Encoding 3SUM

Sergio Cabello[1], Jean Cardinal[2], John Iacono[2,3], Stefan Langerman[2], Pat Morin[4], and Aurélien Ooms[2]

1 **University of Ljubljana**
2 **Université libre de Bruxelles**
3 **New York University**
4 **Carleton University**

Abstract

We consider the following problem: given three sets of real numbers, output a word-RAM data structure from which we can efficiently recover the sign of the sum of any triple of numbers, one in each set. This is similar to a previous work by some of the authors to encode the order type of a finite set of points. While this previous work showed that it was possible to achieve slightly subquadratic space and logarithmic query time, we show here that for the simpler 3SUM problem, one can achieve an encoding that takes $\tilde{O}(N^{\frac{3}{2}})$ space for inputs sets of size N and allows constant time queries in the word-RAM.

1 The Problem

Given three sets of N real numbers $A = \{ a_1 < a_2 < \cdots < a_N \}$, $B = \{ b_1 < b_2 < \cdots < b_N \}$, and $C = \{ c_1 < c_2 < \cdots < c_N \}$, we wish to build a discrete data structure (using bits, words, and pointers) such that, given any triple $(i, j, k) \in [N]^3$ it is possible to compute the sign of $a_i + b_j + c_k$ by only inspecting the data structure (we cannot consult A, B, or C). We refer to the map $\chi : [N]^3 \to \{-, 0, +\}, (i, j, k) \mapsto \operatorname{sgn}(a_i + b_i + c_k)$ as the *3SUM-type* of the instance $\langle A, B, C \rangle$. Obviously, one can simply construct a lookup table of size $O(N^3)$, such that triple queries can be answered in $O(1)$ time. We aim at improving on this trivial solution.

2 Motivation

In the 3SUM problem, we are given an array of numbers as input and are asked whether any three of them sum to 0. In the mid-nineties, this problem was identified as a bottleneck of many important problems in geometry, such as detection of affine degeneracies or motion planning [5]. Since then, it has become a central problem in fine-grained complexity theory [9]. It has long been conjectured to require $\Omega(N^2)$ time. In 2014, it was shown to be solvable in $o(N^2)$ time, but no algorithm with running time $O(N^{2-\delta})$ with constant $\delta > 0$ is known [7].

Lower bounds exist in restricted models of computation. Most notably, $\Omega(N^2)$ 3-linear queries are needed to solve 3SUM [4], and nontrivial lower bounds have also been proven for slightly more powerful linear decision trees [1]. However, in a recent breakthrough contribution, Kane, Lovett, and Moran showed that 3SUM could be solved using $O(N \log^2 N)$ 6-linear queries [8], hence within a $O(\log N)$ factor of the information-theoretic lower bound.

Linear decision trees are examples of *nonuniform algorithms*, in which we are allowed to have different algorithms for different input sizes. Algebraic decision trees generalize linear decision trees by allowing decision based on the sign of constant-degree polynomials at each node [10].

Any decision tree identifying the 3SUM-type of a 3SUM instance yields a concise encoding of this 3SUM-type: just write down the outcome of the successive tests. Knowing the decision tree by convention, this sequence of bits is sufficient to recover the sign of any triple.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Table 1 Table of results

	Query time	Space (in bits)	Preprocessing time
Trivial	$O(1)$	$O(N^3)$	$O(N^3)$
Almost trivial	$O(1)$	$O(N^2 \log N)$	$O(N^2)$
Order-type encoding [2]	$O(\log N)$	$O(\frac{N^2 \log^2 \log N}{\log N})$	$O(N^2)$
Order-type encoding [2]	$O(\frac{\log N}{\log \log N})$	$O(\frac{N^2}{\log^{1-\epsilon} N})$	$O(N^2)$
Numeric representation (§4)	$O(N)$	$O(N^2)$	$N^{O(1)}$
Space-optimal representation (§5)	$N^{O(1)}$	$O(N \log N)$	$N^{O(1)}$
Query-optimal (§6)	$O(1)$	$\tilde{O}(N^{1.5})$	$O(N^2)$

The question we consider here is how to make such a representation efficient, in the sense that not only does it use merely a few bits, but the answer to any triple query can be recovered efficiently. Understanding the interplay between nonuniform algorithms and such data structures hopefully sheds light on the intrinsic structure of the problem.

3 Results

See table 1 for a summary. As there are only $O(N^3)$ queries, a table of size $(\log_2 3)N^3 + O(1)$ bits suffices to give constant query time [3]. This can be improved to $O(N^2 \log N)$ bits of space by storing for each pair (i, j) the values $k_<(i, j) = \max\{0\} \cup \{k\colon\ a_i + b_j + c_k < 0\}$ and $k_>(i, j) = \min\{N + 1\} \cup \{k\colon\ a_i + b_j + c_k > 0\}$. For a query (i, j, k), we compare k against the values $k_<(i, j)$ and $k_>(i, j)$ to recover $\chi(i, j, k)$ in $O(1)$ time. All $k_<(i, j)$ and $k_>(i, j)$ can be computed in $O(N^2)$ time via the classic quadratic time algorithm for 3SUM.

One seemingly simple representation is to store the numbers in A, B and C; however these are reals and thus we need to make them representable using a finite number of bits. In Section 4 we show that a minimal integer representation of a 3SUM instance may require $\Theta(N)$ bits per value, which would give rise to a $O(N)$ query time and $O(N^2)$ space, which is far from impressive. In [2] the problem of given a set of N lines, to create an encoding of them so that the orientation of any triple (the *order type*) can be determined was studied; our problem is a special case of this where the lines only have three slopes. Can we do better for the case of 3SUM? We answer this in the affirmative. In Section 5 we show how to use an optimal $O(N \log N)$ bits of space with a polynomial query time. Finally, in section 6 we show how to use $\tilde{O}(N^{1.5})$ space to achieve $O(1)$-time queries.

4 Representation by numbers

A first natural idea is to encode the real 3SUM instance by *rounding* its numbers to integers. We show a tight bound of $\Theta(N^2)$ bits for this representation.

▶ **Lemma 4.1.** *Every 3SUM instance has an equivalent integer instance where all values have absolute value at most $2^{O(N)}$. Furthermore, there exists an instance of 3SUM where all equivalent integer instances require numbers at least as large as the Nth Fibonacci number and where the standard binary representation of the instance requires $\Omega(N^2)$ bits.*

Proof. Every 3SUM instance $A = \{\, a_1 < a_2 < \ldots < a_N \,\}$, $B = \{\, b_1 < b_2 < \cdots < b_N \,\}$, and $C = \{\, c_1 < c_2 < \cdots < c_N \,\}$ can be interpreted as the point $(a_1, \ldots, a_N, b_1, \ldots, b_N, c_1, \ldots, c_N)$ in $\mathbb{R}^{3N}$. Let us use the variables $x_1, \ldots, x_N$ to encode the first N dimensions of $\mathbb{R}^{3N}$,

$y_1, \ldots, y_N$ to encode the next N dimensions, and $z_1, \ldots, z_N$ for the remaining dimensions. Consider the subset of $\mathbb{R}^{3N}$

$$\Delta = \{(x_1, \ldots, x_N, y_1, \ldots, y_N, z_1, \ldots, z_N) \mid x_i < x_{i+1},\ y_j < y_{j+1},\ z_k < z_{k+1}\ \forall i, j, k \in [N-1]\}$$

and the set Π of N^3 hyperplanes $x_i + y_j + z_k = 0$, where $i, j, k \in [N]$. Let $\mathcal{A}$ be the arrangement defined by Π *inside* Δ. Instances of 3SUM correspond to points in Δ. Moreoever, two 3SUM instances have the same 3SUM-type if and only if they are in the same cell of $\mathcal{A}$.

Consider an instance $\langle A, B, C\rangle$ and let $\sigma = \sigma(A, B, C)$ be the cell of $\mathcal{A}$ that contains it. Then σ is the cell defined by the inequalities

$$\forall i, j, k \in [N]: \quad \begin{cases} x_i + y_j + z_k > 0 & \text{if } \chi(i,j,k) = +1, \\ x_i + y_j + z_k = 0 & \text{if } \chi(i,j,k) = 0, \\ x_i + y_j + z_k < 0 & \text{if } \chi(i,j,k) = -1. \end{cases}$$

$$\forall i, j, k \in [N-1]: \quad \begin{cases} x_i - x_{i+1} < 0, \\ y_j - y_{j+1} < 0, \\ z_k - z_{k+1} < 0. \end{cases}$$

Let σ' be the subset of $\mathbb{R}^{3N}$ defined by the following inequalities:

$$\forall i, j, k \in [N]: \quad \begin{cases} x_i + y_j + z_k \geq 1 & \text{if } \chi(i,j,k) = +1, \\ x_i + y_j + z_k = 0 & \text{if } \chi(i,j,k) = 0, \\ x_i + y_j + z_k \leq -1 & \text{if } \chi(i,j,k) = -1. \end{cases}$$

$$\forall i, j, k \in [N-1]: \quad \begin{cases} x_i - x_{i+1} \leq 1, \\ y_j - y_{j+1} \leq 1, \\ z_k - z_{k+1} \leq 1. \end{cases}$$

Clearly σ' is contained in σ. Moreover, for a sufficiently large $\lambda > 0$ the scaled instance $\langle \lambda A, \lambda B, \lambda C\rangle$ belongs to σ'. Therefore, σ' is nonempty.

Since σ' is defined by a collection of linear inequalities defining closed halfspaces, there exists a point p in σ' defined by a subset of at most $3N$ inequalities, where the inequalities are actually equalities. Let us assume for simplicity that exactly $3N$ equalities define the point p. Then, $p = (x, y, z)$ is the solution to a linear system of equations $M[x\ y\ z]^T = \delta$ where M and δ have their entries in $\{-1, 0, 1\}$ and each row of M has at most three non-zero entries. The solution p to this system of equations is an instance equivalent to $\langle \lambda A, \lambda B, \lambda C\rangle$.

Because of Cramer's rule, the system of linear equations has solution with entries $\det(M_i)/\det(M)$, where M_i is the matrix obtained by replacing the ith column of M by δ. We use the following simple bound on the determinant. Since $\det(M) = \sum_\pi \operatorname{sgn}(\pi) \prod_i m_{i,\pi(i)}$, where π iterates over the permutations of $[3N]$, there are at most 3^{3N} summands where π gives non-zero product $\prod_i m_{i,\pi(i)}$ (we have to select one non-zero entry per row), and the product is always in $\{-1, 0, 1\}$. Therefore $|\det(M)| \leq 3^{3N}$. Similarly, $|\det(M_i)| \leq 4^{3N}$ because each row of M_i has at most 4 non-zero entries. We conclude that the solution to the system $M[x\ y\ z]^T = \delta$ are rationals that can be expressed with $O(N)$ bits. This solution gives a 3SUM instance with rationals that is equivalent to $\langle A, B, C\rangle$. Since all the rationals have the common denominator $(\det(M))$, we can scale the result by $\det(M)$ and we get an equivalent instance with integers, where each integer has $O(N)$ bits.

The proof of the second statement is by implementing the Fibonacci recurrence in each of the arrays A, B, C. This can be achieved by letting:

$$\begin{aligned} a_i + b_1 + c_{N-i+1} &= 0, \text{for } i \in [N] \\ a_1 + b_i + c_{N-i+1} &= 0, \text{for } i \in [N] \\ a_{i-1} + b_{i-2} + c_{N-i+1} &< 0, \text{for } i \in \{3, 4, \ldots, N\}, \end{aligned}$$

The first two sets of equations ensure that the two arrays A and B are identical, while the array C contains the corresponding negated numbers, in reverse order. From the inequalities in the third group, and depending on the choice of the initial values a_1, a_2, each array contains a sequence growing at least as fast as the Fibonacci sequence. ◀

Note that this is a much smaller lower bound than for order types of points sets in the plane, the explicit representation of which can be shown to require exponentially many bits per coordinate [6].

5 Space-optimal representation

By considering the arrangement of hyperplanes defining the 3SUM problem, we get an information-theoretic lower bound on the number of bits in a 3SUM-type.

▶ **Lemma 5.1.** *There are* $2^{\Theta(N \log N)}$ *distinct 3SUM-types of size* N*.*

Proof. 3SUM-types of size N are in one-to-one correspondence with cells of the arrangement of N^3 hyperplanes in $\mathbb{R}^{3N}$. The number of such cells is $O(N^{9N})$ and at least $(N!)^2$. ◀

In order to reach this lower bound, we can simply encode the label of the cell of the arrangement in $\Theta(N \log N)$ bits. However, decoding the information requires to construct the whole arrangement which takes $N^{O(N)}$ time. An alternative solution is to store a vertex of the arrangement of hyperplanes $a_i + b_j + c_k \in \{-1, 0, 1\}$. There exists such a vertex that has the same 3SUM-type as the input point, as shown in the proof of Lemma 4.1. To answer any query, either recompute the vertex from the basis then answer the query using arithmetic, or use linear programming. Hence we can build a data structure of $O(N \log N)$ bits such that triple queries can be answered in polynomial time.

Note that we do not exploit much of the 3SUM structure here. In particular, the same essentially holds for k-SUM, and can also be generalized to a Subset Sum data structure of $O(N^2)$ bits, from which we can extract the sign of the sum of any subset of numbers.

6 Subquadratic space and constant query time

Our encoding is inspired by Grønlund and Pettie's $\tilde{O}(N^{1.5})$ non-uniform algorithm for 3SUM [7]. Our data structure stores three components, which we call the *differences*, the *staircase* and the *square neighbors*.

Differences. Partition A and B into *blocks* of $\sqrt{N}$ consecutive elements. Let D be the set of all differences of the form $a_i - a_j$ and $b_k - b_\ell$ where the items come from the same block. There are $O(N^{1.5})$ such differences. Sort D and store a table indicating for each difference in D its rank among all differences in D. This takes $O(\log N)$ bits for each of the $O(N^{1.5})$ differences, for a total of $O(N^{1.5} \log N)$ bits.

	1	2	10	14	17	22	32	33	40	91	92	97	98	110	120	127
1	2	3	11	15	18	23	33	34	41	92	93	98	99	111	121	128
11	12	13	21	25	28	33	43	44	51	102	103	108	109	121	131	138
13	14	15	23	27	30	35	45	46	53	104	105	110	111	123	133	140
19	20	21	29	33	36	41	51	52	59	110	111	116	117	129	139	146
24	25	26	34	38	41	46	56	57	64	115	116	121	122	134	144	151
34	35	36	44	48	51	56	66	67	74	125	126	131	132	144	154	161
51	52	53	61	65	68	73	83	84	91	142	143	148	149	161	171	178
57	58	59	67	71	74	79	89	90	97	148	149	154	155	167	177	184
59	60	61	69	73	76	81	91	92	99	150	151	156	157	169	179	186
114	115	116	124	128	131	136	146	147	154	205	206	211	212	224	234	241
119	120	121	129	133	136	141	151	152	159	210	211	216	217	229	239	246
127	128	129	137	141	144	149	159	160	167	218	219	224	225	237	247	254
128	129	130	138	142	145	150	160	161	168	219	220	225	226	238	248	255
133	134	135	143	147	150	155	165	166	173	224	225	230	231	243	253	260
138	139	140	148	152	155	160	170	171	178	229	230	235	236	248	258	265
142	143	144	152	156	159	164	174	175	182	233	234	239	240	252	262	269

Figure 1 Illustration of the staircase and square neighbors of the constant query time encoding. Here the 16×16 table is partitioned into a 4×4 grid of squares of size 4×4. If $c_k = 100$, the grey illustrates the squares that form the staircase, containing values both larger and smaller than 100. Predecessors and successors within each staircase square are shown in red and blue.

Staircase. Look at the table G formed by all sums of the form $a_i + b_j$, which is monotonic in its rows and columns due to A and B being sorted and view it as being partitioned into a grid G' of size $\sqrt{N} \times \sqrt{N}$ where each *square* of the grid is also of size $\sqrt{N} \times \sqrt{N}$. For each element $c \in C$, for each $i \in [1, \sqrt{N}]$ we store the largest j such that some elements of the square $G'[i, j]$ are $< c$, denote this as $V[c, i]$. We also store, for each $c \in C$, for each $j \in [1, \sqrt{N}]$ the smallest i such that some elements of the square $G'[i, j]$ are $\geq c$, denote this as $H[c, j]$. We thus store, in V and H, $\sqrt{N}$ values of size $O(\log N)$ for each of the N elements of C, for a total space usage of $O(N^{1.5} \log N)$ bits. We call this the *staircase* as this implicitly classifies, for each $c \in C$, whether each square has elements larger than c, smaller than c, or some larger and some smaller; only $O(\sqrt{N})$ can be in the last case, which we refer to as the *staircase* of c.

Square neighbors. For each element $c \in C$, for each of the $O(\sqrt{N})$ squares on the staircase, we store the location of the predecessor and successor of c in the squares $G'[i, V[c, i]]$ and $G'[H[c, j], j]$, for $i, j \in [1, \sqrt{N}]$. This takes space $O(N^{1.5} \log N)$.

To execute a query (a_i, b_j, c_k), only a constant number of lookups in the tables stored are needed. If $j < \sqrt{N} \cdot H[k, i]$, then we know $a_i + b_j > c_k$. If $i > \sqrt{N} \cdot V[k, j]$, then we know $a_i + b_j < c_k$. If neither of these is true, then the square $G'[\lceil i/\sqrt{N} \rceil, \lceil j/\sqrt{N} \rceil]$ is on the staircase of c_i and thus using the square neighbors table we can determine the location of the predecessor and successor of c_k in this square; suppose they are at $G[s_i, s_j]$ and $G[p_i, p_j]$ and thus $G[s_i, s_j] \leq c_k \leq G[p_i, p_j]$. One need only determine how these two compare to $G[i, j] = a_i + b_j$ to answer the query. But this can be done using the differences as follows: to compare $G[s_i, s_j]$ to $G[i, j]$ this would be determining the sign of $(a_i + b_j) - (a_{s_i} + b_{s_j})$ which is equivalent to determining the result of comparing $a_i - a_{s_i}$ and $b_j - b_{s_j}$, which since both are in the same square, these differences are in D and the comparison can be obtained by examining their stored ranks. By doing this for the predecessor and successor we will determine the relationship between $a_i + b_j$ and c_k.

References

1. Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.
2. Jean Cardinal, Timothy M. Chan, John Iacono, Stefan Langerman, and Aurélien Ooms. Subquadratic encodings for point configurations. In *Symposium on Computational Geometry*, volume 99 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
3. Yevgeniy Dodis, Mihai Patrascu, and Mikkel Thorup. Changing base without losing space. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 593–602, 2010.
4. Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.
5. Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.
6. Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *STOC*, pages 405–410. ACM, 1989.
7. Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018.
8. Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-sum and related problems. In *STOC*, pages 554–563. ACM, 2018.
9. Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.
10. J. Michael Steele and Andrew Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3(1):1–8, 1982.