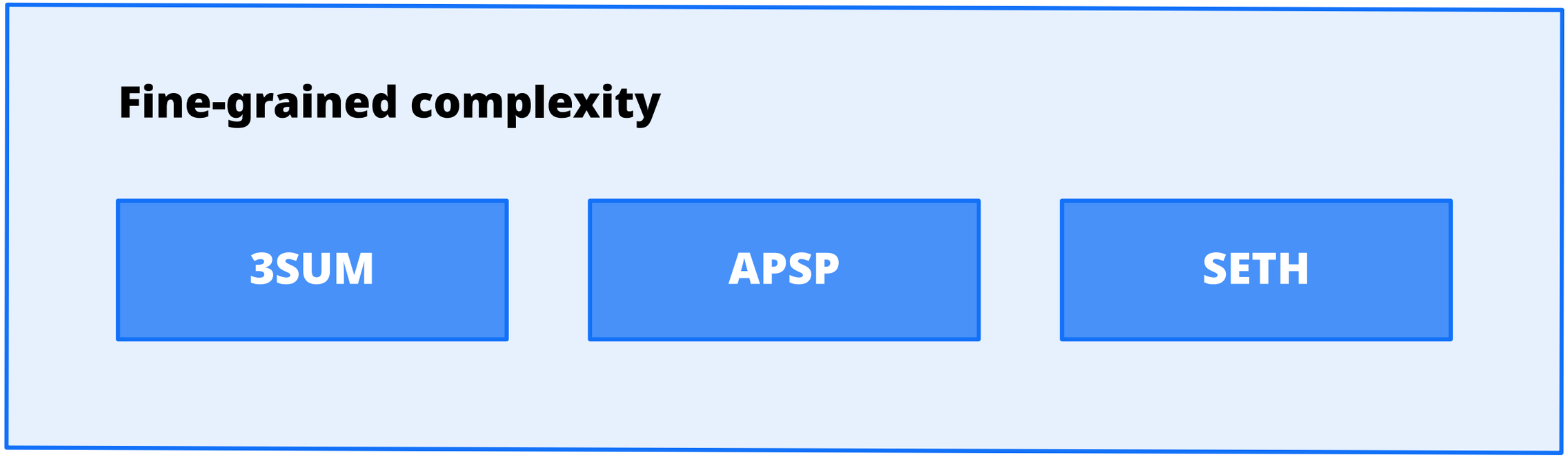
3SUM in Preprocessed Universes

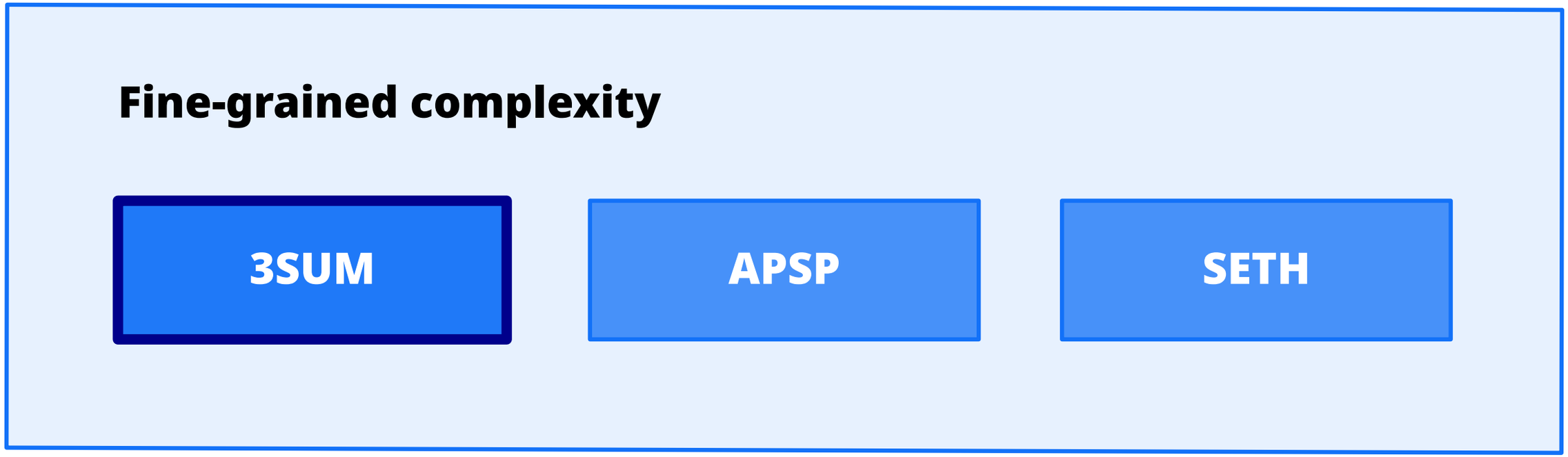
FASTER AND SIMPLER

Shashwat Kasliwal
IIT Delhi

Adam Polak
Bocconi University

Pratyush Sharma
IIT Delhi

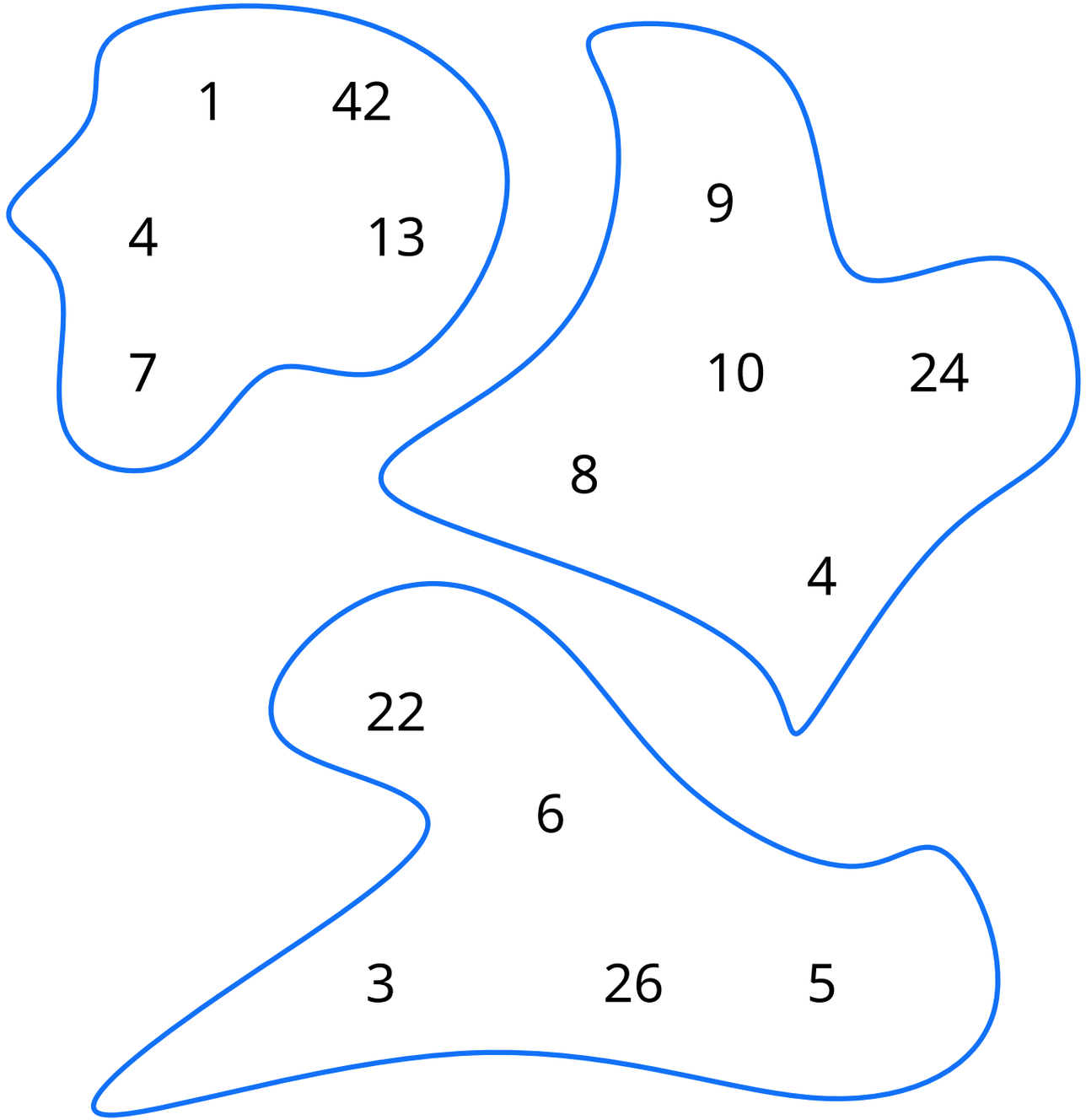

Fine-grained complexity
3SUM
APSP
SETH


Fine-grained complexity
3SUM
APSP
SETH

The 3SUM problem

Input:
Three sets A, B, C of n integers

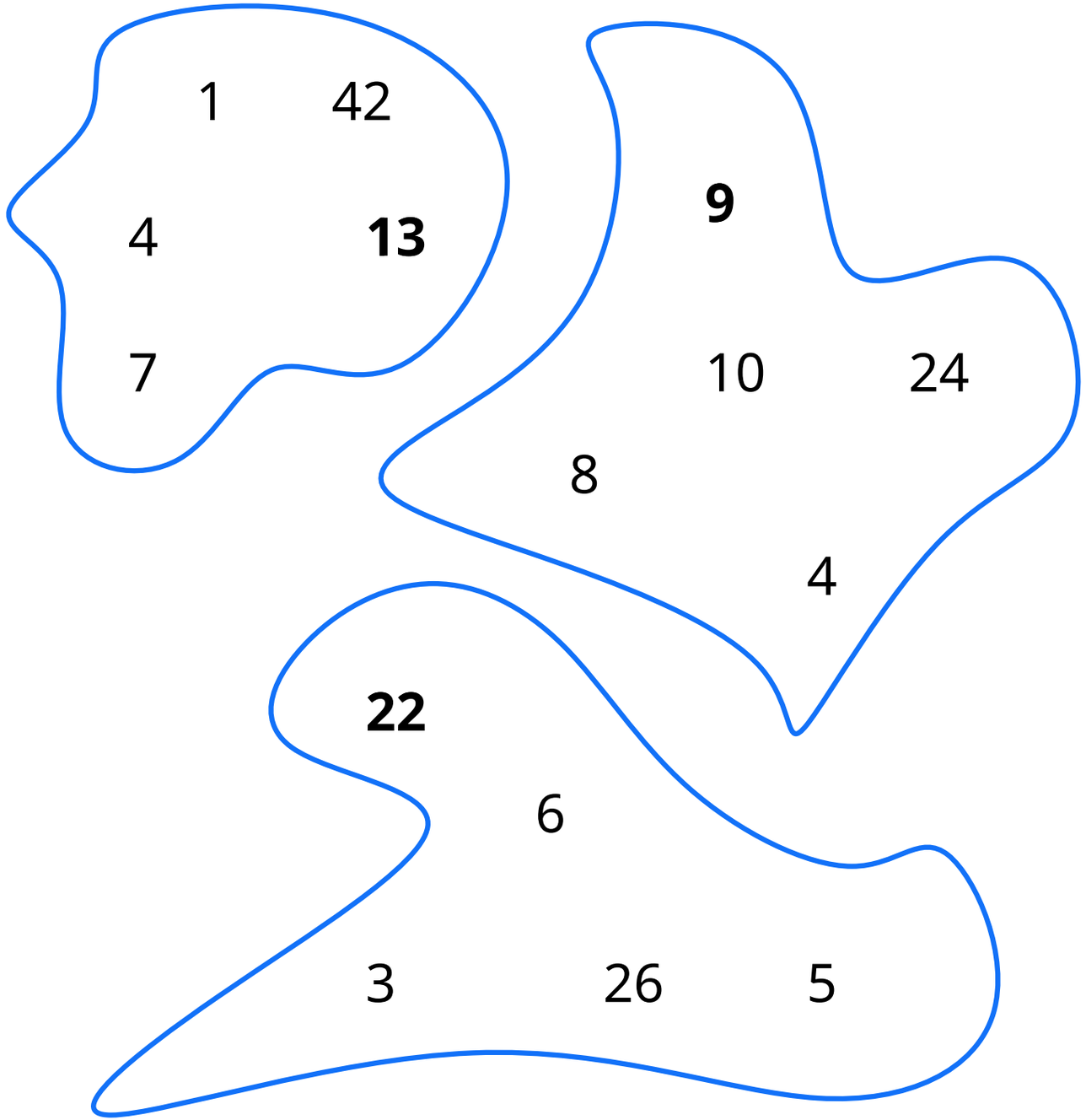
Output:
$\exists\, a \in A, b \in B, c \in C$ with $a + b = c$ **?**



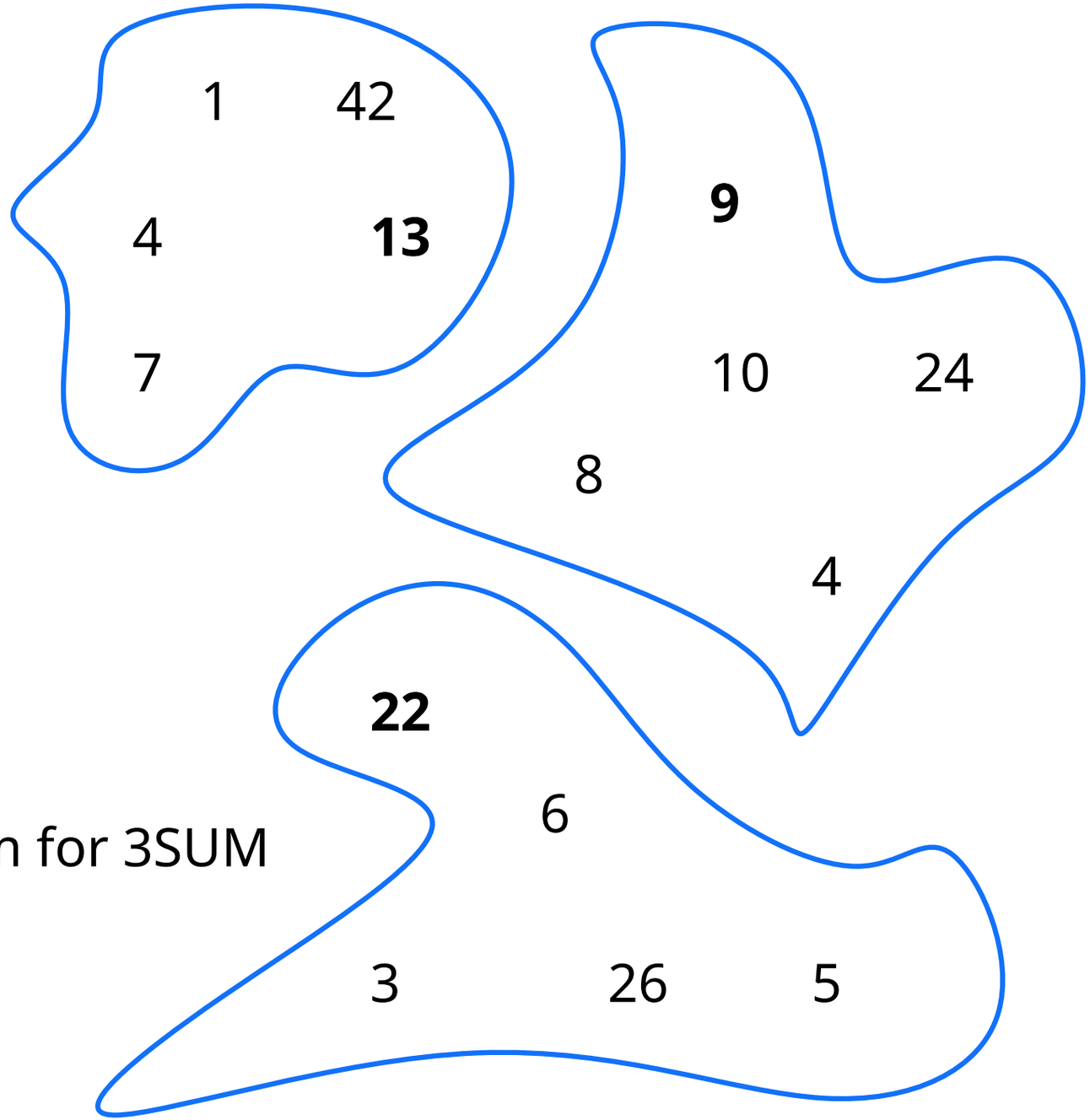
The 3SUM problem

Input:
Three sets A, B, C of n integers

Output:
$\exists\, a \in A, b \in B, c \in C$ with $a + b = c$ **?**



The 3SUM problem



Input:
Three sets A, B, C of n integers

Output:
$\exists\, a \in A, b \in B, c \in C$ with $a + b = c$ **?**

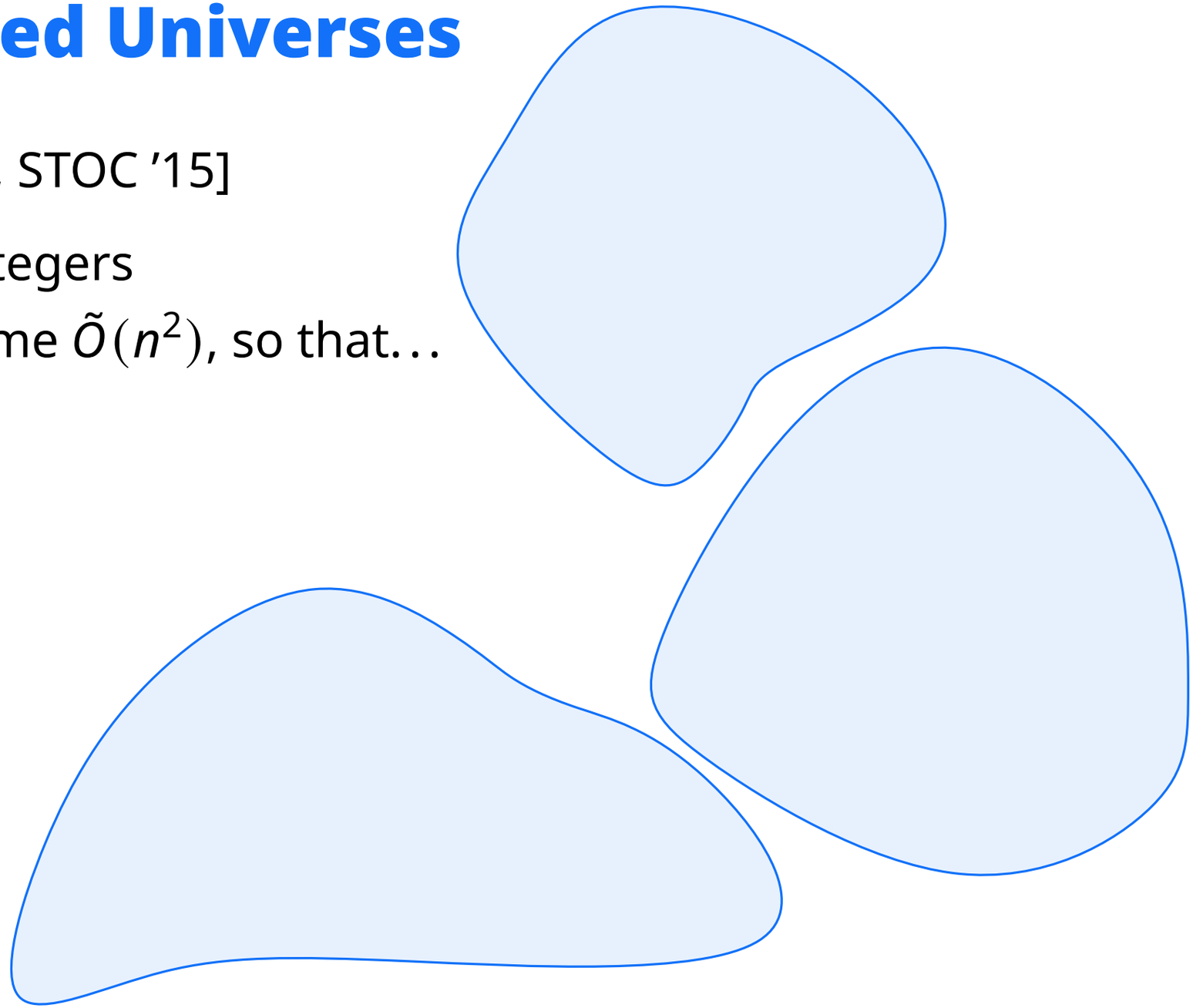
Hypothesis: No $O(n^{1.99})$ time algorithm for 3SUM

3SUM in Preprocessed Universes

Theorem [Chan–Lewenstein, STOC '15]

Given three sets A, B, C of n integers

one can preprocess them, in time $\tilde{O}(n^2)$, so that…



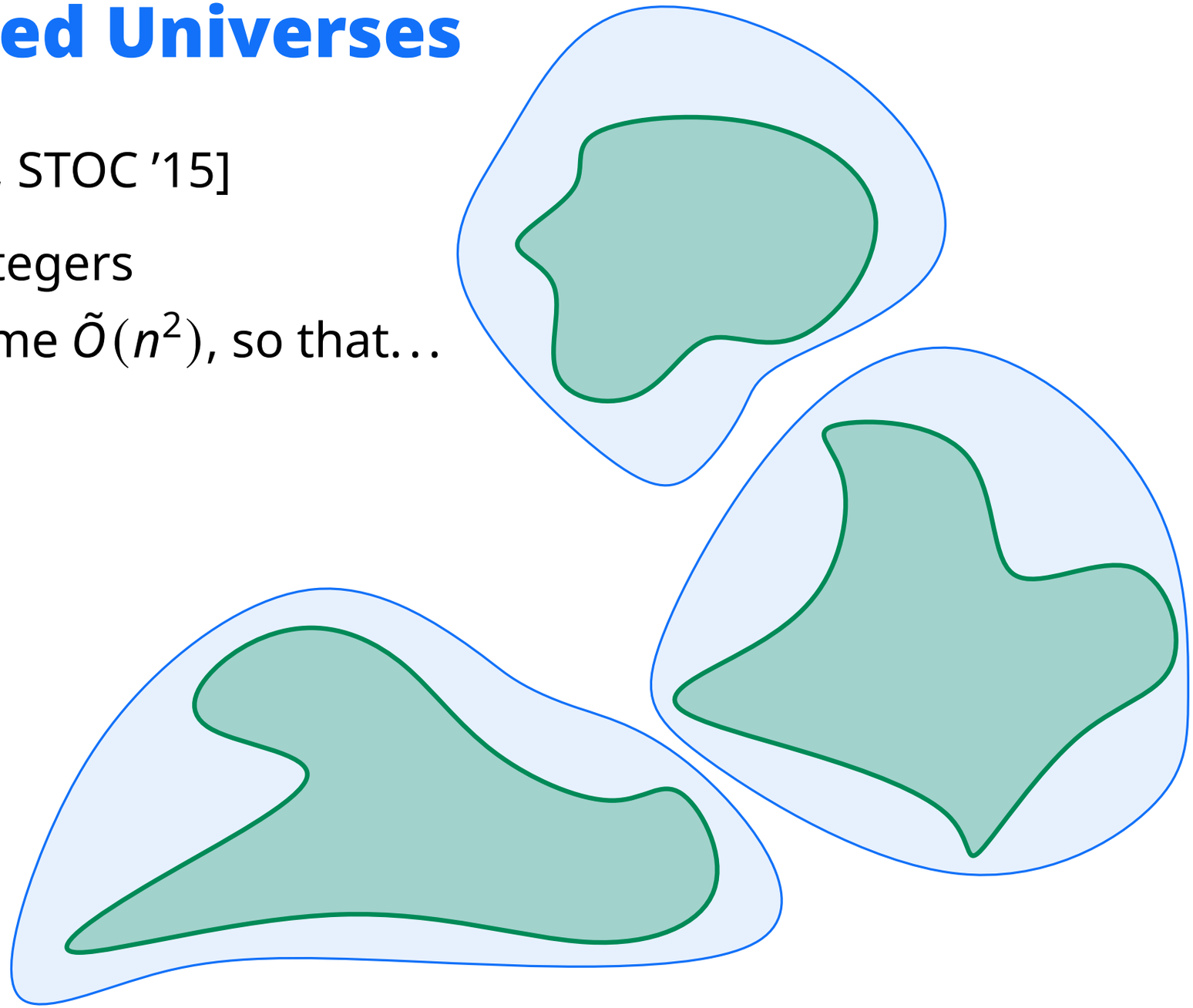
3SUM in Preprocessed Universes

Theorem [Chan–Lewenstein, STOC '15]

Given three sets A, B, C of n integers

one can preprocess them, in time $\tilde{O}(n^2)$, so that…

given $A' \subseteq A$, $B' \subseteq B$ $C' \subseteq C$



3SUM in Preprocessed Universes

Theorem [Chan–Lewenstein, STOC '15]

Given three sets A, B, C of n integers

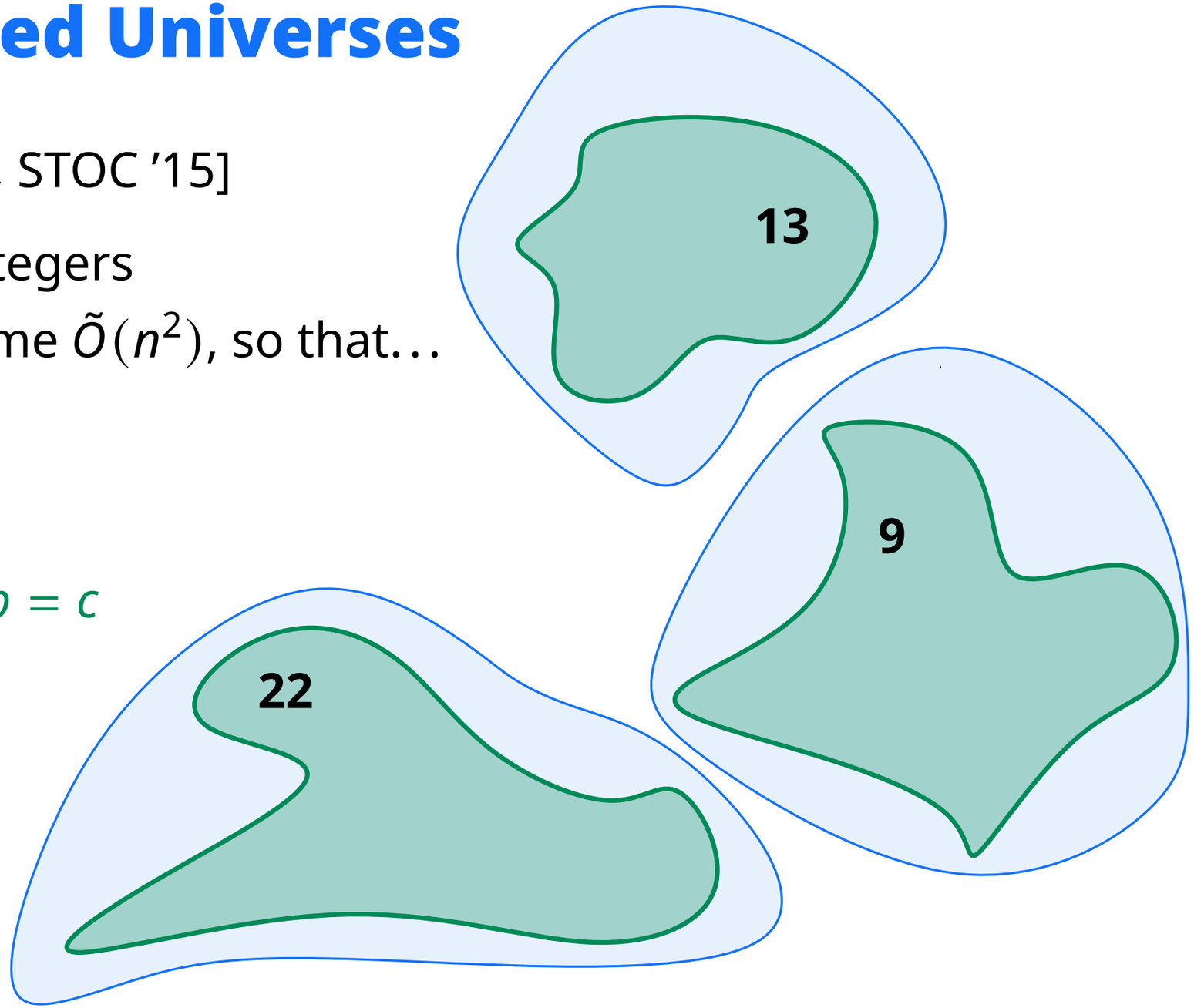
one can preprocess them, in time $\tilde{O}(n^2)$, so that…

given $A' \subseteq A$, $B' \subseteq B$ $C' \subseteq C$

one can decide if there exists

$a \in A'$, $b \in B'$, $c \in C'$ with $a + b = c$

in time $\tilde{O}(n^{13/7})$.



3SUM in Preprocessed Universes

Chan, Lewenstein [STOC '15] $\tilde{O}(n^{13/7})$

3SUM in Preprocessed Universes

Chan, Lewenstein [STOC '15] $\tilde{O}(n^{13/7})$

Chan, Vassilevska Williams, Xu [STOC '23] $\tilde{O}(n^{11/6})$

3SUM in Preprocessed Universes

Chan, Lewenstein [STOC '15]	$\tilde{O}(n^{13/7})$
Chan, Vassilevska Williams, Xu [STOC '23]	$\tilde{O}(n^{11/6})$
This work	$\boldsymbol{O(n^{3/2} \log n)}$

3SUM in Preprocessed Universes

Balog–Szemerédi–Gowers

Chan, Lewenstein [STOC ’15]	$\tilde{O}(n^{13/7})$
Chan, Vassilevska Williams, Xu [STOC ’23]	$\tilde{O}(n^{11/6})$
This work	$\boldsymbol{O(n^{3/2} \log n)}$

3SUM in Preprocessed Universes

Balog–Szemerédi–Gowers

Chan, Lewenstein [STOC '15]	$\tilde{O}(n^{13/7})$
Chan, Vassilevska Williams, Xu [STOC '23]	$\tilde{O}(n^{11/6})$
This work	$\boldsymbol{O(n^{3/2} \log n)}$

elementary

Tool: linear hashing modulo p

$$\{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p\}$$

solutions modulo p

Tool: linear hashing modulo p

$$\{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod p\}$$

solutions modulo p $=$ true solutions $\cup$ false positives

$$\{(a,b,c) \in A \times B \times C : a + b = c\}$$

$$\{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

Our preprocessing

1. Pick a random prime $p \in [n^{1.5}, 2 \cdot n^{1.5})$

2. List all **false positives** **modulo** p

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

Our preprocessing

1. Pick a random prime $p \in [n^{1.5}, 2 \cdot n^{1.5})$

2. List all **false positives** **modulo** p

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

in time $O(n^2 + |F|)$

Observation: $\mathbb{E}[|F|] \leqslant O(n^{1.5} \log n)$

Bounding the number of false positives

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

$$\mathbb{E}[|F|] = \sum_{a+b \neq c} \Pr[a + b \equiv c \bmod p]$$

Bounding the number of false positives

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

$$\mathbb{E}[|F|] = \sum_{a+b\neq c} \Pr[a + b \equiv c \bmod p]$$

$$= \sum_{a+b\neq c} \frac{\text{number of primes dividing } a+b-c}{\text{number of primes we pick from}}$$

Bounding the number of false positives

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

$$\mathbb{E}[|F|] = \sum_{a+b\neq c} \Pr[a + b \equiv c \bmod p]$$

$$= \sum_{a+b\neq c} \frac{\text{number of primes dividing } a+b-c}{\text{number of primes we pick from}}$$

$O(1)$

$\Theta(n^{1.5}/\log(n))$

Bounding the number of false positives

$$F = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

$$\mathbb{E}[|F|] = \sum_{a+b\neq c} \Pr[a + b \equiv c \bmod p]$$

$$= \sum_{a+b\neq c} \frac{\text{number of primes dividing } a+b-c}{\text{number of primes we pick from}}$$

$O(1)$

$\Theta(n^{1.5}/\log(n))$

$$= O\left(n^3 \cdot \frac{1}{n^{1.5}/\log(n)}\right)$$

Bounding the number of false positives

$$F = \{(a, b, c) \in A \times B \times C : a + b \equiv c \bmod p \wedge a + b \neq c\}$$

$$\mathbb{E}[|F|] = \sum_{a+b\neq c} \Pr[a + b \equiv c \bmod p]$$

$$= \sum_{a+b\neq c} \frac{\text{number of primes dividing } a+b-c}{\text{number of primes we pick from}}$$

$O(1)$

$\Theta(n^{1.5}/\log(n))$

$$= O\left(n^3 \cdot \frac{1}{n^{1.5}/\log(n)}\right)$$

$$= O(n^{1.5} \log n) \quad \blacksquare$$

Our algorithm

1. Calculate **number** of **solutions modulo** p, i.e.,

$$\#\left\{(a, b, c) \in A' \times B' \times C' : a + b \equiv c \bmod p\right\}$$

2. Subtract **false positives** present in the **actual instance**, i.e.,

$$F \cap (A' \times B' \times C')$$

Our algorithm

in time $O(p \log p)$, using **FFT**

1. Calculate **number** of **solutions modulo** p, i.e.,

$$\#\{(a, b, c) \in A' \times B' \times C' : a + b \equiv c \bmod p\}$$

2. Subtract **false positives** present in the **actual instance**, i.e.,

$$F \cap (A' \times B' \times C')$$

in time $O(|F|)$

Our algorithm

1. Calculate **number** of **solutions modulo** p, i.e.,

$$\#\{(a,b,c) \in A' \times B' \times C' : a + b \equiv c \bmod p\}$$

in time $O(p \log p)$, using **FFT**

$$Q_A(x) = \sum_{a \in A} x^a$$

e.g.: $Q_{\{2,3,7\}} = x^2 + x^3 + x^7$

$$(Q_A \cdot Q_B)(x) = \sum \alpha_c x^c$$

$$\alpha_c = \#\{(a,b) \in A \times B : a + b = c\}$$

2. Subtract **false positives** present in the **actual instance**, i.e.,

$$F \cap (A' \times B' \times C')$$

in time $O(|F|)$

Our algorithm

in time $O(p \log p)$, using **FFT**

$$Q_A(x) = \sum_{a \in A} x^a$$

e.g.: $Q_{\{2,3,7\}} = x^2 + x^3 + x^7$

$$(Q_A \cdot Q_B)(x) = \sum \alpha_c x^c$$

$$\alpha_c = \#\{(a,b) \in A \times B : a + b = c\}$$

1. Calculate **number** of **solutions modulo** p, i.e.,

$$\#\{(a,b,c) \in A' \times B' \times C' : a + b \equiv c \bmod p\}$$

2. Subtract **false positives** present in the **actual instance**, i.e.,

$$F \cap (A' \times B' \times C')$$

in time $O(|F|)$

Note: Can output for **each** $c \in C'$ if there exist $a \in A'$, $b \in B'$ with $a + b = c$

Derandomizing our preprocessing

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Have: A **random prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ does the job w.h.p.

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Have: A **random prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ does the job w.h.p.

Ergo: There **exists a prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ that does the job

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Have: A **random prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ does the job w.h.p.

Ergo: There **exists a prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ that does the job

Idea: **Check all primes** from $[n^{1.5}, 2 \cdot n^{1.5})$, and select one that does the job

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Have: A **random prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ does the job w.h.p.

Ergo: There **exists a prime** from $[n^{1.5}, 2 \cdot n^{1.5})$ that does the job

Idea: **Check all primes** from $[n^{1.5}, 2 \cdot n^{1.5})$, and select one that does the job

$$\widetilde{O}(n^{1.5} \cdot n^{1.5}) = \widetilde{O}(n^3) \quad :-(

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Better idea:

- **Check all primes** $p_1 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1}| \approx n^{2.5}$

$$\tilde{O}(n^{0.5} \cdot n^{0.5})$$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Better idea:

- **Check all primes** $p_1 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1}| \approx n^{2.5}$
- **Check all primes** $p_2 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2}| \approx n^2$

$$\tilde{O}(n^{1.0} \cdot n^{0.5})$$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Better idea:

- **Check all primes** $p_1 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1}| \approx n^{2.5}$
- **Check all primes** $p_2 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2}| \approx n^2$
- **Check all primes** $p_3 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2 \cdot p_3}| \approx n^{1.5}$

$$\widetilde{O}(n^{1.5} \cdot n^{0.5})$$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Better idea:

- **Check all primes** $p_1 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1}| \approx n^{2.5}$
- **Check all primes** $p_2 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2}| \approx n^2$
- **Check all primes** $p_3 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2 \cdot p_3}| \approx n^{1.5}$
- Set $m = p_1 \cdot p_2 \cdot p_3$

Derandomizing our preprocessing

$$F_m = \{(a,b,c) \in A \times B \times C : a + b \equiv c \bmod m \wedge a + b \neq c\}$$

Need: A **modulus** $m \approx n^{1.5}$ s.t. **# false positives modulo** m is $|F_m| \approx n^3/m$

Better idea:

- **Check all primes** $p_1 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1}| \approx n^{2.5}$
- **Check all primes** $p_2 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2}| \approx n^2$
- **Check all primes** $p_3 \in [n^{0.5}, 2 \cdot n^{0.5})$ **until** $|F_{p_1 \cdot p_2 \cdot p_3}| \approx n^{1.5}$
- Set $m = p_1 \cdot p_2 \cdot p_3$

[Chan–He '20, Fischer–Kaliciak–P. '24]

What else we can do?

- No need to know the set C during preprocessing

What else we can do?

- No need to know the set C during preprocessing

What we cannot do? (a.k.a. open problems)

- An even faster (linear time?) algorithm . . . or a conditional lower bound

What else we can do?

- No need to know the set C during preprocessing

What we cannot do? (a.k.a. open problems)

- An even faster (linear time?) algorithm . . . or a conditional lower bound

What else Chris Ye can do?

- $T(n)$-time **3SUM** query implies $T(n^2)$-time **Boolean product** of $n \times n$ matrices
  - $\widetilde{O}(n^{1.5})$ query **tight** for **"combinatorial"** algorithms under BMM Hypothesis

What else we can do?

- No need to know the set C during preprocessing

What we cannot do? (a.k.a. open problems)

- An even faster (linear time?) algorithm . . . or a conditional lower bound

What else Chris Ye can do?

- $T(n)$-time **3SUM** query implies $T(n^2)$-time **Boolean product** of $n \times n$ matrices
  - $\widetilde{O}(n^{1.5})$ query **tight** for **"combinatorial"** algorithms under BMM Hypothesis
    - Is even our algorithm "combinatorial"? [Pratt–Uffenheimer–Weinstein]

THANK YOU!