A New Algorithm for Solving the rSUM Problem

Valerii Sopin

June 13, 2022

Valerii Sopin

A new algorithm for solving the $rSUM$ problem

A determined algorithm is presented for solving the $rSUM$ problem for any natural r with a sub-quadratic assessment of time complexity in some cases. In terms of an amount of memory used the obtained algorithm is the $n \log^3 n$ order.

§ 1. Introduction

In computational complexity theory, the $3SUM$ problem asks if a given set of n integers, each with absolute value bounded by some polynomial in n, contains three elements that sum to zero. [1, 2]. The generalized version, $rSUM$, asks the same question for r elements. [1, 2].

The $3SUM$ problem was initially set in [1]. Gajentaan and Overmars collected a large list of geometric problems, which may be solved in an order of quadratic complexity, and nobody knows, how to do it faster [1].

Hereinafter, we understand the order of complexity as asymptotic complexity of the algorithm, namely: the computational complexity (number of operations) of a given algorithm is bounded from above with function $f(n)$ (which is the order of complexity) with accuracy to the constant multiplier and for the sufficiently large input length n.

The $3SUM$ problem has a simple and obvious algorithm for solving in the order of n^2 operations [1, 2].

There are a probabilistic, sub-quadratic algorithms [3] in the computational model, which implies parallel memory operation.

A determined algorithm of solving the $3SUM$ problem based on the Fast Fourier Transformation was suggested in [4]. However it assumes that absolute values of these n numbers are limited by the number $\frac{n^2}{\log n}$.

There are a algorithms based on sorting with partial information [5].

A solution to the generalized version of the problem, $rSUM$, may be found in [2]. Its known order of complexity is $n^{\frac{r}{2}}$ (the "meet-in-the-middle" algorithm).

The paper suggests a determined algorithm of solving the $rSUM$ problem for any $r \in \mathbb{N}$, which is of the order of $n \log^3 n$ in terms of the amount of memory used, with computational complexity of the sub-quadratic order in some cases.

The idea of the obtained algorithm is based not considering integer numbers, but rather $k \in \mathbb{N}$ successive bits of these numbers in the binary numeration system. It is shown that if a sum of integer numbers is equal to zero, then the sum of numbers presented by any k successive bits of these numbers must be sufficiently "close" (see Lemma 2, 3) to zero. This makes it possible to discard the numbers, which a fortiori, do not establish the solution.

§ 2. Algorithm for solving the $rSUM$ problem

Hereinafter, $|y|$ designates an absolute value of integer number y, $\lceil y\rceil$ is the smallest integer greater than or equal to y, $\lfloor y\rfloor$ is the smallest integer smaller than or equal to y. A mapping sign(y) returns the sign of integer y (it returns zero for zero).

Introduce mapping $P_j^k : \mathbb{Z} \mapsto \mathbb{F}_2^k$ for any $k \in \mathbb{N}$ and $j \in \mathbb{N} \cup \{0\}$ as follows:

$$P_j^k(z) = \text{sign}(z) z_j,\ \forall z = \text{sign}(z) \sum_{i=0}^{\infty} z_i 2^{ik} \in \mathbb{Z},$$

i.e. j digit of integer z in a numeral system with base 2^k.

Given: set Ω of n integer numbers, m is the degree of a polynomial, which bounds the maximum absolute value of input numbers ($n^m = 2^{m \log_2 n}$).

Algorithm 1.

1) From among the numbers in question, find ζ, which is the maximum in terms of its absolute value. Calculate $l = \lceil \log_2(\zeta)\rceil$.

2) In a cycle on j from 0 to $\lfloor \frac{l+\lceil \log_2 r\rceil}{3\lceil \log_2 r\rceil}\rfloor$ perform the following:

2.1) Consider the numbers in Ω upon application of $P_j^{3\lceil \log_2 r\rceil}$ and set them down in array Φ_j so that the number of identical elements would not exceed r.

With each $\gamma \in \Phi_j$ group such ordinals of elements in Ω, where numbers with such ordinals in Ω and only these numbers would be equal to γ after using of $P_j^{3\lceil \log_2 r\rceil}$. We associate it with table Π_j.

Brute force to find all $y_1 \in \Phi_j$, where $\exists y_2, y_3, \ldots, y_r \in \Phi_j$:

$$|\sum_{i=1}^{r} P_j^{3\lceil \log_2 r\rceil}(y_i)| < r \ mod\ 2^{3\lceil \log_2 r\rceil},$$

for $j = 0$, strict comparison to zero must be performed.

The gotten r-tuples, namely, their ordinals in Φ_j, are to be set down in Υ_j.

3) Return $\Upsilon = \{\ \Upsilon_j\ \}$ and $\Pi = \{\ \Pi_j\ \}$.

Algorithm 2. `Algorithm for solving the rSUM problem`

1) Perform Algorithm 1: Υ^1, Π^1.

2) Shift the elements of Ω cyclically by $\lceil \log_2 r\rceil$ bits to the right, that the sign bit is retained for all numbers.

3) Perform Algorithm 1 on conditions that for $j = 0$ inequality must be performed rather than comparison, and assume the last $\lceil \log_2 r\rceil$ bits of numbers from Ω to be zero bits: Υ^2, Π^2.

4) Shift the elements of Ω cyclically by $\lceil \log_2 r\rceil$ bits to the right, that the sign bit is retained for all numbers.

5) Perform Algorithm 1 on conditions that for $j = 0$ inequality must be performed rather than comparison, and assume the last $2\lceil \log_2 r\rceil$ bits of numbers from Ω to be zero bits: Υ^3, Π^3.

6) Shift the elements of Ω cyclically by $2\lceil \log_2 r\rceil$ bits to the left, that the sign bit is retained for all numbers.

7) Return $\bigcap_{i,j} \Upsilon_j^i$ relative to elements of Ω.

We are now to prove that the presented algorithms are correct.

LEMMA 1. *For any $y_i \in \mathbb{Z}, i = 1, \ldots, r$, it is true that:*

1) if $\sum_{i=1}^{r} y_i = 0$, then $\sum_{i=1}^{r} y_i \equiv 0 \ mod\ 2^k$, where $k \in \mathbb{N}$.

2) if $\sum_{i=1}^{r} y_i \equiv 0 \ mod\ 2^l, l = \max_i(\lceil \log_2(|y_i|) \rceil + \lceil \log_2 r \rceil)$, then $\sum_{i=1}^{r} y_i = 0$.

PROOF. Obvious. This forms the basis of computer algebra.

The second statement is right because of $\sum_{i=1}^{r} 2^t = r2^t$.

LEMMA 2. *For any $y_i \in \mathbb{Z}, i = 1, \ldots, r$, it is true that:*

if $\sum_{i=1}^{r} y_i = 0$, then $|\sum_{i=1}^{r} P_j^k(y_i)| < r \ mod\ 2^k$,

$j = 0, \ldots, \lfloor \frac{l}{k} \rfloor$, $l = \max_i(\lceil \log_2(y_i) \rceil + \lceil \log_2 r \rceil)$, $k > \lceil \log_2 r \rceil \in \mathbb{N}$.

PROOF. For $j = 0$ the condition of Lemma 2 is met by virtue of Lemma 1.

Assume the opposite meaning that for a value $j = s$, for some r numbers meeting the condition of Lemma 2, the required inequality is wrong. At the same time, by virtue of Lemma 1:

$$\sum_{i=1}^{r} y_i \equiv 0 \bmod 2^{sk}.$$

Present each $y_i \bmod 2^{sk}$ as a sum of the value P_s^k (the last k bits of numbers $\text{sign}(y_i)(|y_i| \bmod 2^{sk})$) and the residue by module $2^{(s-1)k}$, then

$$2^{(s-1)k} \sum_{i=1}^{r} P_s^k(y_i) \equiv -(\sum_{i=1}^{r} \text{sign}(y_i)(|y_i| \bmod 2^{(s-1)k})) \equiv \delta 2^{(s-1)k} \bmod 2^{sk},$$

where $|\delta| < r$, as the sum of r numbers, the absolute value of which is smaller than 2^j for a natural j, cannot exceed $r2^j - r$. Besides, we know from Lemma 1 that $\sum_{i=1}^{r} y_i \equiv 0 \bmod 2^{(s-1)k}$. From here, we obtain the required.

LEMMA 3. *For any $y_i \in \mathbb{Z}, i = 1, \ldots, r$, it is true that:*

if $\sum_{i=1}^{r} y_i = 0$, then for $\tilde{y}_i$ the inequality $|\sum_{i=1}^{r} P_j^k(\tilde{y}_i)| < r \ mod\ 2^k$ is true, where $\tilde{y}_i$ is obtained from y_i by arithmetic shift to the right by t bits.

$t, k > \lceil \log_2 r \rceil$ are any natural numbers, and j is any non-negative integer.

PROOF.

$$2^{t+k(j-1)} \sum_{i=1}^{r} P_j^k(\tilde{y}_i) \equiv -(\sum_{i=1}^{r} \text{sign}(y_i)(|y_i| \bmod 2^{t+k(j-1)})) \bmod 2^{t+kj}.$$

Further on, the proof totally replicates the proof of Lemma 2.

THEOREM 1. *Algorithm 2 will issue the solution of the $rSUM$ problem.*

PROOF. As follows from Lemmas 1, 2, 3, if there exists a solution of the $rSUM$ problem then, after execution of Algorithm 2, and even more so after execution of Algorithm 1, these numbers will stay within Ω.

The cycle on j in Algorithm 1 finishes at iteration $\lfloor \frac{l+\lceil \log_2 r\rceil}{3\lceil \log_2 r\rceil}\rfloor$ by virtue of the second if-clause in Lemma 1.

After step 1), for each $y_1, y_2, \ldots, y_r \in \Omega$ takes place $|\sum\limits_{i=1}^{r} P_j^{3\lceil \log_2 r\rceil}(y_i)| < r \mod 2^{3\lceil \log_2 r\rceil}$ for any j under consideration, for $j = 0$ comparison to zero is performed.

It is about the numbers as such, not some values of $P_j^{3\lceil \log_2 r\rceil}$ of various numbers at each step on j; this is why we remembered ordinals in r-tuples for — to coincide at each step of cycle j.

Hence

$$\sum_{i=1}^{r} y_i = \sum_{i=1}^{\lfloor \frac{l+\lceil \log_2 r\rceil}{3\lceil \log_2 r\rceil}\rfloor} z_i 2^{3i\lceil \log_2 r\rceil}, \text{ where } |z_i| < 2r - 1,$$

as, considering y_i after using of $P_j^{3\lceil \log_2 r\rceil}$, we may lose in $\sum\limits_{i=1}^{r} P_j^{3\lceil \log_2 r\rceil}(y_i)$ $r - 1$ carry bits by absolute value relative to the sum $P_j^{3\lceil \log_2 r\rceil}(\sum\limits_{i=1}^{r} y_i)$ (see the proof in Lemma 2); besides, the very inequality from Lemma 2 makes it possible to differentiate from zero by absolute value to $r - 1$.

Yet, at step 3), the sum $P_j^{3\lceil \log_2 r\rceil}$ of $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_r$, where $\tilde{y}_i$ is y_i at step 2) cyclically shifted to the right by $\lceil \log_2 r\rceil$, will not meet the necessary inequality for module $2^{3\lceil \log_2 r\rceil}$ (see Lemma 3) for the first $j : z_j \neq 0$, if $z_j < r$, as in the latter case, this z_j will not be constituted by the least significant $\lceil \log_2 r\rceil$ bits of a $3\lceil \log_2 r\rceil$-bit number in the binary numeral system, but by more significant bits, which is determined by the fact that

$$\sum_{i=1}^{r} \tilde{y}_i = t + \sum_{i=1}^{\lfloor \frac{l+\lceil \log_2 r\rceil}{3\lceil \log_2 r\rceil}\rfloor} z_i 2^{3i\lceil \log_2 r\rceil - \lceil \log_2 r\rceil}, \text{ where } |t| < r.$$

The correctness of this presentation of the sum $\tilde{y}_i$ follows from ideas presented in Lemmas 2, 3, as, with a cyclic shift of numbers y_i, we may lose $r - 1$ carry bits by absolute value.

At step 5) we will exclude these $y_1, \ldots, y_r$, if the first $z_j \neq 0$ is larger than $r - 1$, for the same considerations.

§ 3. Computational complexity of suggested algorithm

Lemma 4. *Algorithm's 1 order of complexity is $n \log n$.*

Proof. Calculating the maximum element by absolute value is n operations.

Applying $P_j^{3\lceil \log_2 r\rceil}$ to elements of Ω is no more than $2n$ operations (taking in modulus and cyclic shift). Adding the obtained values to Φ_j after applying of $P_j^{3\lceil \log_2 r\rceil}$, containing no more r identical elements, using insertion sort with binary search, is not more than $n(r2^{3\lceil \log_2 r\rceil} + 4\lceil \log_2 r\rceil)$ operations, where we use $4\lceil \log_2 r\rceil$ to assess the complexity of binary search, $r2^{3\lceil \log_2 r\rceil}$ is the number of shifts of elements in an array for insertion to a proper place.

At step 2.1) we solve the $rSUM$ problem by modulus $2^{3\lceil \log_2 r\rceil}$ for a quantity of different numbers not exceeding $r2^{3\lceil \log_2 r\rceil}$, though there may be more than one solution. The exhaustive enumeration of all the variants requires $r^r 2^{3r\lceil\log_2 r\rceil}$ operations.

All the above-calculated was a single iteration on cycle of j.

As $l = m\lceil\log_2 n\rceil + \lceil\log_2 r\rceil$ and $r,\ m$ are fixed numbers, we obtain the required assessment.

REMARK 1. It is convenient to assume that each element in the r-tuple from Υ_j (where elements of the r-tuple are ordinals of elements in Φ_j, as determined by us) is a column of such ordinals of elements in Ω, that the numbers corresponding to these ordinals in Ω upon application of $P_j^{3\lceil\log_2 r\rceil}$ will be equal to an element with this ordinal. We may assume so, because we have a table of association of the elements in Φ_j with elements in Ω.

THEOREM 2. *Algorithm's 2 order of complexity is sub-quadratic for some cases.*

PROOF. All steps of the Algorithm 2 except step 7) do not exceed the $n\log n$ order (see Lemma 4).

How to compute $\bigcap\limits_{i,j}\Upsilon_j^i$ relative to elements of Ω?

All r-tuples from Υ_j^i are tables, see Remark 1.

Υ_j^i contains no more $2r!r2^{3\lceil\log_2 r\rceil(r-1)}$ items. Comparing a r-tuple with another according to ordinals in Ω will not make more than $rn\log_2 n$ operations. Consider $\log_2 n$ as elements in Ω are read successively, and hence, ordinals of elements of Ω, related to an element of Φ_j, are set down in an orderly way, which means that we may use binary search. Every time we create new r-tuple with common ordinals of Ω in columns in one r-tuple and the other, if there is at least one common element in each column.

As cycle j ends $\lceil\frac{m\lceil\log_2 n\rceil}{3\lceil\log_2 r\rceil}\rceil$ in Algorithm 1 and there are 3 execution of Algorithm 1 in Algorithm 2, we get upper bound of vertices of such comparing r-tuples tree:

$$(2r!r2^{3(r-1)\lceil\log_2 r\rceil})^{\lceil\frac{m\lceil\log_2 n\rceil}{\lceil\log_2 r\rceil}\rceil}.$$

It's a lot, that's why we compute

$$\Gamma_s = \bigcap_{i,\ j=sh,\dots,(s+1)h-1}\Upsilon_j^i,\ where\ i=1,2,3,\ h=\lceil\frac{\lceil\log_2\log_2 n\rceil}{9r\lceil\log_2 r\rceil}\rceil,\ s=0,\dots,\lceil\frac{m\lceil\log_2 n\rceil}{3h\lceil\log_2 r\rceil}\rceil.$$

Cardinality of Γ_s is less than

$$(2r!r2^{3\lceil\log_2 r\rceil(r-1)})^{\lceil\frac{\lceil\log_2\log_2 n\rceil}{3r\lceil\log_2 r\rceil}\rceil} \leqslant \log_2^2 n.$$

So, the order of complexity of the computation of all Γ_s is less than $n\log_2^3 n$.

Find $\lceil\frac{\log_{\lceil\log_2 n\rceil} n}{3}\rceil$ sets Γ_s with the smallest number of elements (it is of the order of $n\log n$ operation) and compute confluence of them Θ (it is of the order of $n^{\frac{5}{3}}\log_2 n$ operations).

To count the quantity of all variants produced by each r-tuple from Θ, relative to elements of Ω, takes no more than $2rn^{\frac{5}{3}}$ operations (amount of options generated by fixed r-tuple is the product of the number of items in a columns of this r-tuple).

If the total number of r-tuples from Θ, relative to elements of Ω, is less than $n^{\frac{3}{2r}}$, we get sub-quadratic time for our algorithm (brute force all of variants).

If the total number of r-tuples from Θ, relative to elements of Ω, is less than $\frac{n^{\frac{1}{2}}}{\log^{\frac{1}{r}} n}$, brute force still would be faster than using known algorithms.

THEOREM 3. *Algorithm 2 requires an amount of memory of an order $n \log^3 n$ relative to storage of integers.*

PROOF. As will readily be observed, the most memory-consuming step is 7).

Step 7) of Algorithm 2 requires some memory for Υ_j^i (constant quantity) and Π_j^i associating elements in Υ_j^i with elements in Ω (not more than the order of n), $i = 1, 2, 3$, $j = 0, \ldots, m \log n + \log r$.

All together Γ_s require the order of $n \log^3 n$ memory, see Theorem 2.

REMARK 2. What is it about the constant in asymptotic complexity?

As follows from Theorem 2 and Lemma 4 the constant would not exceed $3mr^{4r}$.

REMARK 3. As time and memory complexity of suggested algorithm is of the sub-quadratic order, it seems to be useful to perform it at the beginning of any other known algorithm.

Список литературы

[1] A. Gajentaan and M. Overmars, "On a class of $O(n^2)$ problems in computational geometry", *Computational Geometry: Theory and Applications*, 1995, № 5, 165–185.

[2] J. Erickson, "Lower bounds for linear satisfiability problems", *Chicago Journal of Theoretical Computer Science*, **8** (1999).

[3] I. Baran, E. Demaine and M. Patrascu, "Subquadratic algorithms for $3SUM$", *Lecture Notes in Computer Science*, 2005, № 3608, 409—421.

[4] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2001.

[5] A. Gronlund and S. Pettie, "Threesomes, Degenerates, and Love Triangles", *arXiv:1404.0799*, 2014.

Valerii Sopin

Lomonosov Moscow State University, Moscow

E-mail: VvS@myself.com