Fine-Grained Complexity - Hardness in P

Lecture 3: 3SUM

Karl Bringmann
Landscape of Polytime Problems

SETH-hard

\( \text{SAT} \ 2^n \)

- \( k\text{-DomSet} \ n^k \) [Patrascu, Williams’10]
- \( \text{OV} \ n^2 \) [Williams’05]
- \( \text{Frechet} \ n^2 \) [V-Williams, Roditty’13]
- \( \text{Diameter} \ n^2 \) [B,Künnemann’14, Abboud, Backurs, V-Williams’15]
- \( \text{Longest Palindromic Subsequence} \ n^2 \) [B,Künnemann’15]
- \( \text{SubsetSum} \ n + t \) [Abboud,B,Hermelin, Shabtay’17+]
- \( \text{Edit Distance} \ n^2 \) [B,Künnekmann’15, Abboud, Backurs, V-Williams’15]
- \( \text{Dynamic Time Warping} \ n^2 \) [Impagliazzo]
- \( \text{RegExp Matching} \ n^2 \) [Backurs, Indyk’16]
- \( \text{Maximum Submatrix} \ n^3 \) [B,Gawrychowski, Mozes, Weimann’18]
- \( \text{Tree Edit Distance} \ n^3 \) [B,Gawrychowski, Mozes, Weimann’18]
- \( \text{NegTriangle} \ n^3 \) [B,Gawrychowski, Mozes, Weimann’18]

3SUM-hard

\( \text{3SUM} \ n^2 \)

- \( X+Y \ n^2 \) [Gajentaan, Overmars’95]
- \( \text{GeomBase} \ n^2 \)
- \( \text{Colinear} \ n^2 \)
- \( \text{Separator} \ n^2 \)
- \( \text{Planar Motion Planning} \ n^2 \)

APSP-hard

\( \text{APSP} \ n^3 \)

- \( \text{Radius} \ n^3 \) [V-Williams, Williams’10]
- \( \text{Metricity} \ n^3 \)
- \( \text{Betweenness Centrality} \ n^3 \)
- \( \text{Tree Edit Distance} \ n^3 \) [B,Gawrychowski, Mozes, Weimann’18]
- \( \text{NegTriangle} \ n^3 \) [B,Gawrychowski, Mozes, Weimann’18]
Problem 3SUM: Given integers $a_1, ..., a_n, b_1, ..., b_n, c_1, ..., c_n$ are there $i, j, k$ such that $a_i + b_j = c_k$?

Algorithms: Naïve: $O(n^3)$
Well-known: $O(n^2)$

3SUM-Hypothesis: $\forall \varepsilon > 0$: 3SUM has no $O(n^{2-\varepsilon})$-time algorithm

We assume that we can add/subtract/compare input integers in constant time.

Can assume that the $a_i, b_j, c_k$ are distinct and from some universe $\{1, ..., U\}$

Proof: Set $M$ such that $|a_i|, |b_j|, |c_k| < M$ for all $i, j, k$

Add: $2M$ to every $a_i$
$4M$ to every $b_j$
$6M$ to every $c_k$

Resulting instance is equivalent, has distinct input numbers, and universe $\{1, ..., 7M\}$
Problem 3SUM: Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ are there $i, j, k$ such that $a_i + b_j = c_k$?

Algorithms: Naïve: $O(n^3)$
Well-known: $O(n^2)$

3SUM-Hypothesis: $\forall \varepsilon > 0$: 3SUM has no $O(n^{2-\varepsilon})$-time algorithm

[Gajentaan, Overmars'95]

$O(n^2 \log n)$-time algorithm:
- sort $c_1 \leq \cdots \leq c_n$
- for each $i, j$:
  - binary search for $a_i + b_j$ among $c_1 \leq \cdots \leq c_n$

$O(n^2)$-time randomized algorithm:
- put each $c_k$ into a hashmap
- for each $i, j$:
  - check whether $a_i + b_j$ is in the hashmap
**Quadratic Algorithm**

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

- if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
- if $a_i + b_j > c_k$: $i := i - 1$
- if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>...</th>
<th>$a_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \ldots \leq a_n$, $b_1 \leq \ldots \leq b_n$, $c_1 \leq \ldots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:
    if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
    if $a_i + b_j > c_k$: $i := i - 1$
    if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$

if $a_i + b_j > c_k$: $i := i - 1$

if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”
Quadratic Algorithm

Given integers $a_1, ..., a_n, b_1, ..., b_n, c_1, ..., c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

  if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
  if $a_i + b_j > c_k$: $i := i - 1$
  if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”
**Quadratic Algorithm**

Given integers \( a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n \)
are there \( i, j, k \) such that \( a_i + b_j = c_k \)?

**sort in increasing order:** \( a_1 \leq \cdots \leq a_n, \ b_1 \leq \cdots \leq b_n, \ c_1 \leq \cdots \leq c_n \)

**for each** \( c_k \):  check whether there are \( i, j \) s.t. \( a_i + b_j = c_k \)

initialize \( i = n, j = 1 \)

while \( i > 0 \) and \( j \leq n \):
    if \( a_i + b_j = c_k \):  return \((a_i, b_j, c_k)\)
    if \( a_i + b_j > c_k \):  \( i := i - 1 \)
    if \( a_i + b_j < c_k \):  \( j := j + 1 \)

return “no solution”
Quadratic Algorithm

Given integers \( a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n \)
are there \( i, j, k \) such that \( a_i + b_j = c_k \)?

sort in increasing order: \( a_1 \leq \cdots \leq a_n, \ b_1 \leq \cdots \leq b_n, \ c_1 \leq \cdots \leq c_n \)

for each \( c_k \): check whether there are \( i, j \) s.t. \( a_i + b_j = c_k \)

initialize \( i = n, j = 1 \)

while \( i > 0 \) and \( j \leq n \):
    if \( a_i + b_j = c_k \): return \((a_i, b_j, c_k)\)
    if \( a_i + b_j > c_k \): \( i := i - 1 \)
    if \( a_i + b_j < c_k \): \( j := j + 1 \)

return “no solution”
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
if $a_i + b_j > c_k$: $i := i - 1$
if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n, \ b_1 \leq \cdots \leq b_n, \ c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

  if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
  if $a_i + b_j > c_k$: $i := i - 1$
  if $a_i + b_j < c_k$: $j := j + 1$

return "no solution"
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:
    if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
    if $a_i + b_j > c_k$: $i := i - 1$
    if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”
Quadratic Algorithm

Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$
are there $i, j, k$ such that $a_i + b_j = c_k$?

sort in increasing order: $a_1 \leq \cdots \leq a_n$, $b_1 \leq \cdots \leq b_n$, $c_1 \leq \cdots \leq c_n$

for each $c_k$: check whether there are $i, j$ s.t. $a_i + b_j = c_k$

initialize $i = n, j = 1$

while $i > 0$ and $j \leq n$:

  if $a_i + b_j = c_k$: return $(a_i, b_j, c_k)$
  if $a_i + b_j > c_k$: $i := i - 1$
  if $a_i + b_j < c_k$: $j := j + 1$

return “no solution”

time $O(n)$ per $c_k$
time $O(n^2)$ overall
Landscape of Polytime Problems

SETH-hard

- SAT $2^n$
- SubsetSum $n + t$
- OV $n^2$
- Dynamic Time Warping $n^2$
- NFA-Acceptance $n^2$
- RegExp Matching $n^2$
- Longest Palindromic Subsequence $n^2$
- Diameter $n^2$
- Frechet $n^2$
- k-DomSet $n^k$
- [Patrascu, Williams’10]
- [Williams’05]
- [B’14]
- [V-Williams, Roditty’13]
- [B, Künemann’15, Abboud, Backurs, V-Williams’15]
- [B, Künemann’15]
- [Impagliazzo]
- [Backurs, Indyk’15]
- [Backurs, Indyk’16]
- [Backurs, Dikkala, Tzamos’16]
- [B, Gawrychowski, Mozes, Weimann’18]
- [Backurs, Indyk’15]
- [B, Künemann’15, Abboud, Backurs, V-Williams’15]
- [V-Williams, Roditty’13]
- [B, Künemann’15]
- [Backurs, Indyk’16]
- [Impagliazzo]
Example: GeomBase

given a set of $n$ points on three horizontal lines $y = 0, y = 1, y = 2$, determine whether there exists a non-horizontal line containing three of the points

**Thm:** GeomBase is 3SUM-hard.

Given an instance $(A, B, C)$ of 3SUM construct points:

- $(a, 0)$ for any $a \in A$
- $(b, 2)$ for any $b \in B$
- $(c/2, 1)$ for any $c \in C$

they lie on a line if $c/2 - a = b - c/2 \iff a + b = c$
Example Planar Motion Planning

**Thm:** PlanarMotionPlanning is 3SUM-hard.
Landscape of Polytime Problems

3SUM-hard

- 3SUM $n^2$
  - X+Y $n^2$
  - Colinear $n^2$
  - Conv3SUM $n^2$
  - GeomBase $n^2$
  - Separator $n^2$
  - PlanarMotion Planning $n^2$

- ZeroWeight Triangle $n^3$
  - Jumbled Indexing

Crucial new ingredient:

- Triangle Enumeration
- Set Disjointness
- Dynamic Shortest Path

Several dynamic and data structure problems

- [Gajentaan, Overmars'95]
- [Patrascu'10]
- [Kopelowitz, Pettie, Porat'14]
- [V-Williams, Williams'13]
- [Amir, Chan, Lewenstein, Lewenstein'14]
I. Equivalence of 3SUM and Conv3SUM

II. Subset Sum

III. Further Topics

IV. Conclusion
Equivalence of 3SUM and Conv3SUM

3SUM: given integers $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$ are there $i, j, k$ such that $a_i + b_j = c_k$?

Conv3SUM: given integers $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$ are there $i, j, k$ with $i + j = k$ such that $a_i + b_j = c_k$?

Thm: [Patrascu’10,Kopelowitz,Pettie,Porat’14]
1) If 3SUM is in time $T(n)$ then Conv3SUM is in time $O(T(n))$
2) If Conv3SUM is in time $T(n)$ then 3SUM is in randomized time $O(T(n))$, with one-sided error probability $\leq 1/2$

(Standard boosting yields any constant error probability $\delta > 0$)

All hypotheses are also for randomized algorithms!
From Conv3SUM to 3SUM

**Thm:** 1) If 3SUM is in time $T(n)$ then Conv3SUM is in time $O(T(n))$

Given input $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$ for Conv3SUM, construct:

$$a'_i := a_i \cdot 3n + i \quad b'_j := b_j \cdot 3n + j \quad c'_k := c_k \cdot 3n + k$$

This is a YES-instance for 3SUM iff:

$$\exists i, j, k: \ a'_i + b'_j - c'_k = 0$$

$$\iff \exists i, j, k: \ 3n \cdot (a_i + b_j - c_k) + (i + j - k) = 0$$

divisible by $3n$ iff $i + j = k$

$$\iff \exists i, j, k: \ i + j = k \ and \ a_i + b_j = c_k$$

"3SUM can simulate multiple linear equations"
Equivalent Variants of Conv3SUM

Given integers $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$
are there $i, j, k$ with $i + j = k$ such that $a_i + b_j = c_k$?

$$a'_0, ..., a'_{2n-1} = a_0, ..., a_{n-1}, \infty, ..., \infty$$
$$b'_0, ..., b'_{2n-1} = b_0, ..., b_{n-1}, \infty, ..., \infty$$
$$c'_0, ..., c'_{2n-1} = c_0, ..., c_{n-1}, \infty, ..., \infty$$

Assume $a_i, b_j, c_k$ take values in $\{1, \ldots, U\}$
Use $10 \cdot U$ as $\infty$

Given integers $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$
are there $i, j, k$ with $k = (i + j) \mod n$ such that $a_i + b_j = c_k$?
Equivalent Variants of Conv3SUM

Given integers \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1} \)
are there \( i, j, k \) with \( i + j = k \) such that \( a_i + b_j = c_k \)?

\[
\begin{align*}
  a_0', \ldots, a_{2n-1}' &= a_0, \ldots, a_{n-1}, \infty, \ldots, \infty \\
  b_0', \ldots, b_{2n-1}' &= b_0, \ldots, b_{n-1}, \infty, \ldots, \infty \\
  c_0', \ldots, c_{2n-1}' &= c_0, \ldots, c_{n-1}, c_0, \ldots, c_{n-1}
\end{align*}
\]

Given integers \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1} \)
are there \( i, j, k \) with \( k = (i + j) \mod n \) such that \( a_i + b_j = c_k \)?
Equivalent Variants of Conv3SUM

3SUM: \( \exists i, j, k: a_i + b_j = c_k? \)

Conv-3SUM: \( \exists i + j = k: a_i + b_j = c_k? \)

Given integers \( a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1} \)
are there \( i, j, k \) with \( i + j = k \) such that \( a_i + b_j = c_k? \)

Given integers \( a_1, ..., a_n \)
are there \( i, j, k \) with \( i + j = k \) such that \( a_i + a_j + a_k = 0? \)

Given integers \( a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1} \)
are there \( i, j, k \) with \( k = (i + j) \mod n \) such that \( a_i + b_j = c_k? \)
From 3SUM to Conv3SUM

**Thm:** 2) If Conv3SUM is in time $T(n)$ then 3SUM is in randomized time $O(T(n))$

Given input $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, c_0, ..., c_{n-1}$ for 3SUM

$A := \{a_0, ..., a_{n-1}\}, B := \{b_0, ..., b_{n-1}\}, C := \{c_0, ..., c_{n-1}\}$,

$\Omega := A \cup B \cup C$

(Can assume that input numbers are distinct)

Assume *magic* hash function $h: \Omega \to \{0, \ldots, R - 1\}$ s.t.

**Linearity:** $h(x + y) = (h(x) + h(y)) \mod R$ for all $x, y \in \Omega$

**No overfull buckets:** $|\{x \in \Omega \mid h(x) = r\}| \leq 100n/R$ for all $r \in \{0, \ldots, R - 1\}$
From 3SUM to Conv3SUM

Thm: 2) If Conv3SUM is in time $T(n)$ then 3SUM is in randomized time $O(T(n))$

Given input $A, B, C$ for 3SUM, compute:

For any $x, y, z \in \{1, ..., 100n/R\}$:

- $a'_i :=$ the $x$-th element of bucket $\{a \in A \mid h(a) = i\}$
- $b'_j :=$ the $y$-th element of bucket $\{b \in B \mid h(b) = j\}$
- $c'_k :=$ the $z$-th element of bucket $\{c \in C \mid h(c) = k\}$

If $a'_0, ..., a'_{R-1}, b'_0, ..., b'_{R-1}, c'_0, ..., c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

Running Time: $O\left((n/R)^3 \cdot T(R)\right)$

Setting $R := n$ we obtain time $O(T(n))$ for 3SUM
From 3SUM to Conv3SUM

Thm: 2) If Conv3SUM is in time $T(n)$ then 3SUM is in randomized time $O(T(n))$

3SUM: $\exists i, j, k: a_i + b_j = c_k$?
Conv-3SUM: $k = (i + j) \mod n$
a_i + b_j = c_k?

$h: \Omega \rightarrow \{0, \ldots, R-1\}$
Linearity: $h(x + y) = h(x) + h(y) \mod R$
No overfull buckets: $|\{x \in \Omega | h(x) = r\}| \leq 100n/R$

or $\infty$, if there are less elements in the bucket

Given input $A, B, C$ for 3SUM, compute:

For any $x, y, z \in \{1, \ldots, 100n/R\}$:

$a'_i :=$ the $x$-th element of bucket $\{a \in A | h(a) = i\}$
$b'_j :=$ the $y$-th element of bucket $\{b \in B | h(b) = j\}$
$c'_k :=$ the $z$-th element of bucket $\{c \in C | h(c) = k\}$

If $a'_0, \ldots, a'_{R-1}, b'_0, \ldots, b'_{R-1}, c'_0, \ldots, c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

Correctness I:

Any Conv3SUM-solution $a'_i + b'_j = c'_k$ has $a'_i \in A, b'_j \in B, c'_k \in C$ (not $\infty$!) and thus yields a solution for 3SUM

Thus, if $A, B, C$ has no solution, then we always return NO
From 3SUM to Conv3SUM

Thm: 2) If Conv3SUM is in time $T(n)$ then 3SUM is in randomized time $O(T(n))$

Given input $A, B, C$ for 3SUM, compute:

For any $x, y, z \in \{1, \ldots, 100n/R\}$:
- $a'_i :=$ the $x$-th element of bucket $\{a \in A \mid h(a) = i\}$
- $b'_j :=$ the $y$-th element of bucket $\{b \in B \mid h(b) = j\}$
- $c'_k :=$ the $z$-th element of bucket $\{c \in C \mid h(c) = k\}$

If $a'_0, \ldots, a'_{R-1}, b'_0, \ldots, b'_{R-1}, c'_0, \ldots, c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

Correctness II: If $A, B, C$ has a solution $a \in A, b \in B, c \in C$ with $a + b = c$, then

Set $i := h(a), j := h(b), k := h(c)$

For some $x, y, z \in \{1, \ldots, 100n/R\}$ we have $a'_i = a, b'_j = b, c'_k = c$ $\Rightarrow$ $a'_i + b'_j = c'_k$

And $k = h(c) = h(a + b) = (h(a) + h(b)) \mod R = (i + j) \mod R$

Thus, if $A, B, C$ has a solution, then we always return YES
Now Without Magic

Want: almost-linear random hash function $h$

$Ω \subseteq \{1, \ldots, U\}$

fix any prime $p > 2U$

pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

$h(x) := (m \cdot x \mod p) \mod R$

This random hash function $h: \{1, \ldots, U\} \rightarrow \{0, \ldots, R - 1\}$ satisfies:

**Almost-linearity:** there is a set $D$ of offsets, $|D| = O(1)$, s.t.

for all $x, y \in \{1, \ldots, U\}$ there exists $d \in D$ s.t.

$h(x + y) = (h(x) + h(y) + d) \mod R$

**Unlikely overfull buckets:** for any $x \in Ω$:  

$\Pr[x$ is in overfull bucket$] \leq 1/6$

assuming $R \leq n$

$|\{y \in Ω \mid h(y) = h(x)\}| > 100n/R$

---

3SUM:  

$\exists i, j, k: \quad a_i + b_j = c_k$?

Conv-3SUM:  

$\exists i, j, k: \quad k = (i + j) \mod n$  

$a_i + b_j = c_k$?

$h: Ω \rightarrow \{0, \ldots, R - 1\}$

**Linearity:**

$h(x + y) = h(x) + h(y) \mod R$

**No overfull buckets:**  

$|\{x \in Ω \mid h(x) = r\}| \leq 100n/R$
Now Without Magic

Want: almost-linear random hash function $h$

$\Omega \subseteq \{1, ..., U\}$

fix any prime $p > 2U$

pick $m \in \{1, ..., p - 1\}$ uniformly at random

$h(x) := (m \cdot x \mod p) \mod R$

This random hash function $h: \{1, ..., U\} \to \{0, ..., R - 1\}$ satisfies:

**Almost-linearity:** there is a set $D$ of offsets, $|D| = O(1)$, s.t.

for all $x, y \in \{1, ..., U\}$ there exists $d \in D$ s.t.

$h(x + y) = (h(x) + h(y) + d) \mod R$

**Unlikely overfull buckets:** for any $x \in \Omega$: $\Pr[\text{x in overfull bucket}] \leq 1/6$

assuming $R \leq n$

$|\{y \in \Omega \mid h(y) = h(x)\}| > 100n/R$

---

3SUM: $\exists i, j, k: a_i + b_j = c_k$?

Conv-3SUM: $k = (i + j) \mod n$ \quad $a_i + b_j = c_k$?

$h: \Omega \to \{0, ..., R - 1\}$

Almost-linearity: $h(x + y) = (h(x) + h(y) + d) \mod R$

for some $d \in D$, $|D| = O(1)$

Unlikely overfull buckets:

$\Pr[x \text{ in overfull bucket}] \leq 1/6$
Now Without Magic

Adapted algorithm:

Given input $A, B, C$ for 3SUM, compute:

Pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

For any $x, y, z \in \{1, \ldots, 100n/R\}$ and any $d \in D$:

$a'_i :=$ the $x$-th element of bucket $\{a \in A \mid h(a) = i\}$

$b'_j :=$ the $y$-th element of bucket $\{b \in B \mid h(b) = j\}$

$c'_k :=$ the $z$-th element of bucket $\{c \in C \mid h(c) = (k + d) \mod R\}$

If $a'_0, \ldots, a'_{R-1}, b'_0, \ldots, b'_{R-1}, c'_0, \ldots, c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

Running Time: $O\left((n/R)^3 \cdot T(R)\right)$

Setting $R := n$ we obtain a randomized algorithm in time $O(T(n))$ for 3SUM
Now Without Magic

Adapted algorithm:

Given input $A, B, C$ for 3SUM, compute:

Pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

For any $x, y, z \in \{1, \ldots, 100n/R\}$ and any $d \in D$:

$a_i' := \text{the } x\text{-th element of bucket } \{a \in A \mid h(a) = i\}$

$b_j' := \text{the } y\text{-th element of bucket } \{b \in B \mid h(b) = j\}$

$c_k' := \text{the } z\text{-th element of bucket } \{c \in C \mid h(c) = (k + d) \mod R\}$

If $a'_0, \ldots, a'_{R-1}, b'_0, \ldots, b'_{R-1}, c'_0, \ldots, c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

Correctness I:

Any Conv3SUM-solution $a_i' + b_j' = c_k'$ has $a_i' \in A, b_j' \in B, c_k' \in C$ (not $\infty$)

and thus yields a solution for 3SUM

Thus, if $A, B, C$ has no solution, then we always return NO
Now Without Magic

**Adapted algorithm:**

Given input $A, B, C$ for 3SUM, compute:

Pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

For any $x, y, z \in \{1, \ldots, 100n/R\}$ and any $d \in D$:

- $a'_i := \text{the } x\text{-th element of bucket } \{a \in A \mid h(a) = i\}$
- $b'_j := \text{the } y\text{-th element of bucket } \{b \in B \mid h(b) = j\}$
- $c'_k := \text{the } z\text{-th element of bucket } \{c \in C \mid h(c) = (k + d) \mod R\}$

If $a'_0, \ldots, a'_{R-1}, b'_0, \ldots, b'_{R-1}, c'_0, \ldots, c'_{R-1}$ is a YES-instance of Conv3SUM: return YES

Return NO

**Correctness II:** If $A, B, C$ has a solution $a \in A, b \in B, c \in C$ with $a + b = c$, then:

Error event $\mathcal{E}$: $a, b$ or $c$ are in an overfull bucket

By union bound, $\Pr[\mathcal{E}] \leq 3 \cdot 1/6 = 1/2 \quad \Rightarrow \Pr[\overline{\mathcal{E}}] \geq 1/2$

Assume $\overline{\mathcal{E}}$ from now on
Now Without Magic

Adapted algorithm:

Given input $A, B, C$ for 3SUM, compute:

Pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

For any $x, y, z \in \{1, \ldots, 100n/R\}$ and any $d \in D$:

$p_i :=$ the $x$-th element of bucket $\{a \in A \mid h(a) = i\}$

$b_j :=$ the $y$-th element of bucket $\{b \in B \mid h(b) = j\}$

$c_k :=$ the $z$-th element of bucket $\{c \in C \mid h(c) = (k + d) \mod R\}$

If $a_0', \ldots, a_{R-1}', b_0', \ldots, b_{R-1}', c_0', \ldots, c_{R-1}'$ is a YES-instance of Conv3SUM: return YES

Return NO

Correctness II: If $A, B, C$ has a solution $a \in A, b \in B, c \in C$ with $a + b = c$, then:

Let $d \in D$ s.t. $h(a + b) = (h(a) + h(b) + d) \mod R$

Set $i := h(a), j := h(b), k \in \{0, \ldots, R - 1\}$ s.t. $h(c) = (k + d) \mod R$

For some $x, y, z \in \{1, \ldots, 100n/R\}$ we have $a_i' = a, b_j' = b, c_k' = c$  \[\Rightarrow a_i' + b_j' = c_k'\]

And $(k + d) \mod R = h(c) = h(a + b) = (h(a) + h(b) + d) \mod R = (i + j + d) \mod R$

Thus $k = k \mod R = (i + j) \mod R$  \[\Rightarrow\text{We return YES with probability } \geq 1/2\]
Equivalence of 3SUM and Conv3SUM

3SUM: given integers \(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1}\)
are there \(i, j, k\) such that \(a_i + b_j = c_k\)?

Conv3SUM: given integers \(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1}\)
are there \(i, j, k\) with \(i + j = k\) such that \(a_i + b_j = c_k\)?

Thm:

[Patrascu'10,Kopelowitz,Pettie,Porat’14]

1) If 3SUM is in time \(T(n)\) then Conv3SUM is in time \(O(T(n))\)

2) If Conv3SUM is in time \(T(n)\) then 3SUM is in randomized time \(O(T(n))\), with one-sided error probability \(\leq 1/2\)

(Standard boosting yields any constant error probability \(\delta > 0\))
Hashing Analysis

Almost-linearity: there is a set $D$ of offsets, $|D| = O(1)$, s.t.
for all $x, y \in \{1, \ldots, U\}$ there exists $d \in D$ s.t.
$h(x + y) = (h(x) + h(y) + d) \mod R$

Unlikely overfull buckets: for any $x \in \Omega$: \( \Pr[x \text{ is in overfull bucket}] \leq 1/6 \)
assuming $R \leq n$
|\{y \in \Omega \mid h(y) = h(x)\}| > 100n/R

This random hash function $h: \{1, \ldots, U\} \to \{0, \ldots, R - 1\}$ satisfies:

- $\Omega \subseteq \{1, \ldots, U\}$
- fix any prime $p > 2U$
- pick $m \in \{1, \ldots, p - 1\}$ uniformly at random

$h(x) := (m \cdot x \mod p) \mod R$
**Almost-Linearity**

there is a set $D$ of offsets, $|D| = O(1)$, s.t.
for all $x, y \in \{1, ..., U\}$ there exists $d \in D$ s.t.

$$h(x + y) = (h(x) + h(y) + d) \mod R$$

---

**Proof:**

$$h(x + y) = ((m \cdot x + m \cdot y) \mod p) \mod R$$

$$= \left(\left(\left((m \cdot x \mod p) + (m \cdot y \mod p)\right) \mod p\right) \mod R \right.$$

$$\in \{0, ..., 2(p - 1)\}$$

$$= \left(\left((m \cdot x \mod p) + (m \cdot y \mod p) + d\right) \mod R \right.$$

for some $d \in D := \{0, -p\}$

$$= (h(x) + h(y) + d) \mod R$$
Near-Universality

for any $x \neq y$, $x, y \in \{-U, \ldots, U\}$:

$$\Pr[h(x) = h(y)] \leq 4/R$$

---

Proof: $h(x) = h(y) \iff (m \cdot x \mod p) \mod R = (m \cdot y \mod p) \mod R$

$$\iff (m \cdot x \mod p) - (m \cdot y \mod p) = i \cdot R$$

for some $i \in \mathbb{Z}$

$$\in \{-(p-1), \ldots, p-1\} \quad \Rightarrow -p/R < i < p/R$$

take mod $p$: $\Rightarrow ((m \cdot x \mod p) - (m \cdot y \mod p)) \mod p = i \cdot R \mod p$

$$\iff (m \cdot (x - y)) \mod p = i \cdot R \mod p$$

Since $|x|, |y| \leq U < p/2$: $|x - y| < p$

Since $x \neq y$: $(x - y) \mod p \neq 0$

Since $p$ prime: there is inverse $(x - y)^{-1}$
Near-Universality

for any \( x \neq y, \ x, y \in \{-U, ..., U\} \):
\[
\Pr[h(x) = h(y)] \leq 4/R
\]

Proof: \( h(x) = h(y) \iff (m \cdot x \mod p) \mod R = (m \cdot y \mod p) \mod R \)
\[
\iff (m \cdot x \mod p) - (m \cdot y \mod p) = i \cdot R \quad \text{for some } i \in \mathbb{Z}
\]
\[
\in \{-p-1, ..., p-1\} \implies -p/R < i < p/R
\]

take mod \( p \):
\[
\Rightarrow ((m \cdot x \mod p) - (m \cdot y \mod p)) \mod p = i \cdot R \mod p
\]
\[
\iff (m \cdot (x - y)) \mod p = i \cdot R \mod p
\]

multiply by \((x - y)^{-1}\): \[
\Rightarrow m \mod p = i \cdot R \cdot (x - y)^{-1} \mod p
\]
\[
= m
\]

So \( m \) is among the values \( M = \{i \cdot R \cdot (x - y)^{-1} \mod p \mid -p/R < i < p/R\} \)
\( i \neq 0 \)

Thus \( \Pr[h(x) = h(y)] \leq \Pr[m \in M] = \frac{|M|}{p-1} \leq \frac{2p/R}{p-1} \leq 4/R \)
## Unlikely Overfull Buckets

For any $x \in \Omega$: \( \Pr[ x \text{ is in overfull bucket}] \leq \frac{1}{6} \)

\[ |\{y \in \Omega \mid h(y) = h(x)\}| > \frac{100n}{R} \]

### Proof:

Write $S(x) := |\{y \in \Omega \mid h(y) = h(x)\}|$

\[ \mathbb{E}[S(x)] = \sum_{y \in \Omega} \Pr[h(x) = h(y)] = 1 + \sum_{y \in \Omega \setminus \{x\}} \Pr[h(x) = h(y)] \]

(by linearity of expectation)

\[ \leq 1 + \frac{4}{R} \cdot |\Omega| \quad \text{(by near-universality)} \]

\[ \leq 1 + \frac{4}{R} \cdot 3n \leq \frac{13n}{R} \quad \text{(by } R \leq n) \]

Markov’s inequality: \( \Pr[S(x) > t] \leq \frac{\mathbb{E}[S(x)]}{t} \leq \frac{13n/R}{t} \)

In particular: \( \Pr \left[ S(x) > \frac{100n}{R} \right] \leq \frac{13n/R}{100n/R} \leq \frac{1}{6} \)
Landscape of Polytime Problems

3SUM-hard

- 3SUM $n^2$
  - X+Y $n^2$
  - Colinear $n^2$
  - Conv3SUM $n^2$
    - Triangle Enumeration
    - Set Disjointness
    - Dynamic Shortest Path
  - GeomBase $n^2$
    - Separator $n^2$
  - PlanarMotion Planning $n^2$
  - ZeroWeight Triangle $n^3$
  - Jumbled Indexing

crucial new ingredient:

several dynamic and data structure problems

[Patrascu’10]
[Patrascu’10]
[Kopelowitz,Pettie,Porat’14]
[V-Williams,Williams’13]
[Amir,Chan,Lewenstein,Lewenstein’14]
I. Equivalence of 3SUM and Conv3SUM

II. Subset Sum

III. Further Topics

IV. Conclusion
Subset Sum

Given a set $X$ of $n$ positive integers and a target $t$, is there a subset $Y$ of $X$ summing to exactly $t$?

Note: $n \leq t$

Many applications, connections to other problems, educational value...

**Pseudopolynomial** time algorithm by dynamic programming:

$$T[i, s] := T[i - 1, s] \lor T[i - 1, s - x_i]$$

$X = \{x_1, \ldots, x_n\}$

Time $O(nt)$, space $O(t)$
Attempts to break $O(nt)$

Is time $O(nt)$ optimal? Is there an $\tilde{O}(t)$ algorithm?

use basic Word RAM parallelism, word size $w$: $O(nt/w)$ [Pisinger'03]

consider $s := \max X$; we can assume $s \leq t$: $O(ns)$ [Pisinger'99]

recent breakthrough: $\tilde{O}(\sqrt{n} \cdot t)$ [Koiliaris,Xu Arxiv’15/SODA’17]

all previous algorithms are deterministic

Thm: Subset Sum is in randomized time $\tilde{O}(t)$. [B. SODA’17]

one-sided error probability $1/n$, time $O(t \log t \log^5 n)$
\( \tilde{O}(t) \)-Algorithm - Preliminaries

Let \( A, B \) be sets of non-negative integers.

**sumset:** \( A \oplus B := \{ a + b \mid a \in A \cup \{0\}, \ b \in B \cup \{0\} \} \)

**\( t \)-capped sumset:** \( A \oplus_t B := (A \oplus B) \cap \{0, \ldots, t\} \)

**Fact:** \( A \oplus_t B \) can be computed in time \( O(t \log t) \)

**how to use „\( \oplus_t \)“:** \( X \oplus_t X \) contains forbidden sums \( x + x \)

however, for a **partitioning** \( X = X_1 \cup X_2 \):

\( X_1 \oplus_t X_2 \) contains only valid subset sums of \( X \)

**New goal:** compute all valid subset sums: \( \{ \Sigma(Y) \mid Y \subseteq X \} \cap \{0, \ldots, t\} \)

where \( \Sigma(Y) := \sum_{y \in Y} y \)
\( \tilde{O}(t) \)-Algorithm - Step 1: Color Coding

we use color-coding to detect sums of **small** subsets:

\[
\text{ColorCoding}(X, t, k):
\]

for \( r = 1, \ldots, \tilde{O}(\log n) \):

consider a **random** partitioning \( X = X_1 \cup \cdots \cup X_{k^2} \)

compute \( S_r := X_1 \oplus_t \cdots \oplus_t X_{k^2} \)

return \( \bigcup_r S_r \)

for a solution \( Y \), we say that the partitioning splits \( Y \) if \( |Y \cap X_i| \leq 1 \) for all \( i \)

**if the partitioning splits \( Y \) then \( S_r \) contains \( \Sigma(Y) \)**

since we can choose the element in \( Y \cap X_i \) (or 0) in each \( X_i \) to obtain \( \Sigma(Y) \)

\[
\Pr[\text{random partitioning splits } Y] \geq 1/e \text{ by birthday paradox}
\]
correctness w.h.p., running time \( \tilde{O}(t \cdot k^2) \)
\( \tilde{O}(t) \)-Algorithm – Step 2: Recursion

- fix solution \( Y \) of instance \((X, t)\)
- \( S_L = \text{ColorCoding}(X_L, t, \log^2 n) \) contains \( Y_L \) w.h.p.
- \( \Sigma(Y_i) \leq (1 + \varepsilon)t/2 \) for \( \varepsilon = O(1/\log n) \) w.h.p.
  - since either \( \Sigma(Y_S) \leq t/2 \) or \( |Y_S| = \Omega(\log^2 n) \)
- \( |X_i| \leq (1 + \varepsilon)n/2 \) w.h.p.
- recursively solve \((X_1, (1 + \varepsilon)t/2) \rightarrow S_1\)
- recursively solve \((X_2, (1 + \varepsilon)t/2) \rightarrow S_2\)

return \( S_1 \oplus_t S_2 \oplus_t S_L \)

\[
\text{time } T(n, t) \leq \tilde{O}(t) + 2T((1 + \varepsilon)n/2, (1 + \varepsilon)t/2) \\
\approx \tilde{O}(t) + 2T(n/2, t/2) = \tilde{O}(t)
\]
Conditional Lower Bounds

Thm: Subset Sum is in randomized time $\tilde{O}(t)$. [B. SODA'17]

Is time $\tilde{O}(t)$ optimal? Can we prove a conditional lower bound?

Thm: Subset Sum is not in time $t^{1-\varepsilon}2^{o(n)}$ unless SETH fails. [Abboud,B.,Hermelin,Shabtay'17+]

Strong Exponential Time Hypothesis:
$\forall \varepsilon > 0: \exists k: k\text{-SAT is not in time } O(2^{(1-\varepsilon)n})$
Conditional Lower Bounds

**Thm:** Subset Sum is in randomized time $\tilde{O}(t)$. [B. SODA'17]

*Is time $\tilde{O}(t)$ optimal? Can we prove a conditional lower bound?*

**Thm:** Subset Sum is not in time $t^{1-\varepsilon}2^{o(n)}$ unless SETH fails. [Abboud,B.,Hermelin,Shabtay’17+]

**k-Sum problem:** Given set $A$, are there $a_1, \ldots, a_k \in A$ with $a_1 + \cdots + a_k = 0$?

Recall: $k$-Sum is in time $O(n + t \text{ polylog } t)$ and in time $O(n^{[k/2]} \log n)$ (for const. $k$)

**Cor:** $k$-Sum is not in time $t^{1-\varepsilon}n^{o(k)}$ unless SETH fails. [Abboud,B.,Hermelin,Shabtay’17+]
**Ingredient: k-sum-free Sets**

$S \subseteq \{1, \ldots, U\}$ is called $k$-sum-free if $\forall \ell \leq k: \forall x_1, \ldots, x_\ell, x \in S$:

$$x_1 + \cdots + x_\ell = \ell \cdot x \implies x_1 = \cdots = x_\ell = x$$

**Thm:** For any $k, n$ and $\varepsilon > 0$ there exists a $k$-sum-free set $S$ of size $n$ over universe $U = n^{1+\varepsilon}k^{O(1/\varepsilon)}$ [Behrend’46]

**Proof:** $R := \{y \in [b]^r \mid ||y|| = z\}$ is $k$-sum-free since $||\ell \cdot y|| = \ell \cdot z$ but $||y_1 + \cdots + y_\ell|| < \ell \cdot z$ if $y_i \neq y_j$ for some $i, j$

embed $R$ into the integers:

$S := \{\Sigma_{i=1}^r y[i] \cdot (kb)^{i-1} \mid y \in R\}$ is $k$-sum-free, since there is no overflow

$$\begin{array}{c|c|c|c}
0 \ldots 0 & y[r] & \cdots & 0 \ldots 0 & y[2] & 0 \ldots 0 & y[1]
\end{array}$$
**SETH-Hardness of Subset Sum I**

**k-SAT**: $n$ variables, $m$ clauses

- Sparsification lemma [Impagliazzo, Paturi, Zane’01]

**k-SAT**: $n$ variables, $O_{k,\varepsilon}(n)$ clauses,
  each variable appears in $O_{k,\varepsilon}(1)$ clauses

- Block $a$ variables to a supervariables
- Block $O_{k,\varepsilon}(a)$ clauses to a superconstraint

**Constraint Satisfaction Problem:**
  $n' = n/a$ variables over $[2^a]$, $n/a$ constraints
  each variable appears in $\leq d$ clauses
  each constraint touches $\leq d$ variables
  $d = O_{k,\varepsilon}(\text{poly}(a))$

- Time $O(2^{(1-\varepsilon/2)n})$
- Time $O(2^{(1-\varepsilon)n})$
- Time $O(2^{(1-\varepsilon)n'a})$
SETH-Hardness of Subset Sum II

Subset Sum instance:

target $t$:

item $(x, \alpha)$:

for any variable $x$, assignment $\alpha \in [2^a]$

item $(C, \alpha_1, \ldots, \alpha_s)$:

for any constraint $C = C(x_1, \ldots, x_s)$, satisfying assignment $\alpha_1, \ldots, \alpha_s \in [2^a]$

Choose exactly $2n/a$ items for each variable and each clause
**SETH-Hardness of Subset Sum II**

Subset Sum instance:

<table>
<thead>
<tr>
<th>Target $t$:</th>
<th>highest bits</th>
<th>lowest bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2n/a$</td>
<td>0 ... 0</td>
<td>$n/a \cdot \log(2dU + 1)$ bits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item $(x, \alpha)$:</th>
<th>$1$</th>
<th>$0$ ... $0$</th>
</tr>
</thead>
</table>

for any variable $x$, assignment $\alpha \in [2^a]$

<table>
<thead>
<tr>
<th>Item $(C, \alpha_1, ..., \alpha_s)$:</th>
<th>$1$</th>
<th>$0$ ... $0$</th>
</tr>
</thead>
</table>

for any constraint $C = C(x_1, ..., x_s)$, satisfying assignment $\alpha_1, ..., \alpha_s \in [2^a]$

construct $d$-sum-free set $S \subseteq [U]$ of size $2^a$ with $U = 2^{(1+\varepsilon)a}d^{O(1/\varepsilon)}$

write $S = \{S_1, ..., S_{2^a}\}$, construction time $T(a, k, \varepsilon) = O(1)$
# SETH-Hardness of Subset Sum II

**Subset Sum instance:**

<table>
<thead>
<tr>
<th>highest bits</th>
<th>lowest bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 log ( n ) bits</td>
<td>( n/a \cdot \log(2 , dU + 1) ) bits</td>
</tr>
<tr>
<td>( 2n/a )</td>
<td>( 3n/a )</td>
</tr>
</tbody>
</table>

**target \( t \):**

\[ \begin{array}{c}
2n/a \ 0 \ldots 0 \\
\end{array} \]

**item \( (x, \alpha) \):**

for any variable \( x \),
assignment \( \alpha \in [2^a] \)

\[ \begin{array}{c}
1 \ 0 \ldots 0 \\
0010 \ 0 \ldots 0 \ 0 \ldots 0 \\
0 \ldots 0 \ dU - d_xS_\alpha \ 0 \ldots 0 \\
\end{array} \]

\( d_x = \# \text{constraints touching } x \)

**item \( (C, \alpha_1, \ldots, \alpha_s) \):**

for any constraint \( C = C(x_1, \ldots, x_s) \),

satisfying assignment \( \alpha_1, \ldots, \alpha_s \in [2^a] \)

\[ \begin{array}{c}
1 \ 0 \ldots 0 \\
0 \ldots 0 \ 0100 \ 0 \ldots 0 \\
\end{array} \]

\( S_{\alpha_i} \)

- block sum = \( dU \iff \sum S_{\alpha_i} = d_xS_\alpha \iff \text{all assignments for } x \)

are consistent

- no overflow between blocks since \( d_x \leq d \) and \( S_\alpha \leq U \)

- consistent choice of assignments satisfying all constraints
SETH-Hardness of Subset Sum II

Subset Sum instance:

**target** $t$:  

<table>
<thead>
<tr>
<th>highest bits</th>
<th>lowest bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2n/a$</td>
<td>$0 \ldots 0$</td>
</tr>
<tr>
<td>$3n/a$</td>
<td>$1 \ldots 1$ $1 \ldots 1$ $0 \ldots 0$</td>
</tr>
<tr>
<td>$n/a \cdot \log(2 , dU + 1)$</td>
<td>$\ldots$ $dU$ $\ldots$</td>
</tr>
</tbody>
</table>

**item** $(x, \alpha)$:  

| 1 | 0 \ldots 0 |
| 0010 | 0 \ldots 0 | 0 \ldots 0 |
| 0 \ldots 0 | $dU - d_x \, S_\alpha$ | 0 \ldots 0 |

for any variable $x$, assignment $\alpha \in [2^a]$  

**item** $(C, \alpha_1, \ldots, \alpha_s)$:  

| 1 | 0 \ldots 0 |
| 0 \ldots 0 | 0100 | 0 \ldots 0 |
| $\ldots$ | $S_{\alpha_i}$ | $\ldots$ |

for any constraint $C = C(x_1, \ldots, x_s)$, satisfying assignment $\alpha_1, \ldots, \alpha_s \in [2^a]$  

recall:  

$$d = O_{k, \epsilon}(\text{poly}(a))$$  

$$U = 2^{(1+\epsilon)a} \, d^{O(1/\epsilon)}$$  

$$\#\text{bits} = n/a \cdot \log(O(dU))$$  

$$= (1 + \epsilon)n + O_{k, \epsilon}(n \log(a)/a)$$  

$$\leq (1 + 2\epsilon)n$$  

for sufficiently large $a = a(k, \epsilon)$  

$$\#\text{items} = O_{k, \epsilon}(n)$$  

$t^{1-4\epsilon} \, 2^{o(n)}$ algorithm for Subset Sum would break SETH
Conditional Lower Bounds

Thm: Subset Sum is in randomized time $\tilde{O}(t)$. [B. SODA’17]

Is time $\tilde{O}(t)$ optimal? Can we prove a conditional lower bound?

Thm: Subset Sum is not in time $t^{1-\varepsilon}2^{o(n)}$ unless SETH fails. [Abboud,B.,Hermelin,Shabtay’17+]
I. Equivalence of 3SUM and Conv3SUM

II. Subset Sum

III. Further Topics

IV. Conclusion
More Algorithms for 3SUM

trivial: $O(n^3)$
well-known: $O(n^2)$

using Word RAM bit-tricks: $O\left(n^2 \cdot \frac{\log^2 w}{w}\right)$, $O\left(n^2 \cdot \frac{(\log \log n)^2}{\log^2 n}\right)$

(cell size $w = \Omega(\log n)$, each number fits in a cell)

[Baran,Demaine,Patrascu’05]

no bit-tricks: $O\left(n^2 \cdot \frac{(\log \log n)^{O(1)}}{\log^2 n}\right)$

[Gronlund,Pettie’14]
[Gold,Sharir’15]
[Chan’17]

decision tree complexity: $O\left(n^\frac{3}{2} \cdot \sqrt{\log n}\right)$

$O(n \cdot \log^2 n)$

[Gronlund,Pettie’14]

[Kane,Lovett,Moran’18]
Strong 3SUM Hypothesis

using FFT: 3SUM is in time $O(n + U \text{ polylog } U)$ over universe $\{1, \ldots, U\}$

3SUM over any universe $\{1, \ldots, n^c\}$ is equivalent to 3SUM over universe $\{1, \ldots, n^3\}$
via hashing, follows from [Baran, Demaine, Patrascu’05]

Strong 3SUM Hypothesis: 3SUM over universe $\{1, \ldots, n^2\}$ is not in time $O(n^{2-\epsilon})$
(min,+)-Convolution

Problem (min,+)-Convolution:
Given integers $a_1, \ldots, a_n, b_1, \ldots, b_n$, compute $c_0, \ldots, c_{n-1}$ with

$$c_k := \min_{1 \leq i \leq k} a_i + b_{k-i}$$

(min,+)-Conv-Hypothesis:
\( \forall \varepsilon > 0: (\text{min,+})\text{-Conv has no } O(n^{2-\varepsilon})\text{-time algorithm} \)
Current Landscape of Hypotheses

- **OV, \( n^2 \)**
  - [Williams’05]

- **APSP, \( n^3 \)**
  - [V-Williams, Williams’10]

- **SAT, \( 2^n \)**
  - [Abboud, B, Dell, Nederlof’18]

- **3SUM, \( n^2 \)**
  - [Backurs, Indyk, Schmidt’17]

- **SETH**

- **Neg-\( k \)-Clique, \( n^k \)**
  - [Bremner et al.’06]

- **(min,+)‐Conv, \( n^2 \)**
Harder Problems

Triangle Collection $n^3$
- [Abboud, V-Williams, Yu’15]

ZeroWeight Triangle $n^3$
- [V-Williams, Williams’13]

OV, $n^2$
- [Williams’05]

APSP, $n^3$
- [Abboud, B, Dell, Williams’10]

3SUM, $n^2$
- [Backurs, Indyk, Schmidt’17]

SAT, $2^n$

Neg-$k$-Clique, $n^k$
- [Bremner et al.’06]

SETH

ZeroWeight Triangle Collection $n^3$
- [Abboud, V-Williams, Yu’15]

Triangle Collection $n^3$
- [Abboud, V-Williams, Yu’15]

ZeroWeight Triangle $n^3$
- [V-Williams, Williams’13]

OV, $n^2$
- [Williams’05]

APSP, $n^3$
- [Abboud, B, Dell, Williams’10]

3SUM, $n^2$
- [Backurs, Indyk, Schmidt’17]

SAT, $2^n$

Neg-$k$-Clique, $n^k$
- [Bremner et al.’06]

SETH
I. Equivalence of 3SUM and Conv3SUM

II. Subset Sum

III. Further Topics

IV. Conclusion
**Conv3SUM**

- **SAT** $2^n$
  - **SubsetSum** $n + t$
    - **3SUM** $n^2$
      - **X+Y** $n^2$
      - **Colinear** $n^2$
      - **Conv3SUM** $n^2$
        - **Triangle Enumeration**
        - **Set Disjointness**
        - **Dynamic Shortest Path**
        - **Jumbled Indexing**
      - **ZeroWeight Triangle** $n^3$
        - **PlanarMotion Planning** $n^2$
      - **GeomBase** $n^2$
      - **Separator** $n^2$
      - **[Gajentaan, Overmars ’95]**
      - **[Patrascu ’10]**
      - **[Kopelowitz, Pettie, Porat ’14]**

- **SETH**
  - **SAT** $2^n$
    - **SubetSum** $n + t$

**crucial new ingredient:**

- we have seen:
  - $k$-sum-free sets
  - almost-linear hashing
  - further topics, e.g. (min, +)-Conv

**Open:** **Knapsack**: improve time $O(nW)$ to $O(n^2 + W)$?

**Is 3SUM** over universe $\{1, \ldots, n^2\}$ equivalent to 3SUM?