A Faster Pseudopolynomial Time Algorithm for Subset Sum

Konstantinos Koiliaris, Chao Xu
June 6, 2017

University of Illinois, Urbana-Champaign
The Subset Sum Problem

Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$ and a target number $t$. 
The Subset Sum Problem

Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$ and a target number $t$.

Output: Is there a subset $T$ of $S$ such that $\sum_{x_i \in T} x_i = t$?
The Subset Sum Problem

Classic problem.
The Subset Sum Problem

Classic problem.

One of Karp’s original NP-hard problems.

[Karp ’72]
The Subset Sum Problem

**Classic** problem.

One of Karp’s original **NP-hard** problems.  

**Weakly** NP-complete  

[Karp ’72]
Classic problem.

One of Karp’s original NP-hard problems. [Karp ’72]

Weakly NP-complete

Textbook DP algorithm due to Bellman that runs in $O(nt)$ pseudopolynomial time. [Bellman ’56]
Faster pseudopolynomial time algorithm for subset sum implies faster polynomial time algorithms for various problems.
Applications

As a subroutine:

- knapsack
- scheduling
- graph problems with cardinality constraints

In practice:

- power indices (Voting Theory)
- set-based queries (Database)
- Subset sum based keys (Security)
Previous Work: Deterministic pseudopolynomial algorithms

- Original DP solution: $O(nt)$ — [Bellman ’56]
• Original DP solution: $O(nt)$ — [Bellman ’56]
• Fast for small max $S$: $O(n \max S)$ — [Pisinger ’91]
Previous Work: Deterministic pseudopolynomial algorithms

- Original DP solution: $O(nt)$ — [Bellman ’56]
- Fast for small $\max S$: $O(n \max S)$ — [Pisinger ’91]
- Fast for small $\sigma$: $O(\sigma^{3/2})$ — [Klinz et al. ’99]
Previous Work: Deterministic pseudopolynomial algorithms

- Original DP solution: $O(nt)$ — [Bellman '56]
- Fast for small max $S$: $O(n \max S)$ — [Pisinger '91]
- Fast for small $\sigma$: $O(\sigma^{3/2})$ — [Klinz et al. '99]
- Data structure: $\tilde{O}(n \max S)$ — [Eppstein '97, Serang '14, '15]
Previous Work: Deterministic pseudopolynomial algorithms

- Original DP solution: $O(nt)$ — [Bellman ’56]
- Fast for small max $S$: $O(n\ max\ S)$ — [Pisinger ’91]
- Fast for small $\sigma$: $O(\sigma^{3/2})$ — [Klinz et al. ’99]
- Data structure: $\tilde{O}(n\ max\ S)$ — [Eppstein ’97, Serang ’14, ’15]
- RAM Model implementation of Bellman: $O(nt/\ log\ t)$ — [Pisinger ’03]
Previous Work: Deterministic pseudopolynomial algorithms

- Original DP solution: $O(nt)$ — [Bellman ’56]
- Fast for small max $S$: $O(n \max S)$ — [Pisinger ’91]
- Fast for small $\sigma$: $O(\sigma^{3/2})$ — [Klinz et al. ’99]
- Data structure: $\tilde{O}(n \max S)$ — [Eppstein ’97, Serang ’14, ’15]
- RAM Model implementation of Bellman: $O(nt/\log t)$ — [Pisinger ’03]
- First poly space algorithm: $\tilde{O}(n^3t)$ — [Lokshtanov et al. ’10]
Main Theorem [Koiliaris & Xu ‘17]. The subset sum problem can be decided in $\tilde{O}(\min\{\sqrt{nt}, t^{4/3}\})$ time.
Main Theorem [Koiliaris & Xu ‘17]. The subset sum problem can be decided in $\tilde{O}(\min\{\sqrt{nt}, t^{4/3}\})$ time.

Fastest deterministic pseudopolynomial time algorithm for the problem.
Main Theorem [Koiliaris & Xu ‘17]. The subset sum problem can be decided in $\tilde{O}(\min\{\sqrt{nt}, t^{4/3}\})$ time.

Fastest deterministic pseudopolynomial time algorithm for the problem.

Concurrent to our work, Bringmann showed that if randomization is allowed the subset sum problem can be decided in $\tilde{O}(t)$, with one-sided error probability $1/n$.

[Bringmann ‘17]
Main Theorem [Koiliaris & Xu ‘17]. *The subset sum problem can be decided in $\tilde{O}(\min\{\sqrt{nt}, t^{4/3}\})$ time.*

Fastest **deterministic** pseudopolynomial time algorithm for the problem.

Concurrent to our work, Bringmann showed that if *randomization* is allowed the subset sum problem can be decided in $\tilde{O}(t)$, with one-sided error probability $1/n$. [Bringmann ‘17]

Conditional lower bound: Subset sum solvable in $O(poly(n)t^{1-\epsilon})$ for any $\epsilon > 0$ implies faster algorithms for a wide variety of problems including set cover. [Bringmann ‘17]
Input: A set $S \subseteq \mathbb{Z}_m$ of $n$ numbers a target $t \in \mathbb{Z}_m$.

Output: Is there a subset $T$ of $S$ such that $\sum_{x \in T} x = t$?
Variants: Addition in $\mathbb{Z}_m$

**Input:** A set $S \subseteq \mathbb{Z}_m$ of $n$ numbers a target $t \in \mathbb{Z}_m$.

**Output:** Is there a subset $T$ of $S$ such that $\sum_{x \in T} x = t$?

Solvable in $O(nm)$ time using Bellman’s DP.
Input: A set $S \subseteq \mathbb{Z}_m$ of $n$ numbers a target $t \in \mathbb{Z}_m$.

Output: Is there a subset $T$ of $S$ such that $\sum_{x \in T} x = t$?

Solvable in $O(nm)$ time using Bellman’s DP.

Theorem ([Koiliaris & Xu ‘17])

The subset sum problem in $\mathbb{Z}_m$ can be decided in $\tilde{O}(\min\{\sqrt{nm}, m^{5/4}\})$ time.
**Variants: Addition in** \( \mathbb{Z}_m \)

**Input:** A set \( S \subseteq \mathbb{Z}_m \) of \( n \) numbers and a target \( t \in \mathbb{Z}_m \).

**Output:** Is there a subset \( T \) of \( S \) such that \( \sum_{x \in T} x = t \)?

Solvable in \( O(nm) \) time using Bellman’s DP.

**Theorem ([Koiliaris & Xu ‘17])**

The subset sum problem in \( \mathbb{Z}_m \) can be decided in \( \tilde{O}(\min\{\sqrt{nm}, m^{5/4}\}) \) time.

Different from the algorithm in \( \mathbb{N} \)!
Variants: multiset

**Input:** 2n natural numbers $x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n$ and a target number $t$.

**Output:** Does there exist non-negative integers $c_1, \ldots, c_n$, such that $\sum_{i=1}^{n} c_i x_i = t$ and $c_i \leq b_i$?
**Input:** 2n natural numbers $x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n$ and a target number $t$.

**Output:** Does there exist non-negative integers $c_1, \ldots, c_n$, such that $\sum_{i=1}^{n} c_i x_i = t$ and $c_i \leq b_i$?

- Solvable in $O(nt)$ time directly. [Faaland ‘73]
**Variants: multiset**

**Input:** 2n natural numbers \( x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n \) and a target number \( t \).

**Output:** Does there exist non-negative integers \( c_1, \ldots, c_n \), such that \( \sum_{i=1}^{n} c_i x_i = t \) and \( c_i \leq b_i \)?

- Solvable in \( O(nt) \) time directly. [Faaland '73]
- Reduces to subset sum with polylog factor blowup in near linear time. [Lawler '79]
**Input:** 2n natural numbers $x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n$ and a target number $t$.

**Output:** Does there exist non-negative integers $c_1, \ldots, c_n$, such that $\sum_{i=1}^{n} c_i x_i = t$ and $c_i \leq b_i$?

- Solvable in $O(nt)$ time directly. [Faaland ‘73]
- Reduces to subset sum with polylog factor blowup in near linear time. [Lawler ‘79]
- If all $b_i = \infty$, then it’s the coin change problem.
Variants: multiset

Input: 2n natural numbers $x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n$ and a target number $t$.

Output: Does there exist non-negative integers $c_1, \ldots, c_n$, such that $\sum_{i=1}^{n} c_i x_i = t$ and $c_i \leq b_i$?

• Solvable in $O(nt)$ time directly. [Faaland ‘73]
• Reduces to subset sum with polylog factor blowup in near linear time. [Lawler ‘79]
• If all $b_i = \infty$, then it’s the coin change problem.
  • $O(nx_1)$ time [Böcker and Lipták ‘07]
Input: 2n natural numbers $x_1, x_2, x_3, \ldots, x_n, b_1, \ldots, b_n$ and a target number $t$.

Output: Does there exist non-negative integers $c_1, \ldots, c_n$, such that $\sum_{i=1}^{n} c_i x_i = t$ and $c_i \leq b_i$?

- Solvable in $O(nt)$ time directly. [Faaland '73]
- Reduces to subset sum with polylog factor blowup in near linear time. [Lawler '79]
- If all $b_i = \infty$, then it’s the coin change problem.
  - $O(nx)$ time [Böcker and Lipták ‘07]
  - $\tilde{O}(t)$ time. [Bringmann '17]
Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$, cardinality constraint $k$ and target number $t$.

Output: Does there exist a subset of $S$ of size $k$ that sums to $t$?

• Solvable in $O(knt)$ time by modifying Bellman’s DP.
Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$, cardinality constraint $k$ and target number $t$.

Output: Does there exists a subset of $S$ of size $k$ that sums to $t$?

- Solvable in $O(knt)$ time by modifying Bellman’s DP.
- We can solve it in $\tilde{O}(nt)$ time.
Variants: Return a solution

- Instead of the decision problem, what if we want the actual set that realizes the target?
Variants: Return a solution

• Instead of the decision problem, what if we want the actual set that realizes the target?
• Our algorithm handles it with polylog factor slow down.
Variants: Return a solution

- Instead of the decision problem, what if we want the actual set that realizes the target?
- Our algorithm handles it with polylog factor slow down.
- We can also count the number of solutions faster than the standard dynamic programming algorithm.
We present two algorithms:

- Solve subset sum in $\mathbb{N}$.
- Solve subset sum in $\mathbb{Z}_m$. 
Subset sums in $\mathbb{N}$
To solve the subset sum problem, we will consider the following **all subset sums** problem:
To solve the subset sum problem, we will consider the following all subset sums problem:

*Given a set $S$ of $n$ natural numbers and an (upper bound) $u$, compute all the realizable sums up to $u$.***
Notations

• \([x..y]\) = \{x, x + 1, \ldots, y\} is the set of integers in the interval \([x, y]\).
• \([x..y]\) = \{x, x + 1, \ldots, y\}\) is the set of integers in the interval \([x, y]\).
• \([u]\) = \([0..u]\).
• $[x..y] = \{x, x+1, \ldots, y\}$ is the set of integers in the interval $[x, y]$.
• $[u] = [0..u]$.
• For two sets $X$ and $Y$, $X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}$. 
Notations

• \([x..y] = \{x, x + 1, \ldots, y\}\) is the set of integers in the interval \([x, y]\).
• \([u] = [0..u]\).
• For two sets \(X\) and \(Y\), \(X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}\).
• The set of all subset sums of \(S\) is denoted by

\[
\Sigma(S) = \left\{ \sum_{t \in T} t \middle| T \subseteq S \right\}.
\]
Notations

- \([x..y] = \{x, x + 1, \ldots, y\}\) is the set of integers in the interval \([x, y]\).
- \([u] = [0..u]\).
- For two sets \(X\) and \(Y\), \(X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}\).
- The set of all subset sums of \(S\) is denoted by

\[
\Sigma(S) = \left\{ \sum_{t \in T} t \mid T \subseteq S \right\}.
\]

Finding all subset sums of \(S\) up to \(u\): compute \(\Sigma(S) \cap [u]\).
**Fact.** If $P$ and $Q$ form a partition of a set $S$, then $\sum(P) \oplus \sum(Q) = \sum(S)$.

Straightforward **divide-and-conquer** algorithm for the all subset sums problem:
Fact. If $P$ and $Q$ form a partition of a set $S$, then $\Sigma(P) \oplus \Sigma(Q) = \Sigma(S)$.

Straightforward divide-and-conquer algorithm for the all subset sums problem:

- Partition the set $S$ into two sets
**Fact.** If $P$ and $Q$ form a partition of a set $S$, then $\Sigma(P) \oplus \Sigma(Q) = \Sigma(S)$.

Straightforward **divide-and-conquer** algorithm for the all subset sums problem:

- Partition the set $S$ into two sets
- Recursively compute their subset sums
Fact. If $P$ and $Q$ form a partition of a set $S$, then $\Sigma(P) \oplus \Sigma(Q) = \Sigma(S)$.

Straightforward divide-and-conquer algorithm for the all subset sums problem:

- Partition the set $S$ into two sets
- Recursively compute their subset sums
- Combine them together with $\oplus$. 
Review of the Bellman’s dynamic programming algorithm

Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$ and an upper bound $u$.

Algorithm:

- $T_0 \leftarrow \{0\}$.
- $T_i \leftarrow T_{i-1} \cup \{s + x_i | s \in T_{i-1}, s + x_i \leq u\}$.

$O(nu)$ time.
Input: A set $S$ of $n$ natural numbers $x_1, x_2, x_3, \ldots, x_n$ and an upper bound $u$.

Algorithm:

\begin{itemize}
  \item return $[u] \cap \bigoplus_{i=1}^{n} \sum(\{x_i\})$.
\end{itemize}

$\sum(\{x\}) = \{0, x\}$. 
Theorem. Given $A, B \subseteq [u]$, $A \oplus B$ can be computed in $O(u \log u) = \tilde{O}(u)$ time.

Just use FFT
**Theorem.** Given $A, B \subseteq [u]$, $A \oplus B$ can be computed in $O(u \log u) = \tilde{O}(u)$ time.

Just use FFT

**Theorem.** Given $A, B \subseteq [u] \times [v]$, $A \oplus B$ can be computed in $O(uv \log uv) = \tilde{O}(uv)$ time.
If $S \subseteq [x..x+\ell]$, then we will show that $\Sigma(S) \cap [u]$ can be found in

- $O(n(x + \ell))$ time. (Algorithm 1)
- $O((u/x)^2\ell)$ time. (Algorithm 2)
Two algorithms for all subset sums

If $S \subseteq [x..x + \ell]$, then we will show that $\sum(S) \cap [u]$ can be found in

- $O(n(x + \ell))$ time. (Algorithm 1)
- $O((u/x)^2\ell)$ time. (Algorithm 2)

We balance the running time of both algorithms to get the desired result.
Algorithm 1
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.

Proof Sketch.
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.

Proof Sketch.

- Partition $S$ into two sets $L, R$ of (roughly) equal cardinality, and compute recursively $\Sigma(L)$ and $\Sigma(R)$. 
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.

Proof Sketch.

- Partition $S$ into two sets $L$, $R$ of (roughly) equal cardinality, and compute recursively $\Sigma(L)$ and $\Sigma(R)$.
- The sets $\Sigma(L)$, $\Sigma(R) \subseteq [n(x + \ell)]$. $\Sigma(L) \oplus \Sigma(R)$ in $\tilde{O}(n(x + \ell))$ time.
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.

Proof Sketch.

- Partition $S$ into two sets $L, R$ of (roughly) equal cardinality, and compute recursively $\Sigma(L)$ and $\Sigma(R)$.
- The sets $\Sigma(L), \Sigma(R) \subseteq [n(x + \ell)]$. $\Sigma(L) \oplus \Sigma(R)$ in $\tilde{O}(n(x + \ell))$ time.
- 
  $$T(n) = 2T(n/2) + \tilde{O}(n(x + \ell))$$
Lemma Given a set $S$ of $n$ numbers in $[x..x + \ell]$, one can compute the set of all subset sums $\Sigma(S)$ in $\tilde{O}(n(x + \ell))$ time.

Proof Sketch.

- Partition $S$ into two sets $L, R$ of (roughly) equal cardinality, and compute recursively $\Sigma(L)$ and $\Sigma(R)$.
- The sets $\Sigma(L), \Sigma(R) \subseteq [n(x + \ell)]$. $\Sigma(L) \oplus \Sigma(R)$ in $\tilde{O}(n(x + \ell))$ time.

$$T(n) = 2T(n/2) + \tilde{O}(n(x + \ell))$$

- Solves to $T(n) = \tilde{O}(n(x + \ell))$
Algorithm 2
Lemma. Given a set $S \subseteq [x..x + \ell]$ of size $n$, computing the set $\Sigma(S) \cap [u]$ takes $\tilde{O}((u/x)^2 \ell)$ time.
Algorithm 2: Idea

**Lemma.** Given a set $S \subseteq [x..x + \ell]$ of size $n$, computing the set $\Sigma(S) \cap [u]$ takes $\tilde{O}((u/x)^2 \ell)$ time.

**Main idea** If elements in $\Sigma(S)$ are larger than $u$, we can throw it away.
Lemma. Given a set $S \subseteq [x..x + \ell]$ of size $n$, computing the set $\Sigma(S) \cap [u]$ takes $\tilde{O} \left( \left( \frac{u}{x} \right)^2 \ell \right)$ time.

Main idea If elements in $\Sigma(S)$ are larger than $u$, we can throw it away. Sum of any $\left\lfloor \frac{u}{x} \right\rfloor + 1$ elements is greater than $u$, then we only need subset sums using size $\left\lfloor \frac{u}{x} \right\rfloor$ subsets.
Lemma. Given a set $S \subseteq [x..x + \ell]$ of size $n$, computing the set $\Sigma(S) \cap [u]$ takes $\tilde{O}((u/x)^2 \ell)$ time.

Main idea If elements in $\Sigma(S)$ are larger than $u$, we can throw it away. Sum of any $\lceil \frac{u}{x} \rceil + 1$ elements is greater than $u$, then we only need subset sums using size $\lfloor \frac{u}{x} \rfloor$ subsets.

Proof Sketch. Same algorithm:

1. Partition $S$ into $L$ and $R$
2. Compute $\Sigma(L) \cap [u]$ and $\Sigma(R) \cap [u]$ recursively
3. Combine through (a smarter implementation of) $\oplus$. 
Algorithm 2: A single recursive step
Algorithm 2: A single recursive step
Algorithm 2: A single recursive step

\[(\Sigma(L) \cap [u]) \oplus (\Sigma(R) \cap [u])\]
Algorithm 2: A single recursive step

* $z \in \Sigma(L) \cap [u]$. 
Algorithm 2: A single recursive step

- $z \in \Sigma(L) \cap [u]$.
- For some $L' \subseteq L$, $z = \sum_{s \in L'} s = \sum_{x+t \in L'} x + t$, $t \in [\ell]$. 
Algorithm 2: A single recursive step

• $z \in \Sigma(L) \cap [u]$.
• For some $L' \subseteq L$, $z = \sum_{s \in L'} s = \sum_{x + t \in L'} x + t$, $t \in [\ell]$.
• $|L'| \leq \lfloor u/x \rfloor = k$. 
Algorithm 2: A single recursive step

- \( z \in \Sigma(L) \cap [u] \).
- For some \( L' \subseteq L \), \( z = \sum_{s \in L'} s = \sum_{x+t \in L'} x + t \), \( t \in [\ell] \).
- \( |L'| \leq \lfloor u/x \rfloor = k \).
- \( z = ix + j \), where \( i \in [k] \), \( j \in [\ell k] \).
Algorithm 2: A single recursive step

\[ i \in [k], j \in [\ell k] \]
\[ z = ix + j \]
\[ k = \left\lfloor \frac{u}{x} \right\rfloor \]
\[ \cap \]
\[ \Sigma(L) \cap [u] \]
\[ \Sigma(R) \cap [u] \]
Algorithm 2: A single recursive step

$$i \in [k], j \in [\ell k]$$

$$z = ix + j$$

Lift to 2D

$$k = \left\lfloor \frac{u}{x} \right\rfloor$$

$$\Phi$$

$$\bigcap$$

$$\Sigma(L) \cap [u]$$

$$\Sigma(R) \cap [u]$$

$$(i, j)$$
Algorithm 2: A single recursive step

Lift to 2D

\[ i \in [k], \ j \in [\ell k] \]
\[ z = ix + j \]
\[ k = \left\lfloor \frac{u}{x} \right\rfloor \]
\[ \Phi(\Sigma(L) \cap [u]) \]
\[ \Phi(\Sigma(R) \cap [u]) \]
\[ A = \Phi(\Sigma(L) \cap [u]) \]
\[ B = \Phi(\Sigma(R) \cap [u]) \]
\[ A, B \subseteq [k] \times [\ell k] \]
Algorithm 2: A single recursive step

\[ i \in [k], j \in [\ell k] \]
\[ z = ix + j \]
\[ k = \left\lfloor \frac{u}{x} \right\rfloor \]
\[ \Phi^{-1} \]

\[ \cap \]
\[ \Sigma(L) \cap [u] \]
\[ \Sigma(R) \cap [u] \]
\[ \Phi \]

\[ A = \Phi(\Sigma(L) \cap [u]) \]
\[ B = \Phi(\Sigma(R) \cap [u]) \]
\[ A, B \subseteq [k] \times [\ell k] \]

\[ \Sigma(L) \oplus \Sigma(R) \cap [u] \]
\[ \Phi \]

\[ A \oplus B \]
\[ \tilde{O}(\ell k^2) = \tilde{O}((u/x)^2 \ell) \text{ time} \]
Let $T(n, \ell)$ be the running time of Algorithm 2 with input set $S \subseteq [x..x+\ell]$ of size $n$. 
Let $T(n, \ell)$ be the running time of Algorithm 2 with input set $S \subseteq [x..x + \ell]$ of size $n$.

\[ \ell_1 + \ell_2 = \ell. \]

\[
T(n, \ell) = T(n/2, \ell_1) + T(n/2, \ell_2) + \tilde{O}(\ell(u/x)^2) \\
= \tilde{O}(\ell(u/x)^2)
\]
Algorithm 3
Algorithm

Algorithm 3

AllSubsetSum3(S, u):

- Partition [u] into intervals \( l_i = [r_{i-1}..r_i - 1] \) for \( 0 \leq i \leq k \).
- Let \( S_i \leftarrow l_i \cap S \).
- Compute \( \Sigma(S_0) \) using Algorithm 1.
- Compute \( \Sigma(S_i) \) using Algorithm 2 for \( 1 \leq i \leq k \).
- Return \( \bigoplus_{i=0}^{k} \Sigma(S_i) \).
Algorithm 3

\[ r_i = \lfloor 2^i r_0 \rfloor \]
\[ k = O(\log u) \]
\[ S_i = S \cap [r_{i-1}..r_i - 1] \]
\[ n_i = |S_i| \]

\[ r_0 \quad r_1 \quad r_2 \quad \ldots \quad r_{k-1} \quad r_k = u \]
Algorithm 3

\[ \sum(S_0) \]

Find \( \sum(S_0) \)

Algorithm 1 \( \tilde{O}(n_0r_0) \)

\[ r_k = u \]
Algorithm 3

Find $\sum(S_i)$

Algorithm 2

$\tilde{O}\left(\left(\frac{u}{r_{i-1}}\right)^2 (r_i - r_{i-1})\right) = \tilde{O}\left(\frac{u^2}{r_{i-1}}\right)$
Find $\Sigma(S_i)$ for all $1 \leq i \leq k$

$$\sum_{i=1}^{k} \tilde{O}\left(\frac{u^2}{r_{i-1}}\right) = \tilde{O}\left(\frac{u^2}{r_0}\right)$$
Algorithm 3: Analysis

- Find $\Sigma(S_0)$ in $\tilde{O}(n_0r_0) = \tilde{O}(\min(n, r_0)r_0)$ time.
Algorithm 3: Analysis

- Find $\Sigma(S_0)$ in $\tilde{O}(n_0 r_0) = \tilde{O}(\min(n, r_0)r_0)$ time.
- Find $\Sigma(S_1), \ldots, \Sigma(S_k)$ in $\tilde{O}(u^2/r_0)$ time.
Algorithm 3: Analysis

- Find $\Sigma(S_0)$ in $\tilde{O}(n_0r_0) = \tilde{O}(\min(n, r_0)r_0)$ time.
- Find $\Sigma(S_1), \ldots, \Sigma(S_k)$ in $\tilde{O}(u^2/r_0)$ time.
- Find $\bigoplus_{i=0}^{k} \Sigma(S_i)$ in $\tilde{O}(ku) = \tilde{O}(u)$ time.
Algorithm 3: Analysis

- Find $\Sigma(S_0)$ in $\tilde{O}(n_0 r_0) = \tilde{O}(\min(n, r_0)r_0)$ time.
- Find $\Sigma(S_1), \ldots, \Sigma(S_k)$ in $\tilde{O}(u^2/r_0)$ time.
- Find $\bigoplus_{i=0}^k \Sigma(S_i)$ in $\tilde{O}(ku) = \tilde{O}(u)$ time.
- Total running time $\tilde{O}(u^2/r_0 + \min(n, r_0)r_0 + u)$. 
Algorithm 3: Analysis

- Find $\Sigma(S_0)$ in $\tilde{O}(n_0 r_0) = \tilde{O}(\min(n, r_0) r_0)$ time.
- Find $\Sigma(S_1), \ldots, \Sigma(S_k)$ in $\tilde{O}(u^2 / r_0)$ time.
- Find $\bigoplus_{i=0}^k \Sigma(S_i)$ in $\tilde{O}(k u) = \tilde{O}(u)$ time.
- Total running time $\tilde{O}(u^2 / r_0 + \min(n, r_0) r_0 + u)$.

- Set $r_0 = u / \sqrt{n}$, we get $\tilde{O}(\sqrt{n} u)$.
- Set $r_0 = u^{2/3}$, we get $\tilde{O}(u^{4/3})$. 
There exist inputs $x_1 < \ldots < x_n$, such that any divide-and-conquer algorithm that computes $\Sigma(S)$ by

- add parenthesis to this expression

$$\Sigma(x_1) \oplus \ldots \oplus \Sigma(x_n),$$

- compute all the intermediate output,

takes $\Omega(\min(\sqrt{nt}, t^{4/3}))$ time.
Subset sums in $\mathbb{Z}_m$
Overview of the result

\[ \mathbb{Z}_m = \{0, \ldots, m - 1\}, \text{ the integers modulo } m. \]
\( \mathbb{Z}_m = \{0, \ldots, m - 1\} \), the integers modulo \( m \).

**Theorem**

Let \( S \subseteq \mathbb{Z}_m \) be a set of size \( n \). \( \Sigma(S) \) can be found in \( \tilde{O}(\min(\sqrt{nm}, m^{5/4})) \) time.
Overview of the result

\[ \mathbb{Z}_m = \{0, \ldots, m - 1\}, \text{ the integers modulo } m. \]

**Theorem**

Let \( S \subseteq \mathbb{Z}_m \) be a set of size \( n \). \( \Sigma(S) \) can be found in 
\[ \tilde{O}(\min(\sqrt{nm}, m^{5/4})) \] time.

Not an adaptation of Algorithm 3.
• Algorithm 3 throws away sums that fall outside \([u]\).
The challenge

- Algorithm 3 throws away sums that fall outside $[u]$.
- All operations in $\mathbb{Z}_m$ stays in $\mathbb{Z}_m$. 
$\mathbb{Z}_m^* = \{x | x \in \mathbb{Z}_m, \gcd(x, m) = 1\}$, the set of units of $\mathbb{Z}_m$. 
\[ \mathbb{Z}_m^* = \{ x | x \in \mathbb{Z}_m, \gcd(x, m) = 1 \} \]

the set of units of \( \mathbb{Z}_m \).

Assume \( \ell \) is large enough \( (\Omega(m^{\frac{1}{\log \log m}})) \) in the remainder of the talk.
\[ \mathbb{Z}_m^* = \{x | x \in \mathbb{Z}_m, \gcd(x, m) = 1\} \], the set of units of \( \mathbb{Z}_m \).

Assume \( \ell \) is large enough (\( \Omega(m^{\frac{1}{\log \log m}}) \)) in the remainder of the talk.

The algorithm consists of a black box for solving subset sums when \( S \subseteq \mathbb{Z}_m^* \), and then apply divide and conquer depending on the divisibility of the elements in \( S \).
Subset sums in $\mathbb{Z}_m$

$S \subseteq \mathbb{Z}_m^*$
A segment of length $\ell$ is a set of the form $x[\ell] = \{0, x, 2x, \ldots, \ell x\}$. We denote $X[\ell] = \{ix | x \in X, i \in [\ell]\}$. 
A segment of length $\ell$ is a set of the form $x[\ell] = \{0, x, 2x, \ldots, \ell x\}$. We denote $X[\ell] = \{ix| x \in X, i \in [\ell]\}$.

$\Sigma(S)$ can be found quickly if $S$ is covered by a segment.

**Theorem**

$S \subseteq \mathbb{Z}_m$ is a n element subset of $x[\ell]$, then $\Sigma(S)$ can be found in $\tilde{O}(n\ell)$ time.
The algorithm when input is in $\mathbb{Z}_m^*$

\[ \ell = 3 \quad X = \{1, 2, 5\} \]

We partition the input by segments.

- Find $X$, such that $S \subseteq X[\ell]$. 

The algorithm when input is in $\mathbb{Z}_m^*$

We partition the input by segments.

- Find $X$, such that $S \subseteq X[\ell]$.
- Create a partition $\{S_x | x \in X\}$ of $S$, such that $S_x \subseteq x[\ell]$. 
The algorithm when input is in $\mathbb{Z}_m^*$

\[ \ell = 3 \quad X = \{1, 2, 5\} \]

We partition the input by segments.

- Find $X$, such that $S \subseteq X[\ell]$.
- Create a partition $\{S_x | x \in X\}$ of $S$, such that $S_x \subseteq x[\ell]$.
- return $\bigoplus_{x \in X} \Sigma(S_x)$. 
The algorithm when input is in $\mathbb{Z}_m^*$

The running time:

• The time for finding $X$, say $T_{n,m}$
• Find subset sums for $S_x$ takes $O(S_x)$. The total time over all $S_x$ is $O(n)$. 
• $S_x$ takes $O(X_m)$ time.

The total running time is $O(T_{n,m} + X_m)$. We need to find a small $X$ that induces a cover of $S$, and we have to find one fast.
The algorithm when input is in $\mathbb{Z}_m^*$

The running time:

- The time for finding $X$, say $T(n, \ell, m)$
The running time:

- The time for finding $X$, say $T(n, \ell, m)$
- Find subset sums for $\Sigma(S_x)$ takes $\tilde{O}(|S_x|\ell)$. 
The algorithm when input is in $\mathbb{Z}_m^*$

The running time:

- The time for finding $X$, say $T(n, \ell, m)$
- Find subset sums for $\Sigma(S_x)$ takes $\tilde{O}(|S_x|\ell)$. The total time over all $S_x$ is $\sum_{x \in X} \tilde{O}(|S_x|\ell) = \tilde{O}(n\ell)$. 

The algorithm when input is in $\mathbb{Z}_m^*$

The running time:

- The time for finding $X$, say $T(n, \ell, m)$
- Find subset sums for $\Sigma(S_x)$ takes $\tilde{O}(|S_x|\ell)$. The total time over all $S_x$ is $\sum_{x \in X} \tilde{O}(|S_x|\ell) = \tilde{O}(n\ell)$.
- $\bigoplus_{x \in X} \Sigma(S_x)$ takes $\tilde{O}(|X|m)$ time.

The total running time is $\tilde{O}(T(n, \ell, m) + n\ell + |X|m)$. 
The running time:

- The time for finding $X$, say $T(n, \ell, m)$
- Find subset sums for $\Sigma(S_x)$ takes $\tilde{O}(|S_x|\ell)$. The total time over all $S_x$ is $\sum_{x \in X} \tilde{O}(|S_x|\ell) = \tilde{O}(n\ell)$.
- $\bigoplus_{x \in X} \Sigma(S_x)$ takes $\tilde{O}(|X|m)$ time.

The total running time is $\tilde{O}(T(n, \ell, m) + n\ell + |X|m)$. We need to find a small $X$ that induces a cover of $S$, and we have to find one fast.
Theorem

For any $S \subseteq \mathbb{Z}_m^*$, there exists an $x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega(\frac{\ell}{m}|S|)$. 
Theorem

For any $S \subseteq \mathbb{Z}_m^*$, there exists a $x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega(\frac{\ell}{m} |S|)$.

- $b \in x[\ell]$ if there exists $a \in [\ell]$ such that $ax \equiv b \pmod{m}$. 
Covering $S \subseteq \mathbb{Z}_m^*$ by segments

**Theorem**

For any $S \subseteq \mathbb{Z}_m^*$, there exists $a x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega\left(\frac{\ell}{m} |S|\right)$.

- $b \in x[\ell]$ if there exists $a \in [\ell]$ such that $ax \equiv b \pmod{m}$.
- $ax \equiv b \pmod{m}$ has exactly one solution if $a, b \in \mathbb{Z}_m^*$. 
Covering $S \subseteq \mathbb{Z}_m^*$ by segments

**Theorem**

For any $S \subseteq \mathbb{Z}_m^*$, there exists a $x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega\left(\frac{\ell}{m} |S|\right)$.

- $b \in x[\ell]$ if there exists $a \in [\ell]$ such that $ax \equiv b \pmod{m}$.
- $ax \equiv b \pmod{m}$ has exactly one solution if $a, b \in \mathbb{Z}_m^*$.
- Each $b \in \mathbb{Z}_m^*$ is covered by $[\ell] \cap \mathbb{Z}_m^*$ segments: For each $a \in [\ell] \cap \mathbb{Z}_m^*$, there is a unique $x$ such that $b \in x[\ell]$. 
Covering $S \subseteq \mathbb{Z}_m^*$ by segments

**Theorem**

For any $S \subseteq \mathbb{Z}_m^*$, there exists a $x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega\left(\frac{\ell}{m}|S|\right)$.

- $b \in x[\ell]$ if there exists $a \in [\ell]$ such that $ax \equiv b \pmod{m}$.
- $ax \equiv b \pmod{m}$ has exactly one solution if $a, b \in \mathbb{Z}_m^*$.
- Each $b \in \mathbb{Z}_m^*$ is covered by $[\ell] \cap \mathbb{Z}_m^*$ segments: For each $a \in [\ell] \cap \mathbb{Z}_m^*$, there is a unique $x$ such that $b \in x[\ell]$.

$$\mathbb{E}_{\text{uniform } x \in \mathbb{Z}_m^*} [b \text{ covered by } x[\ell]] = \frac{|[\ell] \cap \mathbb{Z}_m^*|}{|\mathbb{Z}_m^*|} = \Omega\left(\frac{\ell}{m}\right)$$
Covering $S \subseteq \mathbb{Z}_m^*$ by segments

Theorem

For any $S \subseteq \mathbb{Z}_m^*$, there exists a $x \in \mathbb{Z}_m^*$, such that $|S \cap x[\ell]| = \Omega\left(\frac{\ell}{m} |S|\right)$.

- $b \in x[\ell]$ if there exists $a \in [\ell]$ such that $ax \equiv b \pmod{m}$.
- $ax \equiv b \pmod{m}$ has exactly one solution if $a, b \in \mathbb{Z}_m^*$.
- Each $b \in \mathbb{Z}_m^*$ is covered by $[\ell] \cap \mathbb{Z}_m^*$ segments: For each $a \in [\ell] \cap \mathbb{Z}_m^*$, there is a unique $x$ such that $b \in x[\ell]$.

$$\mathbb{E}_{\text{uniform } x \in \mathbb{Z}_m^*}[b \text{ covered by } x[\ell]] = \frac{|[\ell] \cap \mathbb{Z}_m^*|}{|\mathbb{Z}_m^*|} = \Omega\left(\frac{\ell}{m}\right)$$

- For any subset $S \subseteq \mathbb{Z}_m^*$, there is a $x[\ell]$ that covers $|S| \frac{\ell}{m}$ elements in $S$ in expectation.
Algorithm

\textsc{GreedySetCover}(S \subseteq \mathbb{Z}_m^*)

1. Pick \(x[\ell]\) such that \(|x[\ell] \cap S|\) is maximized.
2. \(S \leftarrow S \setminus x[\ell]\)
3. \textsc{GreedySetCover}(S)

Finds a cover of size \(O\left(\frac{m}{\ell} \log n\right)\) in \(O(n\ell)\) time.
Theorem

All subset sums with input $S \subseteq \mathbb{Z}_m^*$ can be solved in $\tilde{O}(\sqrt{nm})$ time.

Proof.

$$\tilde{O}(T(n, \ell, m) + n\ell + (\frac{m}{\ell})m) = \tilde{O}(\frac{m^2}{\ell} + n\ell)$$

Let $\ell = \frac{m}{\sqrt{n}}$. \qed
Theorem

All subset sums with input \( S \subseteq \mathbb{Z}_m^* \) can be solved in \( \tilde{O}(\sqrt{nm}) \) time.

Proof.

\[
\tilde{O}(T(n, \ell, m) + n\ell + (\frac{m}{\ell})m) = \tilde{O}(\frac{m^2}{\ell} + n\ell)
\]

Let \( \ell = \frac{m}{\sqrt{n}} \).

We can assume \( n = O(\sqrt{m}) \).
Theorem

All subset sums with input $S \subseteq \mathbb{Z}_m^*$ can be solved in $\tilde{O}(\sqrt{nm})$ time.

Proof.

$$\tilde{O}(T(n, \ell, m) + n\ell + (\frac{m}{\ell})m) = \tilde{O}(\frac{m^2}{\ell} + n\ell)$$

Let $\ell = \frac{m}{\sqrt{n}}$. \hfill \square

We can assume $n = O(\sqrt{m})$.

Theorem ([Hamidoune, Llad & Serra 08])

If $S \subseteq \mathbb{Z}_m^*$ and $|S| \geq 2\sqrt{m}$, then $\Sigma(S) = \mathbb{Z}_m$. 
Theorem

All subset sums with input \( S \subseteq \mathbb{Z}_m^* \) can be solved in \( \tilde{O}(\sqrt{nm}) \) time.

Proof.

\[
\tilde{O}(T(n, \ell, m) + n\ell + (\frac{m}{\ell})m) = \tilde{O}(\frac{m^2}{\ell} + n\ell)
\]

Let \( \ell = \frac{m}{\sqrt{n}} \).

We can assume \( n = O(\sqrt{m}) \).

Theorem ([Hamidoune, Llad & Serra 08])

If \( S \subseteq \mathbb{Z}_m^* \) and \( |S| \geq 2\sqrt{m} \), then \( \Sigma(S) = \mathbb{Z}_m \).

Theorem

All subset sums in \( \mathbb{Z}_m^* \) can be solved in \( \tilde{O}(\min(\sqrt{nm}, m^{5/4})) \) time.
Subset sums in $\mathbb{Z}_m$

$S \subseteq \mathbb{Z}_m$
\[ \mathbb{Z}_{m,d} = \{ x : x \in \mathbb{Z}_m \text{ and } \gcd(x, m) \mid d \}. \]
Definitions

- $\mathbb{Z}_{m,d} = \{ x : x \in \mathbb{Z}_m \text{ and } \gcd(x, m)|d \}$.
- $\mathbb{Z}^*_m = \mathbb{Z}_m, 1$. 
Definitions

• $\mathbb{Z}_{m,d} = \{x : x \in \mathbb{Z}_m \text{ and } \gcd(x, m)|d\}$.
• $\mathbb{Z}^*_m = \mathbb{Z}_{m,1}$.
• $\mathbb{Z}_m = \mathbb{Z}_{m,m}$.
Definitions

\[ \mathbb{Z}_{m,d} = \{ x : x \in \mathbb{Z}_m \text{ and } \gcd(x, m) \mid d \} \].

\[ \mathbb{Z}^*_m = \mathbb{Z}_{m,1} \].

\[ \mathbb{Z}_m = \mathbb{Z}_{m,m} \].

We define \textsc{AllSubsetSums}(S, m, d) as an algorithm that finds all subset sums of \( S \) in \( \mathbb{Z}_m \), if \( S \subseteq \mathbb{Z}_{m,d} \).
Definitions

- \( \mathbb{Z}_{m,d} = \{x : x \in \mathbb{Z}_m \text{ and } \gcd(x, m)|d\} \).
- \( \mathbb{Z}^*_m = \mathbb{Z}_{m,1} \).
- \( \mathbb{Z}_m = \mathbb{Z}_{m,m} \).

We define \( \text{ALLSUBSETSUMS}(S, m, d) \) as an algorithm that finds all subset sums of \( S \) in \( \mathbb{Z}_m \), if \( S \subseteq \mathbb{Z}_{m,d} \).

We solved the case for \( \text{ALLSUBSETSUMS}(S, m, 1) \).

\[ \Sigma(S) = \text{ALLSUBSETSUMS}(S, m, m) \]
The algorithm for all subset sums in $\mathbb{Z}_m$

\begin{itemize}
  \item $S/p = \{s/p : s \in S, p|s\}$
  \item $S%p = \{s : s \in S, p \nmid s\}$
\end{itemize}
The algorithm for all subset sums in $\mathbb{Z}_m$

\[ S/p = \{s/p : s \in S, p|s\} \]
\[ S\%p = \{s : s \in S, p \not| s\} \]

Algorithm

\textsc{AllSubsetSums}(S, m, d):

1. $d = 1$, use the previous algorithm.
2. $p \leftarrow$ the largest prime factor of $d$
3. [All elements in $S$ divisible by $p$]
   \[ A \leftarrow \text{AllSubsetSums}(S/p, m/p, d/p) \]
4. [All elements in $S$ not divisible by $p$]
   \[ B \leftarrow \text{AllSubsetSums}(S\%p, m, d/p) \]
5. return $(p \cdot A) \oplus B$
Example recursion tree where $S = \mathbb{Z}_6$

\[
S = \mathbb{Z}_6
\]

\[0 \ 1 \ 2 \ 3 \ 4 \ 5\]
Example recursion tree where $S = \mathbb{Z}_6$

\[ S = \mathbb{Z}_6 \]

0 1 2 3 4 5

\[ p = 3, d = 6 \]
Example recursion tree where $S = \mathbb{Z}_6$

\[ S = \mathbb{Z}_6 \]

0 1 2 3 4 5

\[ p = 3, \; d = 6 \]

1 2 4 5

\[ \%p \]
Example recursion tree where $S = \mathbb{Z}_6$

$p = 3, d = 6$

$S = \mathbb{Z}_6$

\[
\begin{array}{ccccc}
0 & 1 & 2 & 3 & 4 \\
\end{array}
\]

\[
\begin{array}{ccccc}
1 & 2 & 4 & 5 \\
\end{array}
\]

\[
\begin{array}{cc}
0 & 1 \\
\end{array}
\]

%$p$

$/p$
Example recursion tree where $S = \mathbb{Z}_6$

$p = 3, d = 6$

$p = 2, d = 2$
Example recursion tree where $S = \mathbb{Z}_6$

$p = 3, d = 6$

$p = 2, d = 2$
Example recursion tree where $S = \mathbb{Z}_6$
Example recursion tree where $S = \mathbb{Z}_6$
Example recursion tree where $S = \mathbb{Z}_6$

$S = \mathbb{Z}_6$

$p = 3, d = 6$

$p = 2, d = 2$

Divisors

$d_1 = 6$

$d_2 = 3$

$d_3 = 2$

$d_4 = 1$
Example recursion tree where $S = \mathbb{Z}_6$

$p = 3, d = 6$

$p = 2, d = 2$

Divisors

$d_1 = 6$
$d_2 = 3$
$d_3 = 2$
$d_4 = 1$

Total size $\sigma_1(m) = O(m \log \log m)$

$\sigma_i(m) = \sum_{d|m} d^i$. 
Run time analysis: Leaves

Compute $\Sigma(S_i)$ for each $i$. $|S_i| = n_i$. $d_i \leq m/i$ is the $i$th largest divisor of $m$.

$$\tilde{O}\left(\sum_i \min(\sqrt{n_i}d_i, d_i^{5/4})\right)$$
$$=\tilde{O}\left(\sum_i \min(\sqrt{n_i}m/i, (m/i)^{5/4})\right)$$
$$=\tilde{O}(\min(\sqrt{nm}, m^{5/4}))$$
Run time analysis: Internal nodes

- There are $O(\log m)$ levels.
- Each level, the time spent on $\oplus$ is
  $\tilde{O}(\sum_{d|m} d) = \tilde{O}(\sigma_1(m)) = \tilde{O}(m)$.
- The total running time over internal nodes are $\tilde{O}(m)$. 
Theorem

All subset sums in $\mathbb{Z}_m$ can be solved in $\tilde{O}(\min(\sqrt{nm}, m^{5/4}))$. 
Open Problems
Is there a deterministic $\tilde{O}(t)$ time algorithm for the subset sum problem matching its conditional lower bound?
Let $k = |\Sigma(S) \cap [t]|$. Assume $k \ll t$.

- Known: subset sum in $O(nk)$ time use Bellman’s DP algorithm.
- Can we obtain an algorithm with $\tilde{O}(\sqrt{nk})$ running time?
Open Problems: Covering $\mathbb{Z}_m$ by segments of length $\ell$

Let $f(m, \ell)$ be the minimum number of segments of length $\ell$ required to cover $\mathbb{Z}_m$. 

Lower Bound:

Upper Bound:

Theorem ([Chen, Shparlinski & 2018])

• $f(m, \ell) = O(m)$ if $m$ is prime.

• $f(m, \ell) = o(m)$.

Theorem ([Koiliaris & 2019])

Conjecture:

$f(m, \ell) = O(m \log m \log \log m)$.
Let $f(m, \ell)$ be the minimum number of segments of length $\ell$ required to cover $\mathbb{Z}_m$.

Lower Bound: $f(m, \ell) \geq \left\lceil \frac{m}{\ell} \right\rceil$
Let $f(m, \ell)$ be the minimum number of segments of length $\ell$ required to cover $\mathbb{Z}_m$.

Lower Bound: $f(m, \ell) \geq \lceil \frac{m}{\ell} \rceil$

Upper Bound:

**Theorem ([Chen, Shparlinski & Winterhof ‘13])**

- $f(m, \ell) = O\left(\frac{m}{\ell}\right)$ if $m$ is prime.
- $f(m, \ell) = \frac{m^{1+o(1)}}{\sqrt{\ell}}$. 
Let \( f(m, \ell) \) be the minimum number of segments of length \( \ell \) required to cover \( \mathbb{Z}_m \).

**Lower Bound:** \( f(m, \ell) \geq \left\lceil \frac{m}{\ell} \right\rceil \)

**Upper Bound:**

**Theorem ([Chen, Shparlinski & Winterhof ‘13])**

- \( f(m, \ell) = O\left(\frac{m}{\ell}\right) \) if \( m \) is prime.
- \( f(m, \ell) = \frac{m^{1+o(1)}}{\sqrt{\ell}} \).

**Theorem ([Koiliaris & Xu ‘17])**

\[
f(m, \ell) = \sigma_0(m) + O\left(\sigma_1(m) \log m / \ell\right) = \frac{m^{1+o(1)}}{\ell}
\]
Let $f(m, \ell)$ be the minimum number of segments of length $\ell$ required to cover $\mathbb{Z}_m$.

**Lower Bound:** $f(m, \ell) \geq \left\lceil \frac{m}{\ell} \right\rceil$

**Upper Bound:**

**Theorem ([Chen, Shparlinski & Winterhof ‘13])**

- $f(m, \ell) = O\left(\frac{m}{\ell}\right)$ if $m$ is prime.
- $f(m, \ell) = \frac{m^{1+o(1)}}{\sqrt{\ell}}$.

**Theorem ([Koiliaris & Xu ‘17])**

$f(m, \ell) = \sigma_0(m) + O(\sigma_1(m) \log m/\ell) = \frac{m^{1+o(1)}}{\ell}$

**Conjecture:** $f(m, \ell) = O\left(\frac{m}{\ell}\right)$
Thank you