Tight Streaming Lower Bounds for Deterministic Approximate Counting

Yichuan Wang
(Tsinghua University)

Jan 13, 2025

The Problem Setting

Problem (k-Counter Approximate Counting)

Given an input $x \in [k]^n$. For each word $j \in [k]$, approximate the number of j's in x with additive error $n/(10k)$. Regime: $n \gg k \gg 1$.

The Problem Setting

Problem (k-Counter Approximate Counting)

Given an input $x \in [k]^n$. For each word $j \in [k]$, approximate the number of j's in x with additive error $n/(10k)$. Regime: $n \gg k \gg 1$.

Computation Model (Streaming Model)

We only have (standard order) read-once access to the input string. Memory size is bounded.

- i.e., after reading the first i input words, we can only carry limited amount of information when moving on to the $i+1, i+2, \cdots, n$-th words.

The Problem Setting

Problem (k-Counter Approximate Counting)

Given an input $x \in [k]^n$. For each word $j \in [k]$, approximate the number of j's in x with additive error $n/(10k)$. Regime: $n \gg k \gg 1$.

Computation Model (Streaming Model)

We only have (standard order) read-once access to the input string. Memory size is bounded.

- i.e., after reading the first i input words, we can only carry limited amount of information when moving on to the $i+1, i+2, \cdots, n$-th words.

Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

- Remark: the trivial algorithm (maintains the exact count) uses $\log \binom{n+k-1}{k-1} \le O(k \log(n/k))$ bits of memory.

Implication: Optimality of Misra-Gries

Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

- **Misra-Gries Algorithm (1982):** Let $U, n \gg k \gg 1$, given an input string $x \in [U]^n$. Their streaming algorithm approximates the count of each $j \in U$ with additive error $n/(10k)$, using $O(k \log(n/k) + k \log(U/k))$ bits of memory.

Implication: Optimality of Misra-Gries

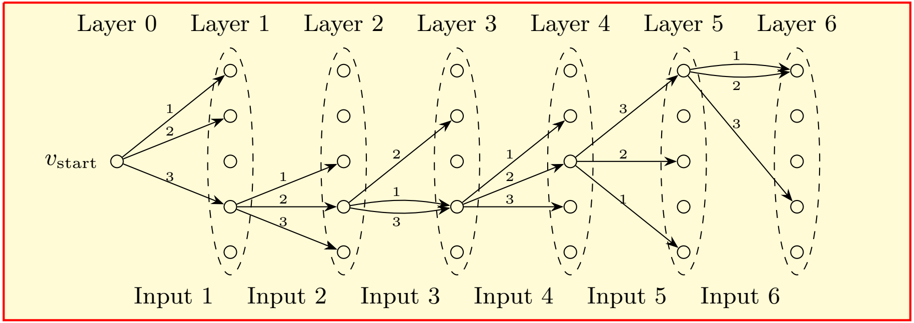
Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

- **Misra-Gries Algorithm (1982):** Let $U, n \gg k \gg 1$, given an input string $x \in [U]^n$. Their streaming algorithm approximates the count of each $j \in U$ with additive error $n/(10k)$, using $O(k \log(n/k) + k \log(U/k))$ bits of memory.
- On the lower bound:
  - Previously, we already known an $\Omega(k \log(U/k))$ lower bound. (Consider the case that k words each appear n/k times, each k-subset of $[U]$ must have a different output.)
  - Our result implies an $\Omega(k \log(n/k))$ lower bound!

Read-Once Branching Programs (ROBP)

- Streaming lower bounds $\Longleftarrow$ ROBP width lower bounds.
- Multi-layered directed graph. Nodes in layer m represents all possible memory states after the first m input words;
- n: input length; w: width (each layer contains $\leq w$ nodes), $w = 2^{\text{memory size}}$;
- Each node has k outgoing edges, which represents what is the next state given the next input word.



Previous Results

Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

Previous Results

Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

- For $k = 2$ (approximate the number of 1's in a $\{0, 1\}$-string), previously we did not even know a super-constant width lower bound!
  - We only know lower bounds in the *multiplicative error* setting: approximate counting with constant multiplicative error requires $n^{\Omega(1)}$ width. [M. Ajtai et al. (2022)]

Previous Results

Theorem (Main Result)

(Deterministic and worst-case) k-counter approximate counting requires $\Omega(k \log(n/k))$ bits of memory in the streaming model.

- For $k = 2$ (approximate the number of 1's in a $\{0, 1\}$-string), previously we did not even know a super-constant width lower bound!
  - We only know lower bounds in the *multiplicative error* setting: approximate counting with constant multiplicative error requires $n^{\Omega(1)}$ width. [M. Ajtai et al. (2022)]
- The standard communication bottleneck method (consider the communication complexity between the two halves of the input) does not work here.
  - Sending an approximation of #1's in the first half only needs $O(1)$ bits.

First Step: Rectangle Labeling

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

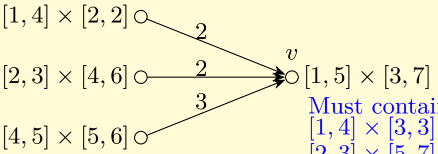
First Step: Rectangle Labeling

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Example: ($k = 3$)



Must contain:
$[1,4] \times [3,3]$,
$[2,3] \times [5,7]$,
$[4,5] \times [5,6]$.

First Step: Rectangle Labeling

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$. a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:

First Step: Rectangle Labeling

Rectangle Labeling

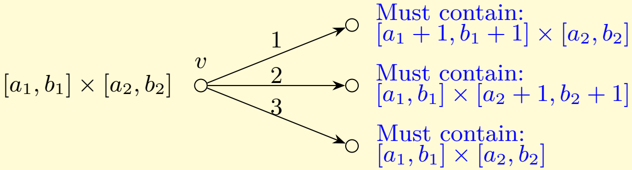
Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:

Example: $(k = 3)$



First Step: Rectangle Labeling

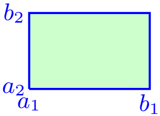
Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:
  - Split each rectangle into k rectangles; (itself + shifting in each direction)



First Step: Rectangle Labeling

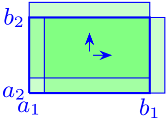
Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:
  - Split each rectangle into k rectangles; (itself + shifting in each direction)



First Step: Rectangle Labeling

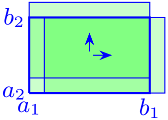
Rectangle Labeling

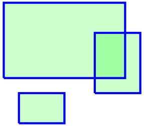
Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:
  - Split each rectangle into k rectangles; (itself + shifting in each direction)





  - Merge some of these new rectangles.

First Step: Rectangle Labeling

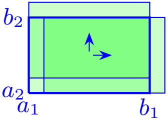
Rectangle Labeling

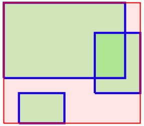
Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:
  - Split each rectangle into k rectangles; (itself + shifting in each direction)





  - Merge some of these new rectangles.

First Step: Rectangle Labeling

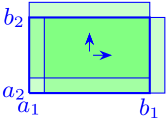
Rectangle Labeling

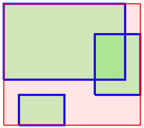
Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $b_j - a_j \leq (2 \cdot \text{additive error})$ in layer n. (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- What happens when moving on to the next layer:
  - Split each rectangle into k rectangles; (itself + shifting in each direction)





  - Merge some of these new rectangles.
  - Each layer contains $\leq w$ rectangles.

First Step: Rectangle Labeling

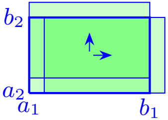
Rectangle Labeling

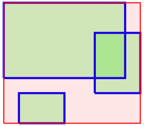
Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- These rectangles can be computed easily by dynamic programming;
- $\boldsymbol{b_j - a_j \leq (2 \cdot}$ **additive error) in layer** $\boldsymbol{n}$**.** (Characterizes the ROBP's correctness.)

Translate the whole problem into analyzing the rectangles:

- **What happens when moving on to the next layer:**
  - Split each rectangle into k rectangles; (itself + shifting in each direction)





  - Merge some of these new rectangles.
  - Each layer contains $\leq w$ rectangles.

Our Proof: Potential Function Analysis

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

Our Proof: Potential Function Analysis

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$. a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- **Plan:** wish to define $\{\Phi_m\}_{0 \le m \le n}$, where Φ_m only depends on the rectangle labels in layer m, such that:
  1. If the ROBP correctly computes k-counter approximate counting, then Φ_n is *small*;
  2. If w is small, then each increment $\Phi_{m+1} - \Phi_m$ is *large*.

Our Proof: Potential Function Analysis

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- **Plan:** wish to define $\{\Phi_m\}_{0 \le m \le n}$, where Φ_m only depends on the rectangle labels in layer m, such that:
  1. If the ROBP correctly computes k-counter approximate counting, then Φ_n is *small*;
  2. If w is small, then each increment $\Phi_{m+1} - \Phi_m$ is *large*.
- If w is too small, then **2.** implies that Φ_n is large, which contradicts **1.**

Our Potential Function

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

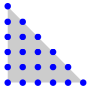
Our Potential Function

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- Only consider Φ_m $(n/2 \le m \le n)$; Consider all tuples in:

$$T := \left\{(x_1, \cdots, x_{k-1}) \in \mathbb{Z}_{\ge 0}^{k-1} : x_1 + \cdots + x_{k-1} \le n/2\right\}.$$



(All possible counts of $1, 2, \cdots, k-1$ in the first $n/2$ inputs.)

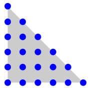
Our Potential Function

Rectangle Labeling

Label each node v in the ROBP with a rectangle $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_{k-1}, b_{k-1}]$.
a_j (resp. b_j) is the smallest (resp. largest) possible number of j's in order to reach v.

- Only consider Φ_m ($n/2 \leq m \leq n$); Consider all tuples in:

$$T := \left\{(x_1, \cdots, x_{k-1}) \in \mathbb{Z}_{\geq 0}^{k-1} : x_1 + \cdots + x_{k-1} \leq n/2\right\}.$$

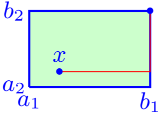


(All possible counts of $1, 2, \cdots, k-1$ in the first $n/2$ inputs.)

- For each tuple $(x_1, \cdots, x_{k-1}) \in T$, let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$;

Finally, let

$$\Phi_m := \sum_{(x_1, \cdots, x_{k-1}) \in T} \phi_m(x_1, \cdots, x_{k-1}).$$



Correctly Computes $\implies$ Φ_n is Small

Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

Correctly Computes $\Longrightarrow$ Φ_n is Small

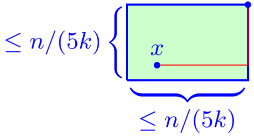
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- In each rectangle in layer n, we have $b_j - a_j \leq (2 \cdot \text{additive error}) = n/(5k)$, so

$$\phi_n(x_1, \cdots, x_{k-1}) \leq \sum_{j=1}^{k-1} (b_j - x_j) \leq (k-1) \cdot n/(5k) \leq n/5$$



Correctly Computes $\Longrightarrow$ Φ_n is Small

Definition (The Potential Function)

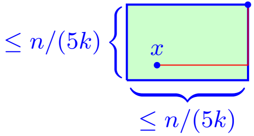
Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \le n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- In each rectangle in layer n, we have $b_j - a_j \le (2 \cdot \text{additive error}) = n/(5k)$, so

$$\phi_n(x_1, \cdots, x_{k-1}) \le \sum_{j=1}^{k-1} (b_j - x_j) \le (k-1) \cdot n/(5k) \le n/5$$

$$\Longrightarrow \Phi_n \le n/5 \cdot \binom{n/2 + k - 1}{k - 1}.$$



Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1+\cdots+x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \geq \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \geq \phi_m(x) + 1$.

- Therefore, $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$. $(n/2 \leq m \leq n-1)$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

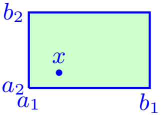
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \le n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \ge \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \ge \phi_m(x) + 1$.

The rectangle for computing $\phi_m(x)$:



- Therefore, $\Phi_{m+1} - \Phi_m \ge \binom{n/2+k-1}{k-1} - w$. $(n/2 \le m \le n-1)$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

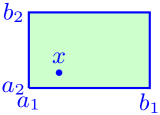
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \le n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \ge \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \ge \phi_m(x) + 1$.

  The rectangle for computing $\phi_m(x)$:

  It is contained in some rectangle in layer $m + 1$;
  Using this rectangle, we get an equal or larger $\phi_{m+1}(x)$.

- Therefore, $\Phi_{m+1} - \Phi_m \ge \binom{n/2+k-1}{k-1} - w$. $(n/2 \le m \le n - 1)$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

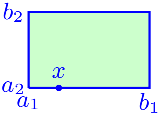
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \le n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \ge \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \ge \phi_m(x) + 1$.

  The rectangle for computing $\phi_m(x)$:

  Moreover part: we can shift this rectangle in some direction such that it still covers x;

- Therefore, $\Phi_{m+1} - \Phi_m \ge \binom{n/2+k-1}{k-1} - w$. $(n/2 \le m \le n-1)$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

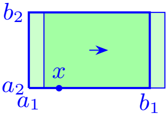
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \geq \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \geq \phi_m(x) + 1$.

  The rectangle for computing $\phi_m(x)$:

  Moreover part: we can shift this rectangle in some direction such that it still covers x;

- Therefore, $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$. $(n/2 \leq m \leq n-1)$

Small $w \implies \Phi_{m+1} - \Phi_m$ is Large

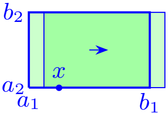
Definition (The Potential Function)

Let $\phi_m(x_1, \cdots, x_{k-1})$ be the maximum of $(b_1 - x_1) + \cdots + (b_{k-1} - x_{k-1})$ over all rectangles $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ in layer m that contains $(x_1, \cdots, x_{k-1})$. Let

$$\Phi_m := \sum_{x_1 + \cdots + x_{k-1} \leq n/2} \phi_m(x_1, \cdots, x_{k-1}).$$

- **Claim:** For each $x = (x_1, \cdots, x_{k-1})$, we have $\phi_{m+1}(x) \geq \phi_m(x)$. Moreover, if x is not at the left bottom corner of any rectangle in layer m, then $\phi_{m+1}(x) \geq \phi_m(x) + 1$.

The rectangle for computing $\phi_m(x)$:



Moreover part: we can shift this rectangle in some direction such that it still covers x;

Using this shifted rectangle, we get a $\phi_{m+1}(x)$ that increases by 1.

- Therefore, $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$. $(n/2 \leq m \leq n-1)$

Putting Them Together

- We have shown:
  - $\Phi_n \leq n/5 \cdot \binom{n/2+k-1}{k-1}$; and $\Phi_{n/2} \geq 0$;
  - $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$; $(n/2 \leq m \leq n-1)$

Putting Them Together

- We have shown:
  - $\Phi_n \leq n/5 \cdot \binom{n/2+k-1}{k-1}$; and $\Phi_{n/2} \geq 0$;
  - $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$; $(n/2 \leq m \leq n-1)$

$$\Longrightarrow n/5 \cdot \binom{n/2+k-1}{k-1} \geq \Phi_n \geq n/2 \cdot \left(\binom{n/2+k-1}{k-1} - w\right)$$

Putting Them Together

- We have shown:
  - $\Phi_n \leq n/5 \cdot \binom{n/2+k-1}{k-1}$; and $\Phi_{n/2} \geq 0$;
  - $\Phi_{m+1} - \Phi_m \geq \binom{n/2+k-1}{k-1} - w$; $(n/2 \leq m \leq n-1)$

$$\implies n/5 \cdot \binom{n/2+k-1}{k-1} \geq \Phi_n \geq n/2 \cdot \left(\binom{n/2+k-1}{k-1} - w\right)$$

$$\implies w \geq 3/5 \cdot \binom{n/2+k-1}{k-1} \geq 2^{\Omega(k \log(n/k))}$$

□

Thank You

- Thank you for listening.