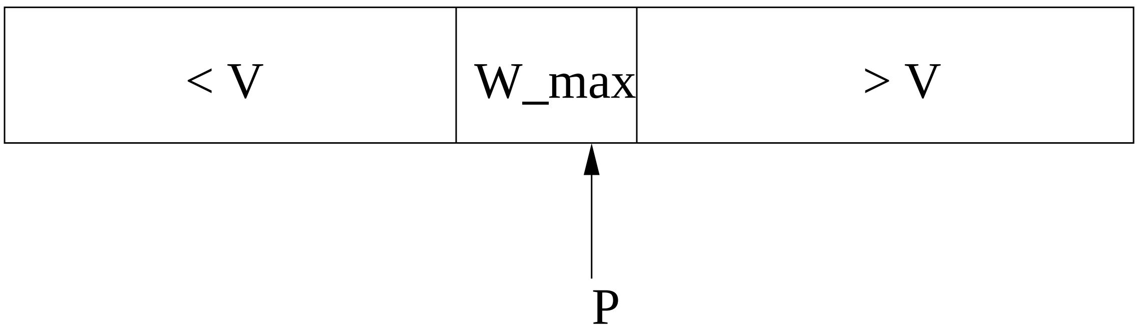
$$Roots.\text{wt\_special} := \sum_{X \textit{ is a root}} X.\text{wt\_special}$$

*weighted count of special elements in roots.*

**Lemma 6**

$$P \leq Roots.\text{wt\_special} + W_{max}.$$

**P:** In final array, element in position $P$ has multiplicity of 1 to $W_{max}$.




□

# Weighted Count under Recursion

**Lemma 7** *If $Y$ has children, then*

$$\sum_{Z \in Y.\text{children}} Z.\text{wt\_special} \geq Y.\text{wt\_special} - (Y.\text{wt} - Y.\text{offset}).$$

**P:** **Case $Y$.special $= \emptyset$:** LHS $\geq 0 \geq$ RHS.

**Case $Y$.special $\neq \emptyset$:**

- $Y$.wt_special := $|Y.\text{special}| * Y.\text{wt}$.
- $Y$.max_special located at $|Y.\text{special}|$.
- Picked from location

$$(|Y.\text{special}| - 1) * Y.\text{wt} + Y.\text{offset}$$

$$= |Y.\text{special}| * Y.\text{wt} - (Y.\text{wt} - Y.\text{offset})$$

$$= Y.\text{wt\_special} - (Y.\text{wt} - Y.\text{offset}).$$

These are exactly the elements in the children that will get distinguished as special. So we have equality. $\square$

# Accumulated Effects

Proceeding from parent to children, reduces weight count of special elements by at most

$$Y.\mathrm{wt} - Y.\mathrm{offset}.$$

So reduction from roots to leaves: bounded by

$$\sum_{Y \in \mathrm{internal}} (Y.\mathrm{wt} - Y.\mathrm{offset})$$

$$= W_\Sigma - \sum_{Y \in \mathrm{internal}} Y.\mathrm{offset}.$$

As the weight of the leaf nodes is one, we have shown:

**Lemma 8** *Let* $Leaves.\mathrm{special}$ *be the number of special elements in the leaves. Then*

$$\begin{aligned} & Leaves.\mathrm{special} \\ \geq\ & Roots.\mathrm{wt\_special} - W_\Sigma + \sum_{Y \in \textit{internal}} Y.\mathrm{offset}\,. \end{aligned}$$

$\square$

# Accumulation of Offsets

**Lemma 9** *The sum of the offsets of the collapse nodes is at least*

$$\frac{W_\Sigma + C - 1}{2}.$$

**P:** Let

$C_o$: number of internal nodes with odd weight,

$C_e^0$: number of nodes of even weight,

$C_e^0$ even weight with $Y$.offset $= Y$.wt$/2$,

$C_e^1$ even weight with $Y$.offset $= 1 + Y$.wt$/2$.

Then $C = C_o + C_e = C_o + C_e^0 + C_e^1$.

The sum of all the offsets is

$$\begin{aligned}\frac{W_\Sigma + C_o}{2} + C_e^1 &= \frac{W_\Sigma + C_o + 2C_e^1}{2} \\ &= \frac{W_\Sigma + C_o + C_e - \mathrm{mod}_2(C_e)}{2} \\ &= \frac{W_\Sigma + C - \mathrm{mod}_2(C_e)}{2}.\end{aligned}$$

□

# Lower Bound on $P_V$

So

$$
\begin{aligned}
& P_V \\
> \; & Leaves.\text{special} \\
\geq \; & Roots.\text{wt\_special} - W_\Sigma + \frac{W_\Sigma + C - 1}{2} \\
= \; & Roots.\text{wt\_special} - \frac{W_\Sigma - C + 1}{2}.
\end{aligned}
$$

**Lemma 10** *Let $P_V$ be the position of $V$ in the sorted list. Then*

$$P_V > P - \left(\frac{W_\Sigma - C + 1}{2} + W_{max}\right).$$

**P:** We have determined that

- $Roots.\text{wt\_special} - \dfrac{W_\Sigma - C + 1}{2} < P_V.$
- $P \leq Roots.\text{wt\_special} + W_{max}$,

  or equivalently
  $$P - \left(\frac{W_\Sigma - C + 1}{2} + W_{max}\right) \leq Roots.\text{wt\_special} - \frac{W_\Sigma - C + 1}{2}.$$

Transitivity now show the result. $\square$

# Uniformity of Variance

**Prop 11 ( Uniform Bound on Variance )**
*No element in the final array is more than*

$$\frac{W_{\Sigma} - C + 1}{2} + W_{max}$$

*from closest location in sorted input.*

**P:** The $P$-th element was an arbitrary.
So each element in the expanded list,
it isn't "too far away" from it's real location.

$\square$

# Basic Constraints

**Lemma 12** *The maximum number of children of any node is bounded by $B$, the number of buffers used.* $\square$

**Goal 13** We want to minimize $BK$ subject to two feasibility constraints:

1. Responses to queries are $\epsilon$-approximations:
$$\frac{W_\Sigma - C - 1}{2} + W_{max} \leq \epsilon N.$$
Depends only on forest shape.

2. History forest holds input of size $N$:
$$KL \geq N,$$
where $L$ is the number of leaf nodes.
Pick smallest such $K$.

# An Algorithm

Associate attribute of level to buffers by:

**2 or more unused buffer:**

- Fill an empty buffer
- place as an isolated node at level 0.

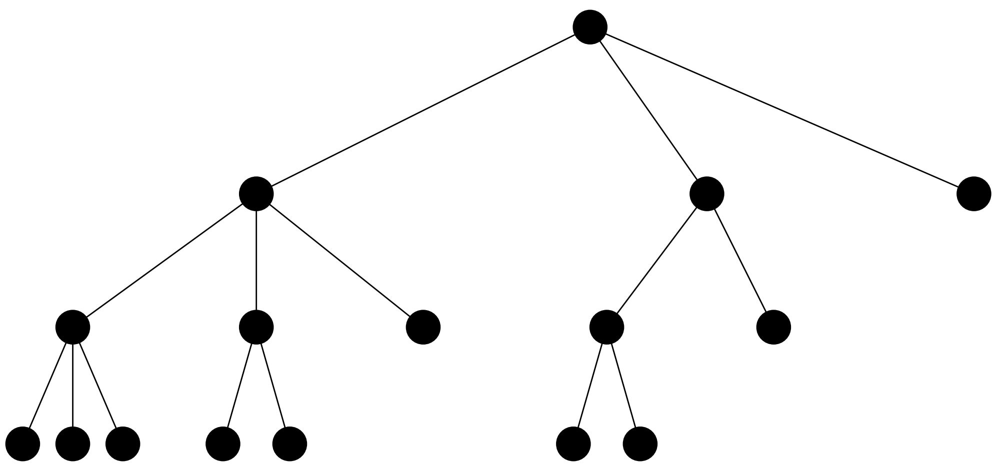
**1 unused buffer:**

- Fill the empty buffer,
- place as an isolated node at the level of the current lowest level buffer.

**0 unused buffer:**

- Collapse buffers at lowest level to buffer,
- assign at level one higher.

Example history tree for $B = 3$, $H = 3$:



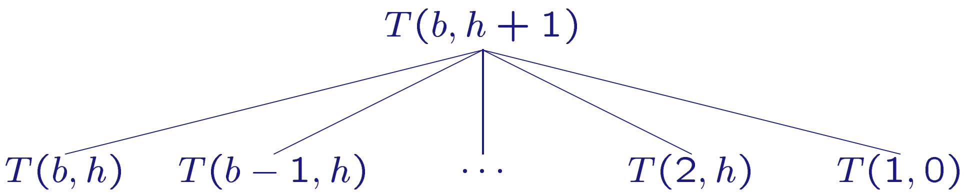
# Fractal Nature

$T(b, h)$: tree on $b$ buffers and height of $h$.

We examine individual trees in the forest:

Note

- Recursive structure when $b \geq 2$ and $h \geq 1$:




- for $b > 2$ and $h > 0$,
  if we remove first (largest) child from $T(b, h)$,
  we get $T(b-1, h)$.

- for $b > 2$ and $h > 1$,
  if we remove the bottom level of $T(b, h)$
  we get $T(b, h-1)$.

# Stats for Single Tree

**Def. 14** *Let $T(b,h)$ be the tree on $b$ buffers of height $h$. Then*

*$I(b,h)$ is the number of internal nodes.*

*$L(b,h)$ denotes the number of leaves.*

*$W_\Sigma(b,h)$ denote the sum of the weights of all the internal nodes.*

**Prop 15** *For $b \geq 2$ and $h \geq 1$,*

$$I(b,h) = \binom{b+h-2}{h-1}.$$

$$L(b,h) = \binom{b+h-1}{h}.$$

$$W_\Sigma(b,h) = h\binom{b+h-1}{h} - \binom{b+h-2}{h-2}.$$

**P:** Repeated applications of the upper summation formula.

See lemmas in appendix for the gory deals... $\square$

# Memory Size Bound

Best foot forward:
Examine constraints just before collapse into a single node: pre-$T(B,H)$

$$\epsilon N \geq \frac{W_\Sigma - C - 1}{2} + W_{max} \approx \frac{W_\Sigma}{2}$$

$$\approx \frac{1}{2} H \binom{B+H-1}{H}.$$

From $KL \geq N$ we get

$$K\,L(B,H) \geq N \quad \equiv \quad K\binom{B+H-1}{H} \geq N.$$

To establish upper bound memory usage, set $H = B$ and use equalities.

From $\epsilon N \approx \frac{1}{2} B \binom{2B-1}{B}$, we get $B \in O(\ln(\epsilon N))$.

From $N \approx K\binom{2B-1}{B} \approx K\frac{2\epsilon N}{B}$, or $K \approx \frac{B}{2\epsilon}$, we get $K \in O(\epsilon^{-1} \ln(\epsilon N))$.

Hence $KB \in O(\epsilon^{-1}[\ln(\epsilon N)]^2)$.

# Summary

Algorithm has upper bound on

Space usage:

$$KB \in O(\epsilon^{-1}[\ln(\epsilon N)]^2).$$

Query response time.

$$\ln(KB) = O(\ln(\epsilon^{-1}[\ln(\epsilon N)]^2)).$$

Further developments:

Component of probabilistic version.

Component of an online version.

# Appendix

# Counting Internal Nodes

**Lemma 16** *For $b \geq 2$ and $h \geq 1$,*

$$I(b,h) = \binom{b+h-2}{h-1}.$$

**P:** We show by induction on the height. Clear for $I(b,1) = 1 = \binom{b+1-2}{0}$.

Suppose it holds for $h \geq 1$, then

$$\begin{aligned} I(b,h+1) &= \sum_{k=2}^{b} I(k,h) + 1 \\ &= \sum_{k=2}^{b} \binom{k+h-2}{h-1} + 1 \\ &= \sum_{k=0}^{b-1} \binom{(k+1)+h-2}{h-1} \\ &= \binom{b+(h+1)-2}{(h+1)-1}. \end{aligned}$$

$\square$

# Counting Leaf Nodes

**Lemma 17** *For $b \geq 2$ and $h \geq 1$,*

$$L(b,h) = \binom{b+h-1}{h}.$$

**P:** We show by induction on the height. This is clear for $L(b,1) = b = \binom{b+1-1}{1}$.

Suppose it holds for $h \geq 1$, then

$$\begin{aligned} L(b,h+1) &= \sum_{k=2}^{b} L(k,h) + 1 \\ &= \sum_{k=2}^{b} \binom{k+h-1}{h} + 1 \\ &= \sum_{k=0}^{b-1} \binom{k+h}{h} \\ &= \binom{b+h}{h+1} \\ &= \binom{b+(h+1)-1}{h+1}. \end{aligned}$$

$\square$

# $W_{max}$

**Cor 18** *For $h \geq 2$, the weight of the maximum child in $T(b,h)$ is*

$$W_{max} = T(b, h-1) = \binom{b+h-2}{h-1}.$$

**P:** The biggest child of $T(b,h)$ is $T(b,h-1)$, so the number of leaves is

$$\binom{b+(h-1)-1}{h-1} = \binom{b+h-2}{h-1}.$$

$\square$

# Weighing All Internal Nodes

**Lemma 19** *For $b \geq 2$ and $h \geq 1$,*

$$W_\Sigma(b,h) = h\binom{b+h-1}{h} - \binom{b+h-2}{h-2}.$$

**P:** We show by induction on the height. This is clear for $W_\Sigma(b,1) = b = 1\binom{b+1-1}{1} - 0$. Suppose it holds for $h \geq 1$, then

$$
\begin{aligned}
& W_\Sigma(b,h+1) \\
&= \sum_{k=2}^{b} W_\Sigma(k,h) + L(b,h+1) \\
&= \sum_{k=2}^{b} h\binom{k+h-1}{h} - \sum_{k=2}^{b}\binom{k+h-2}{h-2} + \binom{b+(h+1)-1}{h+1} \\
&= h\sum_{k=0}^{b-1}\binom{k+h}{h} - h - \sum_{k=2}^{b}\binom{k+h-2}{h-2} + \binom{b+h}{h+1} \\
&= (h+1)\binom{b+h}{h+1} - \sum_{k=2}^{b}\binom{k+h-2}{h-2} - (h-1) - 1 \\
&= (h+1)\binom{b+h}{h+1} - \sum_{k=0}^{b}\binom{k+h-2}{h-2} \\
&= (h+1)\binom{b+(h+1)-1}{h+1} - \binom{b+(h+1)-2}{(h+1)-2}.
\end{aligned}
$$

□

# Constraints

Best foot forward:

Examine constraints just before collapse into a single node: pre-$T(B,H)$

1.
$$\begin{aligned}\epsilon N &\geq \frac{W_\Sigma - C - 1}{2} + W_{max}\\ &= \frac{(W_\Sigma(B,H) - L(B,H)) - (I(B,H) - 1) - 1}{2} + L(B,H-1).\end{aligned}$$

Which becomes

$$\begin{aligned}2\epsilon N &\geq W_\Sigma(B,H) - L(B,H) - I(B,H) + 2L(B,H-1)\\ &= \left[H\binom{B+H-1}{H} - \binom{B+H-2}{H-2}\right]\\ &\quad -\binom{B+H-1}{H} - \binom{B+H-2}{H-1} + 2\binom{B+H-2}{H-1}\\ &= (H-1)\binom{B+H-1}{H} - \binom{B+H-2}{H-2} + \binom{B+H-2}{H-1}.\end{aligned}$$

This first constraint places restrictions on the nature of the history tree, and is independent of size of the buffers.

2. From $KL \geq N$ we get

$$K\,L(B,H) \geq N \quad \equiv \quad K\binom{B+H-1}{H} \geq N.$$

# Bound on $B$

To reduce the analysis complexity,
for upper bound on best selection for $B$
we make the assumption that $B = H$.
Substitute and reduce:

$$\begin{aligned}
2\epsilon N &\geq (B-1)\binom{2B-1}{B} - \binom{2B-2}{B-2} + \binom{2B-2}{B-1} \\
&= B\binom{2B-1}{B} - \binom{2B-1}{B} - \binom{2B-2}{B-2} + \binom{2B-2}{B-1} \\
&= B\binom{2B-1}{B} - \binom{2B-2}{B} - \binom{2B-2}{B-1} - \binom{2B-2}{B-2} + \binom{2B-2}{B-1} \\
&= B\binom{2B-1}{B} - \binom{2B-2}{B} - \binom{2B-2}{B-2} \\
&= B\binom{2B-1}{B} - 2\binom{2B-2}{B}.
\end{aligned}$$

If we select largest $B$ satisfying

$$2\epsilon N \geq B\binom{2B-1}{B},$$

then it satisfies the previous inequality.

FIXME:

For this selection of $B$ we have

$B \in O(\ln(\epsilon N))$.

# Bound on $K$

From $B = H$, our previous constraint reduces to

$$K\binom{2B-1}{B} \geq N,$$

for which we want the smallest such $K$.

The selection of $B$ satisfies

$$2\epsilon N = B\binom{2B-1}{B} \qquad \text{or} \qquad \frac{2\epsilon N}{B} = \binom{2B-1}{B},$$

Substitution yields for this choice of $B$

$$K\frac{2\epsilon N}{B} \geq N \qquad \text{or} \qquad K = \frac{B}{2\epsilon}$$

$$\text{which implies} \qquad K \in O(\epsilon^{-1}\ln(\epsilon N)).$$

Finally, we combine these two upper bounds in

$$\begin{aligned} KB &\in O(\epsilon^{-1}\ln(\epsilon N))O(\ln(\epsilon N)) \\ &= O(\epsilon^{-1}(\ln(\epsilon N))^2). \end{aligned}$$