**Definition.** Let $k$ be a positive integer. Given $n$ numbers, decide whether any $k$ of them sum to zero.

**Example.** $(k = 5, n = 11)$

$$\{-975, -505, -430, -237, -178, -29, 67, 439, 660, 674, 898\}$$

**Hardness.**

Pătrașcu & Williams '10 \*No $n^{o(k)}$ algorithm for $k$-SUM!*

Abboud et al. '14 \* $k$-SUM is $W[1]$-complete!

**Time complexity.**

- $k$ even \hspace{1cm} $O(n^{\frac{k}{2}} \log n)$
- $k$ odd \hspace{1cm} $O(n^{\frac{k+1}{2}})$

**Two-track complexity.**

- Query complexity = $Q(n)$
- "Other operations" complexity = $T(n)$
- "Other operations" time complexity = $\tilde{O}(m n^2)$

**Goal.** With queries as small as possible, \(f(k)n^{O(1)}\) queries and \(n^{\frac{k}{2}+O(1)}\) word-RAM running time.

**Results.**

<table>
<thead>
<tr>
<th>Folklore</th>
<th>Erickson '99</th>
<th>Meiser '93</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>$n^{\frac{k}{2}}$</td>
<td>$n^{\frac{k}{2}+1}$</td>
<td>$n^3$</td>
</tr>
<tr>
<td>$k$ odd</td>
<td>$\Omega(n^{\frac{k}{2}})$</td>
<td>$\Omega(n^{\frac{k}{2}+1})$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>$n$</td>
<td>$n^3$</td>
<td>$n^3$</td>
<td>$n^{\frac{k}{2}}+8$</td>
</tr>
<tr>
<td>$S(n)$</td>
<td>$Q(n)$</td>
<td>$T(n)$</td>
<td></td>
</tr>
</tbody>
</table>

**Examples of $k$-SUM arrangements.**

For $n = 2$, $x_2 = 0$, $x_2 + x_2 = 0$, $x_1 = 0$, $x_1 + x_1 = 0$, $x_1 + x_2 = 0$ for $n = 3$.

**Reduction to a point location problem.** Given an input point $q \in \mathbb{R}^n$ and $m$ hyperplanes, decide whether $q$ lies on any of the hyperplanes.

- $n$ input numbers $(q_1, q_2, \ldots, q_n)$
- an input point in $\mathbb{R}^n$
- \(O(n^k)\) $k$-tuple sums $\sum_{j=1}^k x_{i_j} = 0$
- \(O(n^k)\) hyperplanes

**New Algorithm.** Divide the problem into a few $o(n)$-sized subproblems for which we prune and search using $\varepsilon$-nets. We first pick a large $\varepsilon$-net to filter out most of the hyperplanes then fall back to smaller $\varepsilon$-nets.

**Meiser’s algorithm.**

1. Pick an $\varepsilon$-net $N$ of size $O(n^{2 \log^2 n})$.
2. Compute the cell $C$ of the arrangement $A(N)$ that contains $q$ using linear queries.
3. Compute a simplex $S$ inscribed in $C$ and containing $q$ using LP and ray-shooting.
4. Filter then recurse on at most \(n^k\) hyperplanes meeting $S$.

**Improved algorithm.**

1. Pick an $\varepsilon$-net $N$ of size $O(n^{k/2+2 \log^2 n})$.
2. Compute the cell $C$ of the arrangement $A(N)$ that contains $q$ using Meiser’s algorithm.
3. Compute a simplex $S$ inscribed in $C$ and containing $q$ using simplices of step 2.
4. Filter then call Meiser’s algorithm on $O(n^\frac{k}{2})$ hyperplanes meeting $S$.