1 Theoretical Analysis

Symbol	Definition
K	The set of possible key values
λ	The set of all GS
$ActualCount(k)$	k's occurrence count in the stream
$GS.maxLayer()$	index of GS last layer with an allocated counter
$GS.minLayer()$	index of GS first layer with an allocated counter
$GS.query_{l,r}(k)$	value of k's counter in GS's row r, layer l
$GS.query_r(k)$	GS's row r estimate of k's occurrence count

Table 1: Notations

For this section, we use the notations in Table 1. Note that $GS.query_r(k) = \sum_{l=GS.minLayer()}^{GS.maxLayer()} GS.query_{l,r}(k)$.

Lemma 1.1. *Compressing a sketch does not impact its query results. Formally:*

$$\forall GS \in \lambda, n \in \mathbb{N} : GS' := GS.compress(n) \Rightarrow$$
$$\forall k \in K : GS'.query(k) = GS.query(k)$$

Proof. By definition: $GS.query(k) = min\{GS.query_r(k) | 0 \leq r < D\}$. In GS', $0 \leq m \leq n$ counters are removed from the beginning of each row. If $m = 0$, $GS = GS'$ and we are trivially correct.

If $m = 1$, the first counters of each row, $\{C[r, O] | 0 \leq r < D\}$, were removed, so their value became 0. For every such $C[r, O]$, all of its children, $\{C[r, n] | OB \leq n < (O + 1)B\}$, must be allocated. Otherwise, we could not have removed $C[r, O]$. The value of each of the children was increased by $C[r, O]$'s pre-removal value. We define the functions $OV(C[r, n]), CV(C[r, n])$ that return the counter's values before and after the compression.

Without loss of generality, consider $0 \leq r < D$. If $GS.query_r(k)$ does not require summing $C[r, O]$, then we did not change any of the summed counters (since we only modified $C[r, O]$ and its children). Therefore: $GS'.query_r(k) = GS.query_r(k)$

Otherwise, $GS.query_r(k)$ includes summing $C[r, O]$. A single child of $C[r, O]$, marked C_{child}, would also be summed. We know this since $C[r, O]$ was compressed in GS', so its children must be allocated. Due to our implementation, k would be mapped to one of those children, whose value would be summed.

$$GS'.query_r(k) = \sum_{l=GS'.minLayer()}^{GS'.maxLayer()} GS'.query_{l,r}(k) =$$

$$CV(C[r,0]) + CV(C_{child}) + \sum_{l=GS'.minLayer()+2}^{GS'.maxLayer()} GS'.query_{l,r}(k) =$$

$$OV(C_{child}) + OV(C[r,0]) + \sum_{l=GS.minLayer()+2}^{GS.maxLayer()} GS.query_{l,r}(k) =$$

$$\sum_{l=GS.minLayer()}^{GS.maxLayer()} GS.query_{l,r}(k) = GS.query_r(k)$$

For $m > 1$, we use induction: Given that the assumption holds for $m-1$ compresses, we create $GS'' := GS.compress(m-1)$. Define $GS''' = GS''.compress(1)$, yielding a sketch that would return the same queries. □

Lemma 1.2. *If we expand a sketch, update it, and follow with undoing the expansion, the final state of our sketch is the same as if we updated it without expanding and undoing it.*

$$\begin{aligned}
&\forall GS \in \lambda, n, m \in \mathbb{N}: \\
&GS^* := GS.update(k_0, v_0).[...].update(k_m, v_m) \wedge \\
&GS_1 := GS.expand(n) \wedge \\
&GS_2 := GS_1.update(k_0, v_0).[...].update(k_m, v_m) \wedge \\
&GS_3 := GS_2.undoExpand(n) \Rightarrow GS_3 == GS^*
\end{aligned}$$

Proof. We start with some notations:

$U := \{k_i | 0 \leq i \leq m\}$: set of unique updated keys.

$UpdateSum(k) := \sum_{k_i = k} v_i$: a function mapping each $k \in U$ to the total amount it is updated by.

$Descendants(GS, C)$: a function mapping a counter in a GS to the counter's allocated descendants.

GS^* has the same structure as GS, by which we mean that the length of $Counters$ and the 4 integer fields O, B, D, W are the same for both. This is due to update's implementation.

GS_3 also has the same structure as GS. This is because the $Counters$ array in GS_1 has exactly n more counters than $Counters$ in GS, while in GS_3 the length of $Counters$ is reduced by exactly n counters from GS_1. The number of undone counters is not less than n, since for this to occur, one of the last n counters must not have an allocated parent. Yet, each of the last n counters has a parent counter present in GS, due to the implementation of expand.

Let m be an array index of an allocated counter in GS. We mark counters in array index m: C, C^*, C_1, C_2, C_3, matching the names of the respective sketches.

Denote $U_m = \{k | k \in U \wedge H^*_{r,l}(k) = m\}$. This is the set of updated keys mapped to the counter in the m^{th} array index of our sketches. From the implementation of update, we deduce:

$C^*.value = C.value + \sum\{UpdateSum(k) | k \in U_m\}$.

On the other hand:

$C_3.value = C_2.value + \sum\{C'.value | C' \in Descendants(C_2)\}$.

Since $Descendants(C_2)$ were allocated with value 0 in GS_1, each one's value is exactly the amount it was updated by $\{(k_i, v_i)\}$. Any key from U_m which did not update them has updated C_2. Therefore:

$C_2.value + \sum\{C'.value | C' \in Descendants(C_2)\} = C_1.value + \sum\{UpdateSum(k) | k \in U_m\}$.

Since expanding can not modify C's value, we get:

$C_3.value = C.value + \sum\{UpdateSum(k) | k \in U_m\} = C^*.value$

In conclusion, the structure of $GS^* and GS_3$ is the same. Further, their corresponding counters hold the same values. Therefore, they are identical. □

We proceed with proving useful properties of our error given a specific expansion strategy used in this paper. Note that any expansion strategy can be constructed to suit the properties required from a GS, and said properties can be proven using a method similar to the one presented below.

Lemma 1.3. *Initialize $GS \in \lambda$ with δ such that $D = \lceil ln(\frac{1}{\delta}) \rceil$, $W \in \mathbb{N}$. Additionally, select $M > 0$ - every time we increment GS's counters by a total of $u = \frac{M}{eD(B+1)}$, we expand GS by a single counter. After incrementing GS's counters by a total of N:*

1. *$ActualCount(k) \leq GS.estimate(k)$,*
2. *$Pr[GS.estimate(k) > ActualCount(k) + Mlog_B(N)] \leq \delta$.*

Proof. Consider the counter $C[r, l, H^*_{r,l}(k)]$ which is updated when we modify k's count, and $C[r, l, H^*_{r,l+1}(k)]$ is not yet allocated. Since every update adds a positive value, the counter holds at least the sum of increments for k after $C[r, l, H^*_{r,l}(k)]$ was allocated and before $C[r, l, H^*_{r,l+1}(k)]$ was allocated. At any specific time, $\sum_l C[r, l, H^*_{r,l}(k)] \geq ActualCount(k)$. Since our estimate is the minimum from this sum from each row, our estimate cannot be less than $ActualCount(k)$, thereby proving (1).

Onwards to (2): Since we expand by a single counter every u updates, every uD update all our rows are expanded by a single counter. Therefore, we fill the l^{th} layer $uDWB^l$ updates after allocating the last counter of the previous layer. We always update the lowest counter, therefore after layer $l+1$ is fully allocated layer l will receive no more updates.

Thus, for any specific layer it receives at most $uDWB^{l+1}$ updates while allocating the next layer, and $uDWB^l$ updates while allocating itself, for a total of $(B+1)uDWB^l$. We define indicator variables $I_{i,j,r,l}$, which are 1 only if $(i \neq j) \wedge H^*_{r,l}(i) = H^*_{r,l}(j)$, and 0 otherwise. Due to the pairwise independence of our hash functions, we get:

$E(I_{i,j,r,l}) = Pr[H^*_{r,l}(i) = H^*_{r,l}(j)] \leq \frac{1}{WB^l}$.

We define the variables $X_{i,r}$ randomly over the choices of h, as the excess quantity (error) for the estimation of k's count by the r^{th} row, with n being the unique keys count and m being the max layer at least semi-allocated after N updates. That is, m is the number of layers after N updates. Due to our expansion strategy, we know that $\frac{N}{uD}$ counters were added to each row. Therefore, we know that: $m \leq 1 + log_B(\frac{N(B-1)}{uDWB} + \frac{1}{B}) \leq log_B(N)$. We can thus infer the value of our random variables: $X_{k,r} = \sum_{k=1}^{n} \sum_{l=0}^{m} I_{k,j,r,l} * GS.estimate_{l,r}(k)$. Due to the linearity of expectation, $E[X_{k,r}]$ equals:

$$E[\sum_{k=1}^{n} \sum_{l=0}^{m-1} I_{k,j,r,l} * count_{l,r}(k)] = \sum_{k=1}^{n} \sum_{l=0}^{m-1} count_{l,r}(k) * E[I_{k,j,r,l}] \leq$$
$$\sum_{l=0}^{m-1} \sum_{k=1}^{n} count_{l,r}(k) * \frac{1}{WB^l} = \sum_{l=0}^{m-1} \frac{1}{WB^l} \sum_{k=1}^{n} count_{l,r}(k) \leq$$
$$\sum_{l=0}^{m-1} \frac{1}{WB^l}(B+1)uDWB^l = (B+1)muD \leq (B+1)uD * log_B(N)$$

For brevity, we denote $V = (B+1)uD * log_B(N)$

$$Pr[GS.estimate(k) > ActualCount(k) + eV] =$$
$$Pr[\forall_{D>r\geq 0} GS.estimate_r(k) > ActualCount(k) + eV] =$$
$$Pr[\forall_{D>r\geq 0} ActualCount(k) + X_{k,r} > ActualCount(k) + eV] =$$
$$Pr[\forall_{D>r\geq 0} X_{k,r} > eV] = Pr[X_{k,0} > V] * ... * Pr[X_{k,D-1} > eV] \leq$$
$$(\frac{E[X_{k,r}]}{eV})^D \leq (\frac{V}{eV})^D = e^{-D} \leq e^{-ln(\frac{1}{\delta})} = \delta$$

Then we can replace V and u with their actual values to get: $eV = u * e(B + 1)D * log_B(N) = M * log_B(N)$ □

Corollary 1.3.1. *This accuracy guarantee is independent of W. Therefore, we may select $W = 1$ for a minimal initial memory usage.*

Corollary 1.3.2. *A CMS with $\delta, \varepsilon > 0$ such that $D = \lceil ln(\frac{1}{\delta}) \rceil$ and $W = \lceil ln(\frac{e}{\varepsilon}) \rceil$ provides $Pr[CMS.estimate(k) > ActualCount(k) + \varepsilon N] \leq \delta$*

Corollary 1.3.3. *GS's relative error for a key k is defined*

$$\frac{|ActualCount(k) - GS.estimate(k)|}{ActualCount(k)}$$

Since we know that:

$$ActualCount(k) \leq N \wedge ActualCount(k) \leq GS.estimate(k)$$

we can deduce:

$$Pr[GS.estimate(k) > ActualCount(k) + Mlog_B(N)] =$$
$$Pr[\frac{GS.estimate(k) - ActualCount(k)}{ActualCount(k)} > \frac{Mlog_B(N)}{ActualCount(k)}] \leq$$
$$Pr[\frac{GS.estimate(k) - ActualCount(k)}{ActualCount(k)} > \frac{Mlog_B(N)}{N}] \leq \delta$$

Note that, $\lim_{N\to\infty} \frac{Mlog_B(N)}{N} = 0$*. Therefore, there is a probability greater than* $1-\lambda$ *that for any key* k*,* GS*'s estimation's relative error is smaller than an expression whose limit diminishes to* 0 *as* N *grows. I.e., the larger our stream, the smaller the expected relative error becomes.*