# Heavy Hitters

- We are back to our basic streaming model: $i_1, \ldots, i_n \in [d] = \{1, \cdots, d\}$.
- The frequency vector $x \in \mathbb{Z}^d$: $x_j = |\{t : i_t = j\}|$.

# Heavy Hitters

- We are back to our basic streaming model: $i_1, \ldots, i_n \in [d] = \{1, \cdots, d\}$.
- The frequency vector $x \in \mathbb{Z}^d$: $x_j = |\{t : i_t = j\}|$.
- For an integer $k > 1$, if an item appeared in more than $1/k$ fraction of the time, such an item is a *heavey hitter*.

# Heavy Hitters

- We are back to our basic streaming model: $i_1, \ldots, i_n \in [d] = \{1, \cdots, d\}$.
- The frequency vector $x \in \mathbb{Z}^d$: $x_j = |\{t : i_t = j\}|$.
- For an integer $k > 1$, if an item appeared in more than $1/k$ fraction of the time, such an item is a *heavey hitter*.
- The heavy hitter problem: at the end of the stream, output a set $S$:
  - If $j$ is a heavy hitter, i.e., $x_j > n/k$, then $j \in S$;

# Heavy Hitters

- We are back to our basic streaming model: $i_1, \ldots, i_n \in [d] = \{1, \cdots, d\}$.
- The frequency vector $x \in \mathbb{Z}^d$: $x_j = |\{t : i_t = j\}|$.
- For an integer $k > 1$, if an item appeared in more than $1/k$ fraction of the time, such an item is a *heavey hitter*.
- The heavy hitter problem: at the end of the stream, output a set $S$:
  - If $j$ is a heavy hitter, i.e., $x_j > n/k$, then $j \in S$;
  - If $j$ is not a heavy hitter, it is permitted to be in $S$;

# Heavy Hitters

- We are back to our basic streaming model: $i_1, \ldots, i_n \in [d] = \{1, \cdots, d\}$.
- The frequency vector $x \in \mathbb{Z}^d$: $x_j = |\{t : i_t = j\}|$.
- For an integer $k > 1$, if an item appeared in more than $1/k$ fraction of the time, such an item is a *heavey hitter*.
- The heavy hitter problem: at the end of the stream, output a set $S$:
  - If $j$ is a heavy hitter, i.e., $x_j > n/k$, then $j \in S$;
  - If $j$ is not a heavy hitter, it is permitted to be in $S$;
  - $|S| \leq k - 1$.

# Heavy Hitters by Count-Min or Count-Sketch

- Intuitively, in frequency estimation, we dealt with a harder problem

# Heavy Hitters by Count-Min or Count-Sketch

- Intuitively, in frequency estimation, we dealt with a harder problem
- The information from frequency sketch should suffice to answer heavy hitter.

# Heavy Hitters by COUNT-MIN or COUNT-SKETCH

- Intuitively, in frequency estimation, we dealt with a harder problem
- The information from frequency sketch should suffice to answer heavy hitter.
- Using COUNT-MIN or COUNT-SKETCH, we can output $S$ such that, with high probability,
  - If $j \in S$, then $x_j \geq (\frac{1}{k} - \epsilon)n$.

# Heavy Hitters by COUNT-MIN or COUNT-SKETCH

- Intuitively, in frequency estimation, we dealt with a harder problem
- The information from frequency sketch should suffice to answer heavy hitter.
- Using COUNT-MIN or COUNT-SKETCH, we can output $S$ such that, with high probability,
  - If $j \in S$, then $x_j \geq (\frac{1}{k} - \epsilon)n$.
  - If $j$ is a heavy hitter, then $j \in S$.

# Heavy Hitters by COUNT-MIN or COUNT-SKETCH

- Intuitively, in frequency estimation, we dealt with a harder problem
- The information from frequency sketch should suffice to answer heavy hitter.
- Using COUNT-MIN or COUNT-SKETCH, we can output $S$ such that, with high probability,
  - If $j \in S$, then $x_j \geq (\frac{1}{k} - \epsilon)n$.
  - If $j$ is a heavy hitter, then $j \in S$.
- The algorithm is relatively expensive:
  - we need to maintain $\Omega(\frac{1}{\epsilon} \log d)$ counters;

# Heavy Hitters by Count-Min or Count-Sketch

- Intuitively, in frequency estimation, we dealt with a harder problem
- The information from frequency sketch should suffice to answer heavy hitter.
- Using Count-Min or Count-Sketch, we can output $S$ such that, with high probability,
  - If $j \in S$, then $x_j \geq (\frac{1}{k} - \epsilon)n$.
  - If $j$ is a heavy hitter, then $j \in S$.
- The algorithm is relatively expensive:
  - we need to maintain $\Omega(\frac{1}{\epsilon} \log d)$ counters;
  - when each element arrives, we need to update all of these counters.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
  - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
    - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.
    - When $i_t$ arrives, if $j = i_t$, $C++$;

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
  - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.
  - When $i_t$ arrives, if $j = i_t$, $C++$;
  - Otherwise if $C > 0$, then $C--$;

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
  - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.
  - When $i_t$ arrives, if $j = i_t$, $C++$;
  - Otherwise if $C > 0$, then $C--$;
  - Otherwise replace $j \leftarrow i_t$, $C++$.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
    - Keep one element and a counter: $(j, C)$, initialize $C$ to $0$.
    - When $i_t$ arrives, if $j = i_t$, $C{+}{+}$;
    - Otherwise if $C > 0$, then $C{-}{-}$;
    - Otherwise replace $j \leftarrow i_t$, $C{+}{+}$.
    - In the end, output $j$.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
    - Keep one element and a counter: $(j, C)$, initialize $C$ to $0$.
    - When $i_t$ arrives, if $j = i_t$, $C++$;
    - Otherwise if $C > 0$, then $C--$;
    - Otherwise replace $j \leftarrow i_t$, $C++$.
    - In the end, output $j$.
- Correctness: a heavy hitter element $j$ (if it exists) must survive at the end.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
  - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.
  - When $i_t$ arrives, if $j = i_t$, $C++$;
  - Otherwise if $C > 0$, then $C--$;
  - Otherwise replace $j \leftarrow i_t$, $C++$.
  - In the end, output $j$.
- Correctness: a heavy hitter element $j$ (if it exists) must survive at the end.
- $j$ can be "killed" only by non-heavy hitters, which are fewer.

# Deterministic Algorithm for $k = 2$

- Recall the algorithm we gave in our first lecture for *Majority*:
  - Keep one element and a counter: $(j, C)$, initialize $C$ to 0.
  - When $i_t$ arrives, if $j = i_t$, $C{+}{+}$;
  - Otherwise if $C > 0$, then $C{-}{-}$;
  - Otherwise replace $j \leftarrow i_t$, $C{+}{+}$.
  - In the end, output $j$.
- Correctness: a heavy hitter element $j$ (if it exists) must survive at the end.
- $j$ can be "killed" only by non-heavy hitters, which are fewer.
- Another way to think of it: there can be at most $n/2$ elimination rounds.

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k - 1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k - 1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k-1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:
    - if there exists $s_j = i_t$, then $C_j++$;

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k-1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:
    - if there exists $s_j = i_t$, then $C_j + +$;
    - else, if there exists $C_j = 0$, then $s_j \leftarrow i_t$, $C_j + +$;

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k - 1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:
    - if there exists $s_j = i_t$, then $C_j + +$;
    - else, if there exists $C_j = 0$, then $s_j \leftarrow i_t$, $C_j + +$;
    - else, for all $j = 1, \cdots, k - 1$, $C_j - -$.

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k-1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:
    - if there exists $s_j = i_t$, then $C_j++$;
    - else, if there exists $C_j = 0$, then $s_j \leftarrow i_t$, $C_j++$;
    - else, for all $j = 1, \cdots, k-1$, $C_j--$.
  - In the end, output $s_1, \ldots, s_{k-1}$.

# Generalization to $k > 2$

- Algorithm due to Misra and Gries (1982):
  - Maintain $k - 1$ pairs of elements and their counters: $(s_1, C_1), \ldots, (s_{k-1}, C_{k-1})$. Initialize the counters to 0.
  - At input $i_t$:
    - if there exists $s_j = i_t$, then $C_j + +$;
    - else, if there exists $C_j = 0$, then $s_j \leftarrow i_t$, $C_j + +$;
    - else, for all $j = 1, \cdots, k - 1$, $C_j - -$.
  - In the end, output $s_1, \ldots, s_{k-1}$.

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.
- How many elimination rounds can there be?

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.
- How many elimination rounds can there be?
  - Think of each newly arrived element as joining a queue if it makes a counter increase;

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.
- How many elimination rounds can there be?
  - Think of each newly arrived element as joining a queue if it makes a counter increase;
  - And it is "killed' if either it causes an elimination or if it stands at the head of the queue when an elimination round happens;

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.
- How many elimination rounds can there be?
  - Think of each newly arrived element as joining a queue if it makes a counter increase;
  - And it is “killed’ if either it causes an elimination or if it stands at the head of the queue when an elimination round happens;
  - Each elimination round “kills” $k$ elements, so there can be at most $\lfloor \frac{n}{k} \rfloor$ such rounds.

# Proof of Correctness

- We must show that any heavy hitter $j$ is output in the end.
- How many elimination rounds can there be?
  - Think of each newly arrived element as joining a queue if it makes a counter increase;
  - And it is "killed' if either it causes an elimination or if it stands at the head of the queue when an elimination round happens;
  - Each elimination round "kills" $k$ elements, so there can be at most $\lfloor \frac{n}{k} \rfloor$ such rounds.
- Each heavy hitter occurs strictly more than $\frac{n}{k}$ times, so not all of its occurrences are killed at the end.