A FASTER PSEUDOPOLYNOMIAL TIME ALGORITHM FOR SUBSET SUM

Konstantinos Koiliaris, Chao Xu

SODA 2017

University of Illinois, Urbana-Champaign

INTRODUCTION

Input: A set S of n positive integers $x_1, x_2, x_3, \ldots, x_n$ and a positive target integer t.

Input: A set S of n positive integers $x_1, x_2, x_3, \ldots, x_n$ and a positive target integer t.

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

Classical problem.

THE SUBSET SUM PROBLEM

Classical problem.

One of Karp's original NP-hard problems.

[Karp '72]

THE SUBSET SUM PROBLEM

Classical problem.

One of Karp's original NP-hard problems.

[Karp '72]

Weakly NP-complete

Classical 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]

As a subroutine:

As a subroutine:

- knapsack

As a subroutine:

- knapsack
- scheduling

APPLICATIONS

As a subroutine:

- knapsack
- scheduling
- graph problems with cardinality constraints

APPLICATIONS

As a subroutine:

- knapsack
- scheduling
- graph problems with cardinality constraints

In practice:

- power indices

As a subroutine:

- knapsack
- scheduling
- graph problems with cardinality constraints

In practice:

- power indices
- set-based queries in databases

As a subroutine:

- knapsack
- scheduling
- graph problems with cardinality constraints

In practice:

- power indices
- set-based queries in databases
- applications in security

- 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

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

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

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

PREVIOUS WORK

- Original DP solution: $O(nt)$ — **[Bellman '56]**
- Fast for small $\max S$: $O(n \max S)$ — **[Pisinger '91]**
- Fast for small σ: $O(\sigma^{3/2})$ — **[Klinz et al. '99]**
- Data structure: $\widetilde{O}(n \max S)$ — **[Eppstein '97, Serang '14, '15]**
- RAM Model implementation of Bellman: $O(nt/\log t)$ — **[Pisinger '03]**
- First poly space algorithm: $\widetilde{O}(n^3 t)$ — **[Lokshtanov et al. '10]**

We further improve the state-of-the-art for the subset sum by providing the fastest **deterministic** pseudopolynomial time algorithm for the problem.

OUR CONTRIBUTION

We further improve the state-of-the-art for the subset sum by providing the fastest **deterministic** pseudopolynomial time algorithm for the problem.

Main Theorem [Koiliaris & Xu '17]. *The subset sum problem can be decided in* $\widetilde{O}(\min\{\sqrt{n}t, t^{4/3}\})$ *time.*

We further improve the state-of-the-art for the subset sum by providing the fastest **deterministic** pseudopolynomial time algorithm for the problem.

Main Theorem [Koiliaris & Xu '17]. *The subset sum problem can be decided in* $\widetilde{O}(\min\{\sqrt{n}t, t^{4/3}\})$ *time.*

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

[Bringmann '17]

For this we will consider the following all subset sums problem:

For this we will consider the following all subset sums problem:

*Given a set S of n positive integers and an (upper bound) positive integer u, compute **all** the realizable sums up to u.*

ALL SUBSET SUMS

For this we will consider the following all subset sums problem:

Given a set S of n positive integers and an (upper bound) positive integer u, compute ***all*** *the realizable sums up to u.*

Computing all subset sums for some $u \geq t$ also answers the subset sum problem with target value t.

ALGORITHM

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

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

- Partition the set S into two sets

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

- Partition the set S into two sets
- Recursively compute their subset sums

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 using FFT

Main idea improve the **“conquer”** step:

Main idea improve the **“conquer”** step:

- If sets S, T lie in a short and light interval, then one can combine their subset sums quickly as their total sum will be small.

Main idea improve the **“conquer”** step:

- If sets S, T lie in a short and light interval, then one can combine their subset sums quickly as their total sum will be small.
- If sets S, T lie in a long and heavy interval, then one can combine their subset sums quickly by ignoring most of the sums as they exceed the upper bound.

NOTATION

- $[\![x : y]\!] = \{x, x+1, \ldots, y\}$ is the set of integers in the interval $[x, y]$.

NOTATION

- $[\![x : y]\!] = \{x, x+1, \ldots, y\}$ is the set of integers in the interval $[x, y]$.
- For two sets X and Y, $X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}$ is the set of pairwise sums.

- $[\![x : y]\!] = \{x, x+1, \ldots, y\}$ is the set of integers in the interval $[x, y]$.
- For two sets X and Y, $X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}$ is the set of pairwise sums.
- The set of all subset sums realizable by subsets of S is denoted by

$$\sum(S) = \left\{ \sum_{t \in T} t \;\middle|\; T \subseteq S \right\}.$$

NOTATION

- $[\![x : y]\!] = \{x, x+1, \ldots, y\}$ is the set of integers in the interval $[x, y]$.
- For two sets X and Y, $X \oplus Y = \{x + y \mid x \in X \text{ and } y \in Y\}$ is the set of pairwise sums.
- The set of all subset sums realizable by subsets of S is denoted by

$$\sum(S) = \left\{ \sum_{t \in T} t \,\middle|\, T \subseteq S \right\}.$$

- If P and Q form a partition of a set S, then $\sum(P) \oplus \sum(Q) = \sum(S)$.

Lemma [Short and Light]. *Given a set of positive integers S with total sum σ, one can compute the set of all subset sums $\sum(S) \cap [\![0 : u]\!]$ in $O(\sigma \log \sigma \log n)$ time.*

Lemma [Short and Light]. *Given a set of positive integers S with total sum σ, one can compute the set of all subset sums $\sum(S) \cap [\![0 : u]\!]$ in $O(\sigma \log \sigma \log n)$ time.*

Proof Sketch.

Lemma [Short and Light]. *Given a set of positive integers S with total sum σ, one can compute the set of all subset sums $\sum(S) \cap [\![0 : u]\!]$ in $O(\sigma \log \sigma \log n)$ time.*

Proof Sketch.

- Partition S into two sets L, R of (roughly) equal cardinality, and compute recursively $L' = \sum(L)$ and $R' = \sum(R)$.

Lemma [Short and Light]. *Given a set of positive integers S with total sum σ, one can compute the set of all subset sums $\sum(S) \cap [\![0 : u]\!]$ in $O(\sigma \log \sigma \log n)$ time.*

Proof Sketch.

- Partition S into two sets L, R of (roughly) equal cardinality, and compute recursively $L' = \sum(L)$ and $R' = \sum(R)$.
- Notice that since the sets $L', R' \subseteq [\![0 : \sigma]\!]$, we can compute $L' \oplus R'$ in $O(\sigma \log \sigma)$ time via FFT.

Lemma [Short and Light]. *Given a set of positive integers S with total sum σ, one can compute the set of all subset sums $\sum(S) \cap [\![0 : u]\!]$ in $O(\sigma \log \sigma \log n)$ time.*

Proof Sketch.

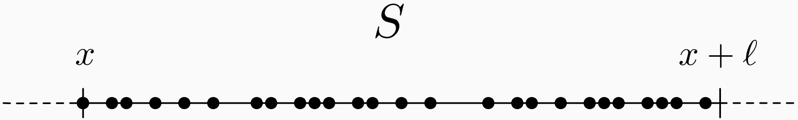
- Partition S into two sets L, R of (roughly) equal cardinality, and compute recursively $L' = \sum(L)$ and $R' = \sum(R)$.
- Notice that since the sets $L', R' \subseteq [\![0 : \sigma]\!]$, we can compute $L' \oplus R'$ in $O(\sigma \log \sigma)$ time via FFT.
- Since the divide-and-conquer recursion has $\log n$ levels, we get the $O(\log n(\sigma \log \sigma))$ promised running time.

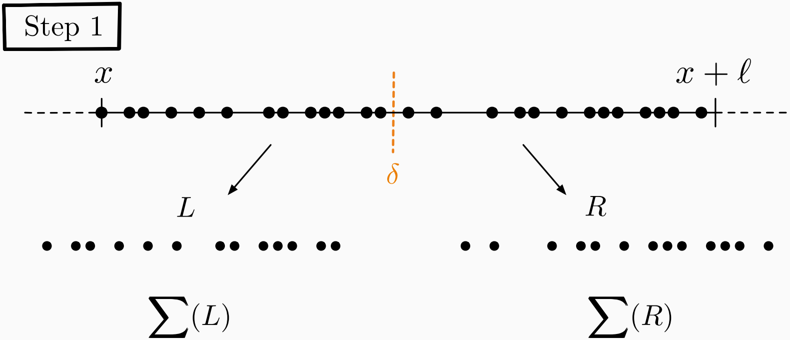
$\square$

Lemma [Long and Heavy]. *Given a set* $S \subseteq [\![x : x+\ell]\!]$ *of size* n*, computing the set* $\sum(S) \cap [\![0 : u]\!]$ *takes* $\widetilde{O}\big((u/x)^2\ell\big)$ *time.*

Lemma [Long and Heavy]. *Given a set* $S \subseteq [\![x : x + \ell]\!]$ *of size* n*, computing the set* $\sum(S) \cap [\![0 : u]\!]$ *takes* $\widetilde{O}\big((u/x)^2 \ell\big)$ *time.*

Proof Sketch.


S
x
$x + \ell$


Step 1
x
$x + \ell$
δ
L
R
$\sum(L)$
$\sum(R)$

$\boxed{\text{Step 2}}$

$$\sum(L), \sum(R)$$

$$\begin{array}{c}\Cup \\ z = ix + j\end{array}$$

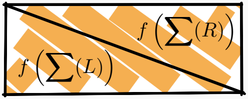
$\boxed{\text{Step 2}}$

$$\sum(L), \sum(R)$$

$$\underset{z}{\Cup}\! = ix + j \xrightarrow{f} (i, j) \in [\![0 : k]\!] \times [\![0 : \ell k]\!]$$

$\boxed{\text{Step 2}}$

$\sum(L), \sum(R) \xrightarrow{f}$

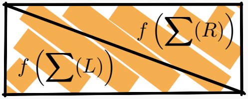


$\Cup$

$z = ix + j \xrightarrow{f} (i, j) \in [\![0:k]\!] \times [\![0:\ell k]\!]$

Higher # dimensions but shorter intervals

$\boxed{\text{Step 2}}$

$\sum(L), \sum(R) \xrightarrow{f}$ 

$\cup$

$z = ix + j \xrightarrow{f} (i, j) \in [\![0 : k]\!] \times [\![0 : \ell k]\!]$

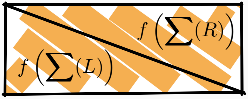
Higher # dimensions but shorter intervals

$$\text{FFT} \Downarrow \text{ in } \widetilde{O}(\ell k^2)$$

$$f\left(\sum(L)\right) \oplus f\left(\sum(R)\right)$$

$\boxed{\text{Step 2}}$

$$\sum(L), \sum(R) \xrightarrow{f}$$



Higher # dimensions but shorter intervals

$$\Cup$$

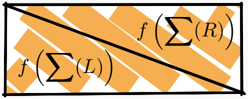
$$z = ix + j \xrightarrow{f} (i, j) \in [\![0:k]\!] \times [\![0:\ell k]\!]$$

$$\text{FFT} \Downarrow \text{ in } \widetilde{O}(\ell k^2)$$

$$\sum(L \cup R) \xleftarrow{f^{-1}} f\left(\sum(L)\right) \oplus f\left(\sum(R)\right)$$

$\boxed{\text{Step 2}}$

$$\textstyle\sum(L), \sum(R) \xrightarrow{f}$$



$$\cup$$

$$z = ix + j \xrightarrow{f} (i,j) \in [\![0:k]\!] \times [\![0:\ell k]\!]$$

Higher # dimensions but shorter intervals

$$\text{FFT} \Downarrow \text{ in } \widetilde{O}(\ell k^2)$$

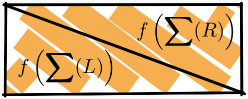
$$\textstyle\sum(L \cup R) \xleftarrow{f^{-1}} f\left(\sum(L)\right) \oplus f\left(\sum(R)\right)$$

$$||$$

$$\textstyle\sum(S)$$

$\boxed{\text{Step 2}}$

$$\sum(L), \sum(R) \xrightarrow{f}$$



$$z = ix + j \xrightarrow{f} (i, j) \in [\![0:k]\!] \times [\![0:\ell k]\!]$$

Higher # dimensions but shorter intervals

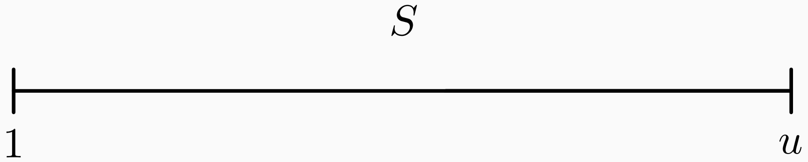
$$\text{FFT} \Downarrow \text{ in } \widetilde{O}(\ell k^2)$$

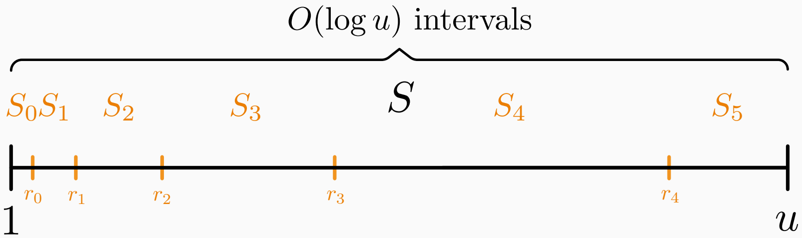
$$\sum(L \cup R) \xleftarrow{f^{-1}} f\left(\sum(L)\right) \oplus f\left(\sum(R)\right)$$

$$\sum(L \cup R) = \sum(S)$$

and if $k = \left\lfloor \frac{u}{x} \right\rfloor \implies \widetilde{O}\left(\left(\frac{u}{x}\right)^2 \ell\right)$.

THE ALGORITHM

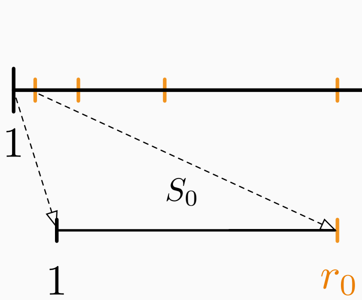


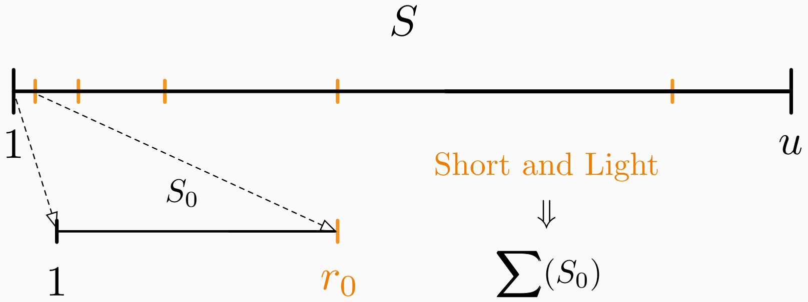


$$S_0 = S \cap [\![1 : r_0]\!]$$
$$S_i = S \cap [\![r_{i-1} + 1 : r_i]\!]$$
$$|S_i| = n_i$$

$r_0 \geq 1$

$r_i = \lfloor 2^i r_0 \rfloor$


S
1
u
S_0
1
r_0



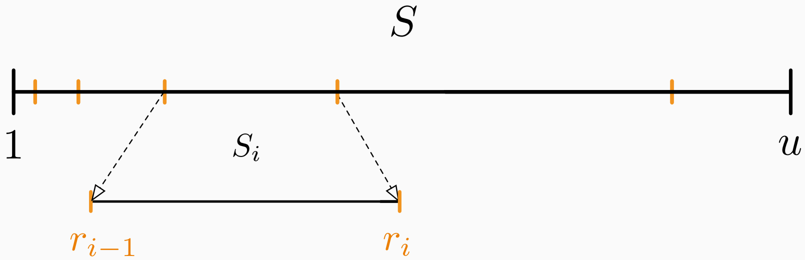
Short and Light

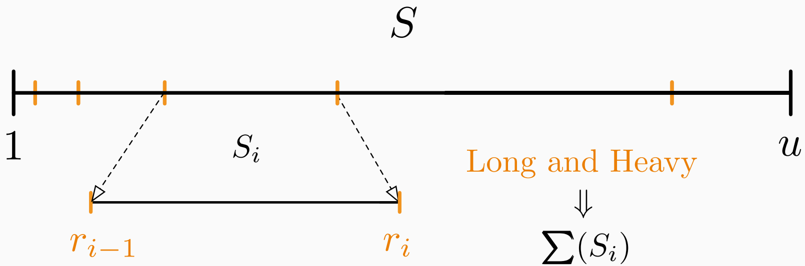
$$\Downarrow$$

$$\sum(S_0)$$

in

$$\widetilde{O}(n_0 r_0) = \widetilde{O}\left(\min\{n, r_0\} r_0\right)$$


S
1
u
S_i
r_{i-1}
r_i



Long and Heavy

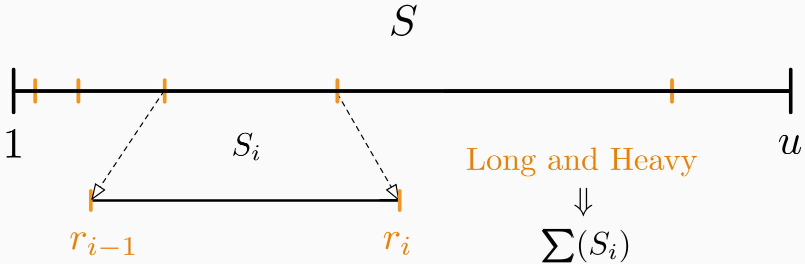
$\Downarrow$

$\sum(S_i)$

in

$$\widetilde{O}\left(\left(\frac{u}{r_{i-1}}\right)^2 \ell_i\right) = \widetilde{O}\left(\frac{u^2}{r_{i-1}}\right)$$

THE ALGORITHM



Long and Heavy

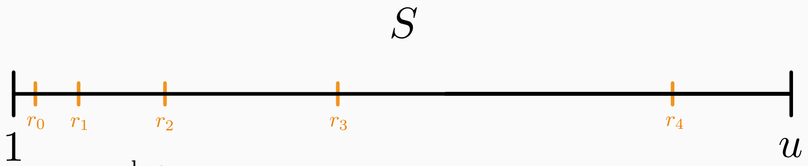
$$\Downarrow$$

$$\sum(S_i)$$

in

$$\widetilde{O}\left(\left(\frac{u}{r_{i-1}}\right)^2 \ell_i\right) = \widetilde{O}\left(\frac{u^2}{r_{i-1}}\right)$$

$$\implies \widetilde{O}\left(\frac{u^2}{r_0}\right) \text{ for all intervals}$$



$$\bigoplus_{i=0}^{\log u} \sum(S_i) \cap [\![1 : u]\!] = \sum(S) \cap [\![1 : u]\!]$$

$$\Downarrow$$

$$O\left(\log u(u \log u)\right) = \widetilde{O}(u)$$

$$S \rightsquigarrow \sum(S) \cap [\![1 : u]\!]$$

In total time:

$$\widetilde{O}\left(\min\{n, r_0\} r_0 + \frac{u^2}{r_0} + u\right)$$

$r_0 = \dfrac{u}{\sqrt{n}}$

$r_0 = u^{2/3}$

$\widetilde{O}\left(\sqrt{n}u\right)$

$\widetilde{O}\left(u^{4/3}\right)$

The second algorithm solves the **all** subset sums problem *modulo m* in finite cyclic groups; i.e., it returns all realizable sums within the group.

The second algorithm solves the **all** subset sums problem *modulo m* in finite cyclic groups; i.e., it returns all realizable sums within the group.

The second algorithm runs in $\widetilde{O}(\sqrt{n}m)$ time, where m is the order of the group.

The second algorithm solves the **all** subset sums problem *modulo m* in finite cyclic groups; i.e., it returns all realizable sums within the group.

The second algorithm runs in $\widetilde{O}(\sqrt{n}m)$ time, where m is the order of the group.

Not trivial!

The challenge is that the previous algorithm throws away many sums that fall outside of $[\![1 : u]\!]$ during its execution, but this can no longer be done for finite cyclic groups, since these sums stay in the group and as such must be accounted for.

FUTURE WORK

Can we use these observations on other similar number-based problems?

OPEN PROBLEMS

Can we use these observations on other similar number-based problems?

Is there a deterministic $\widetilde{O}(t)$ time algorithm for the subset sum problem matching its conditional lower bound?

Thank you