UNIVERSITÄT DES SAARLANDES

max planck institut informatik

Sublinear Algorithms

Lecture 10: Applications II – (Modular) Subset Sum

European Research Council

Established by the European Commission

Karl Bringmann

July 6, 2020

Recap: Sparse Convolution

Sparse Polynomial Multiplication:

For $P(X) = \sum_{i=0}^{d-1} p_i X^i$ and $Q(X) = \sum_{i=0}^{d-1} q_i X^i$

their product $R = P \cdot Q$ has coefficients $r_i = \sum_{j=0}^{i} p_j q_{i-j}$

Out-of-bounds values are interpreted as 0

Given P, Q as lists of non-zero coefficients,

we can compute the coefficients of R in time $\tilde{O}(out)$ where $out = \#\{i \mid r_i \neq 0\}$

$\tilde{O}$ hides factor $\text{polylog}(d)$

Sparse Convolution:

For $u, v \in \mathbb{R}^d$, their convolution is the vector $u * v$ with $(u * v)_i = \sum_{j=0}^{i} u_j v_{i-j}$

Given u, v as lists of non-zero entries,

$u * v$ can be computed in time $\tilde{O}(out)$ where $out = \#\{i \mid (u * v)_i \neq 0\}$

Recap: Sparse Convolution

Sparse Polynomial Multiplication:

For $P(X) = \sum_{i=0}^{d-1} p_i X^i$ and $Q(X) = \sum_{i=0}^{d-1} q_i X^i$

their product $R = P \cdot Q$ has coefficients $r_i = \sum_{j=0}^{i} p_j q_{i-j}$

Given P, Q as lists of non-zero coefficients,

we can compute the coefficients of R in time $\tilde{O}(out)$ where $out = \#\{i \mid r_i \neq 0\}$

Sumset:

For $A, B \subseteq \{0,1,\dots,d\}$, their sumset is $A + B = \{a + b \mid a \in A, b \in B\}$

Given A, B as lists of all elements,

$A + B$ can be computed in time $\tilde{O}(out)$ where $out = |A + B|$

w.h.p.

$A + B$ corresponds to the non-zero coefficients of $\left(\sum_{a \in A} x^a\right)\left(\sum_{b \in B} x^b\right)$

Subset Sum

Goal: $\tilde{O}(out)$ time

Given a set X of n positive numbers, compute all sums of subsets of X

$out = |S(X)|$

Subset Sum:

Given $X = \{x_1, \dots, x_n\} \subseteq \{1, \dots, d\}$

Compute set of all subset sums:

$$S(X) := \{\sum_{y \in Y} y \mid Y \subseteq X\}$$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

$X = \{2,3,5\}$ $\quad S(X) = \{0,2,3,5,7,8,10\}$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Subset Sum:

Given $X = \{x_1, \dots, x_n\} \subseteq \{1, \dots, d\}$

Compute set of all subset sums:

$$S(X) := \left\{\sum_{y \in Y} y \mid Y \subseteq X\right\}$$

$$S_{i-1} \cup (S_{i-1} + x_i) = S_{i-1} + \{0, x_i\}$$

$$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Chain-like: $O(n \cdot out)$

1) Initialize $S_0 := \{0\}$

2) For $i = 1, \dots, n$:

$$S_i := S_{i-1} \cup (S_{i-1} + x_i)$$

3) Return S_n

[Bellman'57]

$$S_i = S(\{x_1, \dots, x_i\})$$

compute S_i in time

$$O(|S_i|) = O(|S_n|) = O(out)$$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Subset Sum:

Given $X = \{x_1, \dots, x_n\} \subseteq \{1, \dots, d\}$

Compute set of all subset sums:

$$S(X) := \left\{\sum_{y \in Y} y \mid Y \subseteq X\right\}$$

$$S_{i-1} \cup (S_{i-1} + x_i) = S_{i-1} + \{0, x_i\}$$

$$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 := \{0\}$

2) For $i = 1, \dots, n$:

$S_i := S_{i-1} + \{0, x_i\}$

3) Return S_n

$S_i = S(\{x_1, \dots, x_i\})$

compute S_i in time $\tilde{O}(|S_i|) = \tilde{O}(|S_n|) = \tilde{O}(out)$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} := \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^\ell$:

$$S_{\ell,i} := S_{\ell-1,2i-1} + S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$

Assumes that n is power of 2

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 := \{0\}$

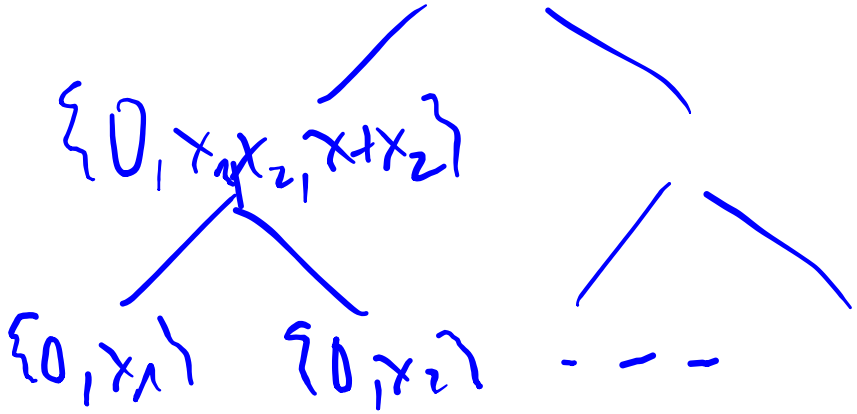
2) For $i = 1, \dots, n$:

$$S_i := S_{i-1} + \{0, x_i\}$$

3) Return S_n

$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$



Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} := \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^{\ell}$:

$$S_{\ell,i} := S_{\ell-1,2i-1} + S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$ Assumes that n is power of 2

$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Lem: For any non-empty A, B

$$|A + B| \geq |A| + |B| - 1$$

Proof:

Sort $A = \{a_1, \dots, a_n\}, B = \{b_1, \dots, b_m\}$. Then:

$$a_1 + b_1 < \cdots < a_n + b_1 < \cdots < a_n + b_m$$

$|A|$

$a_i + b_1$

$a_n + b_i$

$|B|$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} := \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^\ell$:

$$S_{\ell,i} := S_{\ell-1,2i-1} + S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$ Assumes that n is power of 2

$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Lem: For any non-empty A, B

$$|A + B| \geq |A| + |B| - 1$$

Cor: For any non-empty $A_1, \dots, A_k$

$$|A_1| + \cdots + |A_k| \leq |A_1 + \cdots + A_k| + k - 1$$

Proof:

$$\begin{aligned} |A_1 + \cdots + A_k| &\geq |A_1 + \cdots + A_{k-1}| + |A_k| - 1 \\ &\geq |A_1 + \cdots + A_{k-2}| + |A_{k-1}| + |A_k| - 2 \\ &\geq |A_1| + \cdots + |A_k| - (k-1) \end{aligned}$$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Tree-like: ~~$\tilde{O}(n \cdot out)$~~ $\tilde{O}(out)$

1) Initialize $S_{0,i} := \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^\ell$:

$S_{\ell,i} := S_{\ell-1,2i-1} + S_{\ell-1,2i}$

3) Return $S_{\log n,1}$ Assumes that n is power of 2

Lem: For any non-empty A, B

$$|A + B| \geq |A| + |B| - 1$$

Cor: For any non-empty $A_1, \dots, A_k$

$$|A_1| + \cdots + |A_k| \leq |A_1 + \cdots + A_k| + k - 1$$

Running Time:

$$\tilde{O}\left(\sum_\ell \sum_i |S_{\ell,i}|\right) = \tilde{O}\left(\sum_\ell \left(out + \frac{n}{2^\ell}\right)\right)$$

$$= \tilde{O}(out + n) = \tilde{O}(out)$$

$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Subset Sum

Given a set X of n positive numbers, compute all sums of subsets of X

Tree-like: ~~$\tilde{O}(n \cdot out)$~~ $\tilde{O}(out)$

1) Initialize $S_{0,i} \coloneqq \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^{\ell}$:

$$S_{\ell,i} \coloneqq S_{\ell-1,2i-1} + S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$ Assumes that n is power of 2

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 \coloneqq \{0\}$

2) For $i = 1, \dots, n$:

$$S_i \coloneqq S_{i-1} + \{0, x_i\}$$

3) Return S_n

$S(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Output-sensitive *algorithm for all Subset Sums*

follows from ***Sparse Convolution****,*

which is solved via ***sublinear*** *techniques*

Modular Subset Sum

Goal: $\tilde{O}(out)$ time

Given a set X of n positive numbers and integer m, compute all sums of subsets of X modulo m

Modular Subset Sum:

Given $X \subseteq \{1, \dots, d\}$ of size n

Compute $\mathcal{S}_m(X) := \mathcal{S}(X) \bmod m$

$= \{(\sum_{y \in Y} y) \bmod m \mid Y \subseteq X\}$

$\mathcal{S}(X) = \{0, x_1\} + \cdots + \{0, x_n\}$

$A + B = \{a + b \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

$X = \{2,3,5\}$ $\quad \mathcal{S}(X) \bmod 7 = \{0,1,2,3,5\}$

Modular Subset Sum

Given a set X of n positive numbers and integer m, compute all sums of subsets of X modulo m

Modular Subset Sum:

Given $X \subseteq \{1, \dots, d\}$ of size n

Compute $\mathcal{S}_m(X) \coloneqq \mathcal{S}(X) \bmod m$

$= \{(\sum_{y \in Y} y) \bmod m \mid Y \subseteq X\}$

1) Compute $C \coloneqq A + B$
2) Return $\{c \bmod m \mid c \in C\}$

$\mathcal{S}_m(X) = \{0, x_1\} +_m \dots +_m \{0, x_n\}$

$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 \coloneqq \{0\}$
2) For $i = 1, \dots, n$:

$S_i \coloneqq S_{i-1} +_m \{0, x_i\}$

3) Return S_n

$S_i = \mathcal{S}_m(\{x_1, \dots, x_i\})$

compute S_i in time
$\tilde{O}(|S_i|) = \tilde{O}(|S_n|) = \tilde{O}(out)$

Modular Subset Sum

Given a set X of n positive numbers and integer m, compute all sums of subsets of X modulo m

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} \coloneqq \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^{\ell}$:

$$S_{\ell,i} \coloneqq S_{\ell-1,2i-1} +_m S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 \coloneqq \{0\}$

2) For $i = 1, \dots, n$:

$$S_i \coloneqq S_{i-1} +_m \{0, x_i\}$$

3) Return S_n

$S_m(X) = \{0, x_1\} +_m \dots +_m \{0, x_n\}$

$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Modular Subset Sum

Given a set X of n positive numbers and integer m, compute all sums of subsets of X modulo m

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} \coloneqq \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^{\ell}$:

$$S_{\ell,i} \coloneqq S_{\ell-1,2i-1} +_m S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$

Lem: For any non-empty A, B

$$|A + B| \geq |A| + |B| - 1$$

$$|A +_m B| \geq |A| + |B| - 1$$

is wrong in general!

e.g. $A = B = \{0, \dots, m-1\}$

$S_m(X) = \{0, x_1\} +_m \dots +_m \{0, x_n\}$

$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Modular Subset Sum

Given a set X of n positive numbers and integer m, compute all sums of subsets of X modulo m

Tree-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_{0,i} \coloneqq \{0, x_i\}$ for all i

2) For $\ell = 1, \dots, \log n$: For $i = 1, \dots, n/2^{\ell}$:

$$S_{\ell,i} \coloneqq S_{\ell-1,2i-1} +_m S_{\ell-1,2i}$$

3) Return $S_{\log n,1}$

Approaches to time $\tilde{O}(out)$:

1) via sketching

2) via additive combinatorics:

Kneser's Thm:

$$|A +_m B| \geq |A| + |B| - |Sym(A + B)|$$

where $Sym(Z) = \{x \mid Z +_m x = Z\}$

3) via dynamic strings

[Mehlhorn, Sundar, Uhrig 1997]

$S_m(X) = \{0, x_1\} +_m \dots +_m \{0, x_n\}$

$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$ in time $\tilde{O}(out)$

Modular Subset Sum by Sketching

Data Structure $\mathcal{D}(v)$: maintains vector $v \in \mathbb{R}^m$ s.t.

- **Initialize:** set $v = (0, \dots, 0)$
- **Read**(i)**:** return v_i
- **Write**(i, s)**:** set $v_i := s$
- **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.

time $\tilde{O}(1) = \text{polylog}(m)$

- **Support:** computes $\text{supp}(v)$; correct w.h.p.

time $\tilde{O}(|\text{supp}(v)|)$ w.h.p.

- **Linear:** given access to $\mathcal{D}(u), \mathcal{D}(v)$ and α, β, we can simulate $\mathcal{D}(\alpha u + \beta v)$
- **Rotation:** given access to $\mathcal{D}(v)$ and x, we can simulate $\mathcal{D}(\text{rot}_x(v)) = \mathcal{D}(v_x, \dots, v_{m-1}, v_0, \dots, v_{x-1})$

time: constant-factor overhead

Modular Subset Sum by Sketching

Goal: Update S_{i-1} to S_i in time $\tilde{O}(|S_i \setminus S_{i-1}|)$

Then total time is $\tilde{O}(\sum_i |S_i \setminus S_{i-1}|)$

$$= \tilde{O}(\sum_i (|S_i| - |S_{i-1}|))$$

$$= \tilde{O}(|S_n| - |S_0|)$$

$$= \tilde{O}(out)$$

Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 := \{0\}$

2) For $i = 1, \ldots, n$:

$$S_i := S_{i-1} +_m \{0, x_i\}$$

3) Return S_n

$$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$$

Modular Subset Sum by Sketching

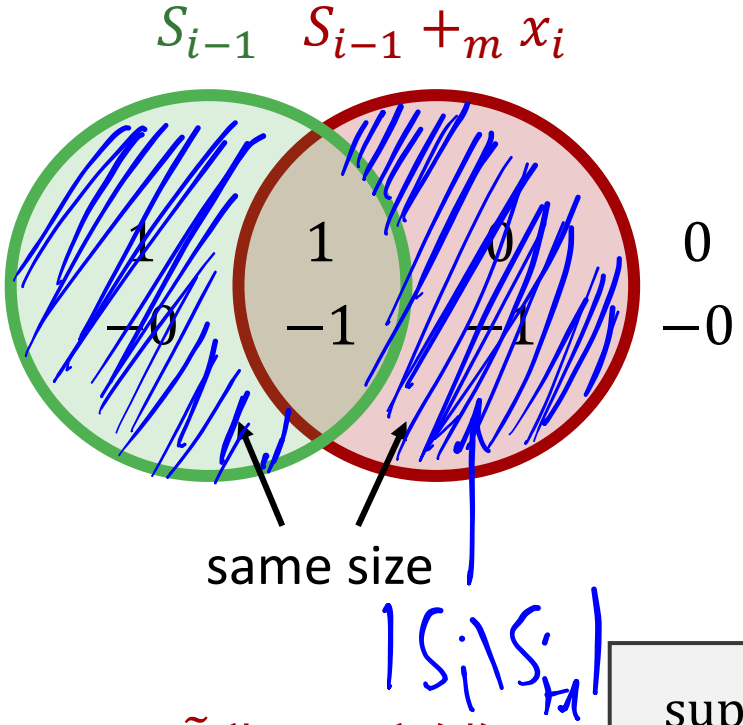
Goal: Update S_{i-1} to S_i in time $\tilde{O}(|S_i \setminus S_{i-1}|)$

Note $S_i = S_{i-1} \cup (S_{i-1} +_m x_i)$

$v := \mathbb{1}[S_{i-1}] - \mathbb{1}[S_{i-1} +_m x_i]$

$S_i = S_{i-1} \cup \mathrm{supp}(v)$

$|\mathrm{supp}(v)| = 2|S_i \setminus S_{i-1}|$



Chain-like: $\tilde{O}(n \cdot out)$

1) Initialize $S_0 := \{0\}$

2) For $i = 1, \ldots, n$:

$S_i := S_{i-1} +_m \{0, x_i\}$

3) Return S_n

Refined Goal: Compute $\mathrm{supp}(v)$ in time $\tilde{O}(|\mathrm{supp}(v)|)$

Total time is $\tilde{O}(\sum_i |S_i \setminus S_{i-1}|) = \tilde{O}(out)$

$\mathrm{supp}(v) = \{i \mid v_i \neq 0\}$

indicator vector $\mathbb{1}[A]$:

$$\mathbb{1}[A]_a = \begin{cases} 1 & \text{if } a \in A \\ 0 & \text{otherwise} \end{cases}$$

$$A +_m B = \{(a + b) \bmod m \mid a \in A, b \in B\}$$

Modular Subset Sum by Sketching

Data Structure $\mathcal{D}(v)$: maintains vector $v \in \mathbb{R}^m$ s.t.

- **Initialize:** set $v = (0, \dots, 0)$
- **Read**(i)**:** return v_i
- **Write**(i, s)**:** set $v_i \coloneqq s$
- **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.

} time $\tilde{O}(1)$

- **Support:** computes $\mathrm{supp}(v)$; correct w.h.p.

time $\tilde{O}(|\mathrm{supp}(v)|)$ w.h.p.

- **Linear:** given access to $\mathcal{D}(u), \mathcal{D}(v)$ and α, β, we can simulate $\mathcal{D}(\alpha u + \beta v)$
- **Rotation:** given access to $\mathcal{D}(v)$ and x, we can simulate $\mathcal{D}\big(\mathrm{rot}_x(v)\big) = \mathcal{D}(v_x, \dots, v_{d-1}, v_0, \dots, v_{x-1})$

} time: constant-factor overhead

Modular Subset Sum by Sketching

$v \coloneqq \mathbb{1}[S_{i-1}] - \mathbb{1}[S_{i-1} +_m x_i]$

$S_i = S_{i-1} \cup \mathrm{supp}(v)$

$|\mathrm{supp}(v)| = 2|S_i \setminus S_{i-1}|$

Refined Goal: Compute $\mathrm{supp}(v)$ in time $\tilde{O}(|\mathrm{supp}(v)|)$

Total time is $\tilde{O}(\sum_i |S_i \setminus S_{i-1}|) = \tilde{O}(out)$

Modular Subset Sum: $\tilde{O}(out)$

Maintain $\mathcal{D} = \mathcal{D}(\mathbb{1}[S_i])$

1) *// Initialize* $S_0 \coloneqq \{0\}$

   $\mathcal{D}$.Initialize and $\mathcal{D}$.Write$(0,1)$

2) For $i = 1, \ldots, n$:

   // here: $\mathcal{D} = \mathcal{D}(\mathbb{1}[S_{i-1}])$

   Simulate $\mathcal{D}' \coloneqq$
   $\mathcal{D}(\mathbb{1}[S_{i-1}] - \mathbb{1}[S_{i-1} +_m x_i])$

   $T \coloneqq \mathcal{D}'$.Support

   For each $x \in T$: $\mathcal{D}$.Write$(x, 1)$

Modular Subset Sum by Sketching

Data Structure $\mathcal{D}(v)$: maintains vector $v \in \mathbb{R}^m$ s.t.

- **Initialize:** set $v = (0, \dots, 0)$
- **Read**(i)**:** return v_i
- **Write**(i, s)**:** set $v_i \coloneqq s$
- **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.
- **Support:** computes $\mathrm{supp}(v)$; correct w.h.p.
- **Linear:** given access to $\mathcal{D}(u), \mathcal{D}(v)$ and α, β, we can simulate $\mathcal{D}(\alpha u + \beta v)$
- **Rotation:** given access to $\mathcal{D}(v)$ and x, we can simulate $\mathcal{D}\big(\mathrm{rot}_x(v)\big) = \mathcal{D}(v_x, \dots, v_{d-1}, v_0, \dots, v_{x-1})$

Modular Subset Sum: $\tilde{O}(out)$

Maintain $\mathcal{D} = \mathcal{D}(\mathbb{1}[S_i])$

1) *// Initialize* $S_0 \coloneqq \{0\}$

   $\mathcal{D}$.Initialize and $\mathcal{D}$.Write(0,1)

2) For $i = 1, \dots, n$:

   // here: $\mathcal{D} = \mathcal{D}(\mathbb{1}[S_{i-1}])$

   Simulate $\mathcal{D}' \coloneqq$

   $\mathcal{D}(\mathbb{1}[S_{i-1}] - \mathbb{1}[S_{i-1} +_m x_i])$

   $T \coloneqq \mathcal{D}'$.Support

   For each $x \in T$: $\mathcal{D}$.Write$(x, 1)$

Implementation of $\mathcal{D}$

Data Structure $\mathcal{D}(v)$: maintains vector $v \in \mathbb{R}^m$ s.t.

- **Initialize:** set $v = (0, \dots, 0)$
- **Read**(i)**:** return v_i
- **Write**(i, s)**:** set $v_i := s$
- **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.
- **Support:** computes $\mathrm{supp}(v)$; correct w.h.p.
- **Linear:** given access to $\mathcal{D}(u), \mathcal{D}(v)$ and α, β, we can simulate $\mathcal{D}(\alpha u + \beta v)$
- **Rotation:** given access to $\mathcal{D}(v)$ and x, we can simulate $\mathcal{D}(\mathrm{rot}_x(v)) = \mathcal{D}(v_x, \dots, v_{d-1}, v_0, \dots, v_{x-1})$

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta(2^b \log^2 m)$

$P := \bigcup_b P_b$

sparse representation

Store $v \in \mathbb{R}^m$

ω is m-th root of unity

Store $\mathrm{fold}(v, p) \in \mathbb{R}^p$ for each $p \in P$

$$\mathrm{fold}(v, p)_i = \sum_{j : j \bmod p = i} v_j$$

Store $\mathrm{fold}'(v, p) \in \mathbb{C}^p$ for each $p \in P$

$$\mathrm{fold}'(v, p)_i = \sum_{j : j \bmod p = i} v_j \omega^j$$

Implementation of $\mathcal{D}$

– **Rotation:** given $\mathcal{D}(v)$ and x, simulate $\mathcal{D}(v')$

for $v' := \mathrm{rot}_x(v) = (v_x, \dots, v_{d-1}, v_0, \dots, v_{x-1})$

$$v'_j = v_{j+_m x}$$

$$\begin{aligned}\mathrm{fold}(v', p)_i &= \textstyle\sum_{j : j \bmod p = i} v_{j+_m x} \\ &= \textstyle\sum_{j : j \bmod p = i+_m x} v_j \\ &= \mathrm{fold}(v, p)_{i+_m x}\end{aligned}$$

$$\begin{aligned}\mathrm{fold}'(v', p)_i &= \textstyle\sum_{j : j \bmod p = i} v_{j+_m x} \cdot \omega^j \\ &= \textstyle\sum_{j : j \bmod p = i+_m x} v_j \cdot \omega^{j-x} \\ &= \mathrm{fold}'(v, p)_{i+_m x} \cdot \omega^{-x}\end{aligned}$$

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta\left(2^b \log^2 m\right)$

$P := \bigcup_b P_b$

Store $v \in \mathbb{R}^m$ ω is m-th root of unity

Store $\mathrm{fold}(v, p) \in \mathbb{R}^p$ for each $p \in P$

$$\mathrm{fold}(v, p)_i = \textstyle\sum_{j : j \bmod p = i} v_j$$

Store $\mathrm{fold}'(v, p) \in \mathbb{C}^p$ for each $p \in P$

$$\mathrm{fold}'(v, p)_i = \textstyle\sum_{j : j \bmod p = i} v_j \omega^j$$

Implementation of $\mathcal{D}$

– **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.

1) For each $p \in P$:

  2) For $\Theta(\log^2 m)$ random $r \in \{0, .., p-1\}$:

    3) If $\text{fold}(v, p)_r \neq 0$: return $v \neq \vec{0}$

4) return $v = \vec{0}$

If $v \neq \vec{0}$: let $b = \log|\text{supp}(v)|$, fix $p \in P_b$

Each $v_j \neq 0$ is **isolated** with probability $\Omega(1)$, since #divisors prime

$2^b = \#j$'s

$$\mathbb{E}\left[\begin{matrix}\#j' \neq j \text{ s.t. } v_{j'} \neq 0 \text{ and} \\ (j - j') \bmod p = 0\end{matrix}\right] \leq \frac{|\text{supp}(v)| \cdot \log m}{\#\text{primes in } \Theta(2^b \log^2 m)} \leq \frac{1}{10}$$

$\Omega(2^b \log m)$

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

  Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta(2^b \log^2 m)$

$P \coloneqq \bigcup_b P_b$

Store $v \in \mathbb{R}^m$ ω is m-th root of unity

Store $\text{fold}(v, p) \in \mathbb{R}^p$ for each $p \in P$

$$\text{fold}(v, p)_i = \sum_{j : j \bmod p = i} v_j = v_j$$

Store $\text{fold}'(v, p) \in \mathbb{C}^p$ for each $p \in P$

$$\text{fold}'(v, p)_i = \sum_{j : j \bmod p = i} v_j \omega^j$$

Implementation of $\mathcal{D}$

– **TestZero:** checks if $v = (0, \dots, 0)$; correct w.h.p.

1) For each $p \in P$:

  2) For $\Theta(\log^2 m)$ random $r \in \{0, .., p-1\}$:

    3) If $\text{fold}(v,p)_r \neq 0$: return $v \neq \vec{0}$

4) return $v = \vec{0}$

If $v \neq \vec{0}$: let $b = \log|\text{supp}(v)|$, fix $p \in P_b$

Each $v_j \neq 0$ is **isolated** with probability $\Omega(1)$

W. prob. $\Omega(1)$, an $\Omega(1)$-fraction of all $v_j \neq 0$ are isolated

W. prob. $\Omega(1)$, an $\Omega\left(\frac{1}{\log^2 m}\right)$-fraction of all r's yield $v \neq \vec{0}$

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

  Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta\left(2^b \log^2 m\right)$

$P \coloneqq \bigcup_b P_b$

Store $v \in \mathbb{R}^m$ ω is m-th root of unity

Store $\text{fold}(v,p) \in \mathbb{R}^p$ for each $p \in P$

$$\text{fold}(v,p)_i = \sum_{j: j \bmod p = i} v_j$$

Store $\text{fold}'(v,p) \in \mathbb{C}^p$ for each $p \in P$

$$\text{fold}'(v,p)_i = \sum_{j: j \bmod p = i} v_j \omega^j$$

Implementation of $\mathcal{D}$

– **Support:** computes $\mathrm{supp}(v)$; correct w.h.p.

First suppose we know $b = \log|\mathrm{supp}(v)|$

PromiseSupport(b):

1) Initialize $S = \emptyset,\ u = (0, \dots, 0)$

2) For each $p \in P_b$: For $i = 0, \dots, p-1$:

3) $z \coloneqq \mathrm{fold}'(v,p)_i / \mathrm{fold}(v,p)_i$

4) $x \coloneqq \mathrm{Invert}(z)$ // $\omega^j \mapsto j$

5) If $v_x \neq 0$: $S \coloneqq S \cup \{x\},\ u_x \coloneqq v_x$

5) Return (S, u)

Each $v_j \neq 0$ is **isolated** by some p w.h.p.

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta(2^b \log^2 m)$

$P \coloneqq \bigcup_b P_b$

Store $v \in \mathbb{R}^m$ ω is m-th root of unity

Store $\mathrm{fold}(v,p) \in \mathbb{R}^p$ for each $p \in P$

$$\mathrm{fold}(v,p)_i = \sum_{j : j \bmod p = i} v_j$$

Store $\mathrm{fold}'(v,p) \in \mathbb{C}^p$ for each $p \in P$

$$\mathrm{fold}'(v,p)_i = \sum_{j : j \bmod p = i} v_j \omega^j$$

Implementation of $\mathcal{D}$

– **Support:** computes $\mathrm{supp}(v)$; correct w.h.p.

First suppose we know $b = \log|\mathrm{supp}(v)|$

Now if we do not know b:

1) For $b = 1, \dots, \log m$:

   3) $(S, u) \coloneqq$ **PromiseSupport**(b)

   4) Build $\mathcal{D}(u)$

   5) If $\mathcal{D}(u - v)$.TestZero: return S

Can only terminate with correct vector $u = v$

Terminates at the latest for $b = \log|\mathrm{supp}(v)|$

Implementation of $\mathcal{D}(v)$:

For $b = 1, \dots, \log m$:

Pick set P_b of $\Theta(\log m)$ random primes in the range $\Theta\left(2^b \log^2 m\right)$

$P \coloneqq \bigcup_b P_b$

Store $v \in \mathbb{R}^m$ ω is m-th root of unity

Store $\mathrm{fold}(v, p) \in \mathbb{R}^p$ for each $p \in P$

$$\mathrm{fold}(v, p)_i = \sum_{j : j \bmod p = i} v_j$$

Store $\mathrm{fold}'(v, p) \in \mathbb{C}^p$ for each $p \in P$

$$\mathrm{fold}'(v, p)_i = \sum_{j : j \bmod p = i} v_j \omega^j$$

More Material

Modular Subset Sum by Sketching:

[Axiotis, Backurs, Tzamos „Fast Modular Subset Sum using Linear Sketching“ 2019]

See also:

[Bringmann „A Near-Linear Pseudopolynomial Time Algorithm for Subset Sum“ 2017]

[Bringmann, Nakos „Top-k Convolution and the Quest for Near-Linear Output-Sensitive Subset Sum“ 2020]

See you on Thursday!