J. Symbolic Computation (2000) **29**, 33–57
doi: 10.1006/jsco.1999.0295
Available online at http://www.idealibrary.com on IDEAL

Fast Constructive Recognition of a Black Box Group Isomorphic to S_n or A_n using Goldbach's Conjecture

SERGEY BRATUS[†] AND IGOR PAK[‡]

[†]*Department of Mathematics, Northeastern University, Boston, MA 022115, U.S.A.*
[‡]*Department of Mathematics, Yale University, New Hoven, CT 06520, U.S.A.*

The well-known Goldbach Conjecture (GC) states that any sufficiently large even number can be represented as a sum of two odd primes. Although not yet demonstrated, it has been checked for integers up to 10^{14}. Using two stronger versions of the conjecture, we offer a simple and fast method for *recognition* of a gray box group G known to be isomorphic to S_n (or A_n) with *known* $n \geq 20$, i.e. for construction[†] of an isomorphism from G to S_n (or A_n). Correctness and rigorous worst case complexity estimates rely heavily on the conjectures, and yield times of $O([\rho+\nu+\mu]\, n \log^2 n)$ or $O([\rho+\nu+\mu]\, n \log n / \log\log n)$ depending on which of the stronger versions of the GC is assumed to hold. Here, ρ is the complexity of generating a uniform random element of G, ν is the complexity of finding the order of a group element in G, and μ is the time necessary for group multiplication in G. Rigorous lower bound and probabilistic approach to the time complexity of the algorithm are discussed in the Appendix.

1. Introduction

Groups that most often become the object of computer-aided computations, are those that arise as automorphism groups of combinatorial structures. Permutation groups so far remain the best understood class of these, with the most efficient algorithms available. These algorithms were built on the fundamental algorithms of Sims (1971) that made it possible to determine group membership and group order.

While algorithms available for matrix groups over finite fields are not nearly as efficient, the present massive efforts in this direction give hope that this class of groups will soon also be well understood. An important base case for the matrix group recognition project is recognition of classical simple groups of Lie type presented as subgroups of general matrix groups (possibly of different dimensions, or finite fields, or both).

The efforts at recognition of finite simple matrix groups have been a major motivation for the theory of black box recognition of groups (defined below). Neumann and Praeger (1992) presented the first major result along these lines. The papers by Babai (1997) and Niemeyer and Praeger (1997) contain excellent surveys of more recent progress. The paper by Beals and Babai (1993) deals with recognition methods for general finite simple groups, including A_n.

The result of this paper adds to these base cases by considering constructive recognition

[†]Whereas our method can be applied to construct the image of every element of G in S_n, we actually construct the images of $(1, 2)$ and $(1, 2, \ldots, n)$ and show how to construct the images of any other element.

of S_n and A_n, where $n \geq 20$. In this case, the natural matrix representation corresponds to the natural permutation representation, and so the paper will work directly with black box recognition of the groups as permutation groups. We diverge from pure black box results by assuming the availability of algorithms for finding random group elements and for finding the order of a group element. A theoretical algorithm for finding random group elements in black box groups in polynomial time exists in the work of Babai (1991). The work of Celler *et al.* (1995) introduces the so-called *product replacement* algorithm that appears to work well in practice. Its theoretical complexity is currently under investigation (see Babai, 1997, and Diaconis and Saloff-Coste, 1998).

Randomized methods work well for the problems of *group recognition*, when the object of computation is a large group of a known type, but presented in a non-standard way. Such problems arise, for instance, in the classification of certain classes of finite groups. Randomized methods have been used to answer questions such as whether a group contains a classical subgroup of specific type (see Cameron and Cannon, 1991; Neumann and Praeger, 1992, and Niemeyer and Praeger, 1998), or to construct a natural linear representation (or a permutation representation) of a group presented in a non-standard way (see Cooperman *et al.*, 1997; Celler and Leedham-Green, 1997a, and Kantor and Seress, 1997).

The concept of a *black box group* provides a uniform framework for problems involving randomized algorithms for large groups presented in a non-standard way. Such problems are often referred to as recognition problems.

By a *black box group* we mean a group whose elements are represented by binary strings of uniform length N, and the group operation is performed by an oracle (the black box). The oracle (black box) can compute the product of elements, find the inverse, and recognize the identity element[†] in time polynomial[‡] in N. For a more formal treatment see Cooperman and Finkelstein (1998).

By *recognizing* a black box group G known to be isomorphic to a classical group we mean constructing an isomorphism between G and the natural representation of the classical group. In other words, we can think of a black box group as a group given only by its multiplication table, which we can view one entry at a time. Therefore, when working with the black box model, one can rely only on the group structure, not on the properties of a particular representation of the group in question.

We use the term *gray box group*, introduced by Celler and Leedham-Green, in a sense different from the original, to denote a black box group G such that there is an efficient algorithm (or an oracle) for computing the order $r(x)$ of any element $x \in G$. A number of efficient ideas for computing the order of an element in a matrix group can be found in Celler and Leedham-Green (1997b). We also show how to extend our methods to black box groups (i.e. to the situation when no efficient way of computing the order of an element is available). Namely, rather than finding the orders of elements explicitly, we use an algorithmic shortcut to verify the necessary properties of elements by using only black box multiplication (see Section 6).

In this paper we solve the recognition problem for a gray box group known to be

[†]A popular variation of Babai's original definition assumes that elements of the black box group are encoded by bit strings *uniquely*, so that the operation of recognizing an element as the identity element of the group becomes trivial. Computationally, this assumption was motivated by the use of hashing on bit stings. The present paper makes no such assumption.

[‡]We use the terms *quadratic*, *linear* and *polynomial* to mean times $O(n^2)$, $O(n)$, and $O(n^c)$, respectively, omitting the group-specific factors ρ, μ, ν, where such omissions do not lead to confusion.

isomorphic to the symmetric group S_n with known n. Given a gray box group G, defined by a set of generators, we construct an isomorphism $\phi : G \to S_n$.

More precisely, we first find two elements $a, b \in G$ such that there exists a unique isomorphism ϕ that maps a into the transposition $(1, 2)$ and b into the long cycle $(1, 2, \dots, n)$. ϕ We then provide a simple algorithm that, for an arbitrary element $x \in G$, computes the permutation $\phi(x) \in S_n$ to which x is mapped by our isomorphism. In this sense, our algorithm is *constructive*.

We make a further assumption concerning the group G. Namely, we assume that there is an efficient algorithm for producing (nearly) uniformly distributed elements of $G \cong S_n$ given the set of generators of G. This assumption is motivated by a provable polynomial-time algorithm (Babai, 1991) and an efficient heuristic algorithm (Celler *et al.*, 1995).

Let G be a gray box group specified by a generating set $\mathcal{G}$. Further, let G be isomorphic to S_n for a known n. Let ρ be the time required to compute a (nearly) uniform random element of G, let ν be the time required to find the order of an arbitrary element of G, and let μ be the time required to perform a group operation in G. Also, recall the standard notions of Monte Carlo and Las Vegas algorithms (see e.g. Babai, 1997).

Our results are conditional on the two number theoretic conjectures presented in Section 5, which we call the *Extended Goldbach Conjecture* (EGC) and the *Weighted Goldbach Conjecture* (WGC). Namely, while the algorithm remains the same, its complexity estimate depends on the particular version of the conjecture (EGC or WGC), which is reflected by Theorems 1 and 1′.

Theorem 1. *Under the definitions of the preceding paragraphs, and provided that $n \geq 20$ is known and the EGC*[†] *holds, a Las Vegas algorithm exists, which in time $O([\rho + \nu + \mu]\, n \log^2 n)$ can produce a data structure which can then be used to compute the image $\phi(x)$ in S_n of any $x \in G$ under a fixed isomorphism $\phi : G \to S_n$. Each such computation of $\phi(x), x \in G$ will take $O(\mu\, n \log n)$ time.*

Theorem 1′. *Under the definitions of the preceding paragraphs, and provided that $n \geq 20$ is known and the WGC*[†] *holds, the Las Vegas algorithm of Theorem 1 will produce the required data structures that define $\phi(x)$ in time $O([\rho + \nu + \mu]\, n \log n / \log\log n)$.*

Remark 1. Suppose that the group G is only a black box group, i.e. no algorithm for computing the orders of its elements is available. The information about the elements of G required for our algorithm can still be extracted, by using just the black box multiplication in G as described in Section 6. In this case the value of ν in the above theorems becomes $\nu = O(\mu n)$.

Remark 2. For several values of n less than 20 our algorithm fails. However, in those cases one can still try using the information on the distribution of orders in S_n and A_n to obtain a constructive recognition. These groups are relatively small, so the ad hoc approach seem satisfactory. It is not considered in this paper. From now on we will always assume that $n \geq 20$.

Remark 3. In the Appendix we present a *lower bound* for the time complexity of the algorithm which matches the upper bound in Theorem 1′. The proof is a rigorous exercise

[†]Actually, we need the conjecture to hold only for n, $n - 2$ if n is even, or for $n - 1$, $n - 5$ if n is odd.

in analytic number theory. In addition, we present a heuristic probabilistic reasoning in favor of the WGC.

In Section 5 we present weaker bounds that do not assume the Goldbach Conjecture (GC). In addition to the main algorithm, we discuss its variations for recognizing A_n (see Section 7), and for checking whether $G \cong S_n$ (see Section 8). Finally, a variation that recognizes S_n when n is not known is discussed in Section 9. There we prove the following result.

Theorem 2. *Given an upper bound M for $n \geq 20$, and provided that the EGC holds, a Monte Carlo algorithm exists which in time $O([\rho + \nu + \mu]M \log^2 M)$ can find n, and which can be made Las Vegas requiring $O([\rho + \mu + \nu]M^2)$ time. Moreover, if the WGC holds, a variation of the above Monte Carlo algorithm will find n in time $O([\rho + \nu + \mu]M \log M/ \log\log M)$.*

An upper bound, M, can be derived from the uniform bit string length N of the black box group by observing that $n! \leq 2^N$.

The idea of the algorithm is as follows. We pick elements $g \in G$ randomly until we find an element of order $r(g) = p_1 p_2$, where p_1, p_2 are primes such that either $p_1 + p_2 = n$ (for n even), or $p_1 + p_2 = n - 1$ (for n odd). The existence of such elements follows from the GC. It is clear that such an element consists of exactly two cycles of lengths p_1 and p_2. We refer to them as *Goldbach elements.* It turns out that there are sufficiently many of them. Next we find one or two (depending on the parity of n) transpositions such that together with our Goldbach element they generate the entire symmetric group S_n. Finding these transpositions involves a modification of Goldbach elements and some technical details. Conjugating the transposition by the Goldbach element, we obtain the desired $a, b \in G$ that define the isomorphism ϕ. Thus, the complexity estimate of the algorithm depends on the proportion P_n of Goldbach elements in S_n. The estimates on P_n will be given in Section 5 and are based on various number theoretical results concerning the GC. They are also confirmed by experimental evidence obtained on computers. We refer to the Appendix for a mathematical treatment of P_n.

We state the algorithm first, then prove its correctness assuming the conjectures, and finally elaborate on possible improvements (including removing the conjectures altogether). We need the following trivial observation: given a transposition $\tau = (i_1, i_2)$ and a permutation σ, we know that $\sigma\tau\sigma^{-1}$ is also a transposition, and

$$\sigma\tau\sigma^{-1} = (\sigma^{-1}(i_1), \sigma^{-1}(i_2)).$$

Therefore if σ is (nearly) uniform in S_n, then $\sigma\tau\sigma^{-1}$ is (nearly) uniform in the set of transpositions of S_n.

We say that a permutation σ *touches* a point if it does not fix that point (i.e. the point belongs to the support of σ). We say that the transposition $\tau = (i_{1,} i_2)$ *touches* the cycle $\sigma = (j_1, j_2, \ldots, j_{p_2})$ if either $i_1$or i_2 or both belong to the support of the cycle σ.

A few words about the practical implementation of the algorithm. The authors have implemented it in C and GAP and ran the tests for the values of n up to several thousands. We believe that, given enough RAM and processing power, the algorithm remains practical for the values of n up to 10^5.

2. The Algorithm

Our algorithm calls on a procedure for finding the order $r(x)$ of an element $x \in G$, and a procedure that produces (nearly) randomly distributed elements of G. Both the algorithm and the latter procedure take as input the set of generators $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$ of $G \cong S_n$ as a black box group, and the number n, which is the degree of $G \cong S_n$. It outputs two elements $a, b \in G$ such that there exists an isomorphism $\phi : G \to S_n$ carrying a to $(1,2)$ and b to $(1, 2, \ldots, n)$. Its complexity for general n is $O([\rho + \mu + \nu]/P_n)$, where P_n is the proportion of Goldbach elements in S_n.

Elements a, b produced by the above algorithm are then used by the procedure that takes an element $x \in G$ and outputs the permutation $\phi(x)$ into which x is carried by $\phi : G \to S_n$ for which $\phi(a) = (1,2)$, $\phi(b) = (1, \ldots, n)$. This procedure is described in Section 4.

The algorithms for S_{2k} and S_{2k+1} differ slightly. Namely, when n is odd, we find two transpositions, whereas for even n finding one is sufficient.

2.1. THE ALGORITHM FOR S_{2k}

(1) *Locate a Goldbach element in G.*
Find an element $y \in G$ whose order, $r(y)$, satisfies

$$r(y) = p_1 p_2,$$

where p_1, p_2 are distinct primes such that $p_1 + p_2 = n$. By the slight extension of the GC there will be at least one such pair of primes (p_1, p_2). An element y as above consists of two cycles of lengths p_1 and p_2, respectively. Clearly, $y_1 = y^{p_2}$ and $y_2 = y^{p_1}$ are cycles of lengths p_1 and p_2, respectively. Denote

$$y_1 = (i_1, i_2, \ldots, i_{p_1}),$$
$$y_2 = (j_1, j_2, \ldots, j_{p_2}).$$

We can re-number the points so that

$$\{i_1, i_2, \ldots, i_{p_1}\} = \{1, \ldots, p_1\}$$

and $\{j_1, j_2, \ldots, j_{p_2}\} = \{p_1 + 1, \ldots, n\}$.

(2) *Locate a transposition in G.*
To this end, search for an element $x \in G$ such that

$$r(x) = 2\, q_1\, q_2,$$

where $q_1 + q_2 = n - 2$, q_1, q_2 are distinct odd primes.
Then x consists of 2 odd prime cycles of lengths q_1 and q_2 and a transposition. Hence

$$a = x^{q_1 q_2}$$

is a transposition. Once we have found a transposition, we can easily find other transpositions, sampled independently from a nearly uniformly distribution on S_n, by conjugating the original transposition by random elements of G. See Beals and Babai (1993), Cooperman *et al.* (1997) and Kantor and Seress (1997) for different versions of a similar approach.

(3) *Check if the transposition a touches both y_1 and y_2. If not, produce and try random transpositions until we have this property.*

We are looking for a transposition that interchanges a point from the cycle y_1 with a point from the cycle y_2 (i.e. touches both cycles). This is the case if and only if a commutes with neither y_1 nor y_2. Consequently, we check that

$$ay_1 \neq y_1 a \qquad \text{and} \qquad ay_2 \neq y_2 a.$$

(4) *Compute $b = y_1 a y_2$. This is a long cycle in G.*
If $a = (l_1, l_2)$ commutes with neither cycle, then one of l_1 and l_2 lies in the support of y_1, and the other in the support of y_2. Assume, without loss of generality, that

$$\{l_1, l_2\} = \{p_1, p_1 + 1\}.$$

It is easy to see that

$$b = y_1 a y_2$$

is a cycle of length $p_1 + p_2 = n$ in G, and that there is a homomorphism $\phi : G \to S_n$ that carries b into the long cycle $(1, 2, \ldots, n)$ and a into $(p_1, p_1 + 1)$.

(5) *Conjugate a by $b^{p_1 - 1}$ to obtain a'.*
Now

$$\phi(b) = (1, 2, \ldots, n), \qquad \phi(a) = (p_1, p_1 + 1), \qquad \phi(a') = (1, 2),$$

and a', b is the standard pair of generators for S_n. As $\{a, b\}$ generate S_n (any transposition together with any cycle of length n do so), we have fixed the isomorphism $\phi : G \to S_n$, and, in particular, can compute the image of any element $z \in G$ under the desired isomorphism ϕ explicitly (see Section 4).

2.2. THE ALGORITHM FOR S_{2k+1}

(1) *Locate a Goldbach element in G.*
Find an element $y \in G$ such that

$$r(y) = p_1 p_2,$$

where $p_1 + p_2 = n - 1$. Then y consists of two cycles of lengths p_1 and p_2 and a fixed point m. We obtain two cycles $y_1 = y^{p_2}$ and $y_2 = y^{p_1}$ of lengths p_1 and p_2, respectively. Denote, as above,

$$y_1 = (i_1, i_2, \ldots, i_{p_1}),$$
$$y_2 = (j_1, j_2, \ldots, j_{p_2}),$$

and re-number the points so that $\{i_1, i_2, \ldots, i_{p_1}\} = \{1, \ldots, p_1\}$ and $\{j_1, j_2, \ldots, j_{p_2}\} = \{p_1 + 1, \ldots, n - 1\}$.

(2) *Locate a transposition in G.*
Look for an element $x \in G$ such that

$$r(x) = 6\, q_1\, q_2,$$

where $q_1 + q_2 + 5 = n$, and q_1, q_2 are distinct odd primes > 3. The existence of such a pair of primes constitutes a slight extension of the GC. The element x will have the cycle structure of $(q_1, q_2, 3, 2)$. The element $a_1 = x^{3 q_1 q_2}$ is then a transposition. Any number of (nearly) uniformly distributed random transpositions can now be obtained by conjugating a_1 by random elements of G.

(3) *Check if the transposition a_1 touches both cycles y_1 and y_2. If not, keep producing and trying random transpositions until successful.*
The transposition a_1 touches the cycle y_1 if and only if they do not commute, and similarly for cycle y_2. So, as above, we only need to check that

$$a_1 y_1 \neq y_1 a_1 \qquad \text{and} \qquad a_1 y_2 \neq y_2 a_1.$$

(4) *Locate a transposition that touches both the fixed point m and one of the cycles (y_1 or y_2). Keep producing and trying random transpositions until this is the case.*
Obtain a random transposition a_2 (by conjugating a_1 with a random element of G). First check whether a_2 commutes with y_1 and y_2. If a_2 commutes with both, or with neither of them, discard the current a_2 and obtain a new one. Suppose a_2 commutes with y_2 but does not commute with y_1. We need to verify if a_2 touches only one point from the support of y_1. To this end, check if a_2 commutes with $y_1 a_2 y_1^{-1}$ and $y_1^2 a_2 y_1^{-2}$. If a_2 commutes with at least one of these, then a_2 only permutes the vertices inside the support of y_1, and hence it does not touch m (see Lemmas 1 and 2 of Section 3). Special care needs to be taken in the case when $r(y_1) = 3$, which, fortunately, can be easily detected. If the support of a_2 lies entirely inside that of y_1, discard a_2 and start over. The case when a_2 commutes with y_1 but does not commute with y_2 is analogous.

(5) *Compute $b = y_1 a_1 y_2 a_2$. This is a long cycle in G.*
Indeed, re-numbering the points in a suitable fashion, we can have $a_1 = (p_1, p_1 + 1)$ and $a_2 = (n-1, n)$. Then

$$b = y_1 a_1 y_2 a_2$$

is a long cycle, carried to $(1, 2, \ldots, n)$ by some isomorphism $\phi : G \to S_n$.

(6) *Conjugate a_1 by $b^{p_1 - 1}$to obtain a'.*
We now have the transposition a' and the long cycle b which together generate the entire $G \cong S_n$ and

$$\phi(a') = (1, 2),$$
$$\phi(b) = (1, 2, \ldots, n).$$

3. Why does this Work?

Let us state the following two extensions of the GC. We will need these conjectures for estimating the probability P_n of finding a Goldbach element.

Extended Goldbach Conjecture. For every even $n \geq 20$, there exist primes $p_1 > p_2 > 3$ such that $p_1 + p_2 = n$. Moreover, the number $\pi_2(n)$ of such pairs is

$$\pi_2(n) > \frac{1}{3} \frac{n}{\log^2 n}.$$

We have verified this conjecture for $20 \leq n \leq 10^6$. With different constants this conjecture goes back to Sylvester (1871) and Brun (1915). The exact asymptotic formula was later found and conjectured by Hardy and Littlewood (1922). In Section 5 we will explore this connection and elaborate on similar results for triples of primes, etc. In the Appendix we present a heuristic argument which demonstrates where the factor $n/\log^2 n$ in the Hardy–Littlewood formula comes from.

Let n be even. Recall that P_n is equal to the proportion of Goldbach elements in S_n. In the following subsection we will show that P_n can be defined as follows:

$$P_n = \sum_{p>q>2 \ \text{primes}, p+q=n} \frac{1}{p\,q}.$$

Below we will show that the EGC implies $1/P_n = O(n \log^2 n)$.

Weighted Goldbach Conjecture. There exists a positive constant $c > 0$ such that for all even $n \geq 20$ we have

$$P_n > \frac{1}{4} \frac{\log \log n}{n \log n}.$$

This conjecture is probably new and unproven. It is supported by both numerical and theoretical results, which are presented in the Appendix. We have verified this conjecture for $20 \leq n \leq 10^6$. We use the term *weighted* here since in contrast with the EGC we sum $1/pq$ rather than 1 for each instance of primes $p + q = n$.

3.1. finding a Goldbach element

The first step of the algorithm consists of finding a Goldbach element. The EGC guarantees that such elements exist for all n. Let us estimate the probability that a randomly chosen element of G is a Goldbach element. Goldbach elements of order $p_1\,p_2$ form a conjugacy class corresponding to partitions (p_1, p_2) (for even n) and $(p_1, p_2, 1)$ (for odd n). It is well known that the probability of obtaining an element $g \in G$ from the conjugacy class corresponding to the partition $(\lambda_1, \lambda_2, \ldots)$, $\lambda_1 > \lambda_2 > \cdots$, is equal to

$$\frac{1}{\lambda_1\,\lambda_2 \ldots}.$$

Hence the probability of finding an element of order $p_1\,p_2$ in G is

$$\frac{1}{p_1\,p_2} \geq \frac{4}{n^2}.$$

Using the EGC we find that

$$P_n = \sum_{p>q>2 \ \text{primes}, p+q=n} \frac{1}{p\,q} = \Omega\left(\pi_2(n)\frac{4}{n^2}\right) = \Omega\left(\frac{1}{n \log^2 n}\right).$$

Therefore, assuming the EGC the expected number of tries one needs to make before obtaining a Goldbach element is $O(n \log^2 n)$.

Recall that the WGC claims that $P_n = \Omega(\log \log n / n \log n)$. Therefore, assuming the WGC, the expected number of tries one need to make before obtaining a Goldbach element is $O(n \log n / \log \log n)$.

3.2. finding an arbitrary transposition

For subsequent steps of the algorithm we need to find random transpositions. Once a single transposition is found, this can be easily achieved by conjugating it with random elements of G (we assume availability of an efficient algorithm for producing (nearly) uniformly distributed random elements of G). The question, then, is how to find the initial transposition in the gray box setting.

We start with elements x in the conjugacy classes corresponding to partitions $(q_1, q_2, 2)$ (even n) and $(q_1, q_2, 3, 2)$ (odd n). The latter is the only possible cycle form of an element $x \in G$ of order $r(x) = 6q_1q_2$, where $q_1 + q_2 + 5 = n$. For fixed q_1, q_2 the probabilities of finding them are $\frac{1}{2\,q_1\,q_2}$ and $\frac{1}{6\,q_1\,q_2}$, respectively. Therefore the expected number of tries before locating such an element x is $2/P_{n-2}$ and $6/P_{n-5}$, respectively. Assuming the EGC, both quantities are $O(n \log^2 n)$, while assuming the WGC, both quantities are $O(n \log n/ \log\log n)$.

3.3. TRANSPOSITIONS TOUCHING CYCLES

In both the even and the odd cases we find a transposition that touches each of the two cycles y_1, y_2 of prime lengths p_1 and p_2 ($p_1 + p_2 = n$ or $p_1 + p_2 = n - 1$). In the odd case we also need a transposition that touches the point m and one of these cycles (i.e. simply permutes m with a point from the support of the cycle).

Note that, as we have remarked in the Introduction, the permutations resulting from conjugating the initial permutation with (nearly) uniformly distributed elements of G are themselves (nearly) uniformly distributed. The probability that a randomly chosen transposition touches both y_1 and y_2 is, of course,

$$\frac{p_1 \cdot p_2}{\binom{n}{2}} \geq \frac{1}{n}.$$

Thus it will take no more than n tries to locate the required transposition; much fewer if p_1 and p_2 are big enough. For instance, should we decide to consider only Goldbach elements with cycles of lengths $p_1 > p_2 > n/6$, the number of transpositions touching both cycles will be at least $(n-1)/6 \cdot 5(n-1)/6$. Thus the probability that we find such a transposition will simply be constant,

$$\frac{\frac{n-1}{6} \cdot \frac{5(n-1)}{6}}{\binom{n}{2}} \geq \frac{2}{9}.$$

Analogously, the probability of finding a transposition that touches the fixed point m and one of the circles is at least $1/n$.

Given a transposition a and a cycle y of length greater than 3, we want to be able to verify that a touches y (i.e. that their supports are not disjoint). We need a few simple lemmas here.

LEMMA 1. *A transposition and a cycle of length greater than* 2 *touch if and only if they do not commute.*

LEMMA 2. *A transposition a and a cycle y of length greater than* 3 *touch at only one point (i.e. their supports have only that one point in common, a permutes a point inside the support of y with a point outside the latter) if and only if a commutes with neither y, nor yay^{-1}, nor y^2ay^{-2}.*

PROOF. Suppose the supports of a and y are disjoint. Then $ay = ya$.

Suppose the support of a lies inside that of y (a permutes two points inside the cycle). Then both yay^{-1} and y^2ay^{-2} will permute points inside the cycle, and either yay^{-1} or y^2ay^{-2} will be disjoint from a and hence will commute with a. Finally, if a and y have only one point in common, while the other point from the support of a is outside the cycle, then both yay^{-1} and y^2ay^{-2} permute that other point from a with some point from y, hence neither of the two commutes with a.□

LEMMA 3. *If y_1, y_2 are disjoint cycles of lengths l_1, l_2, and a is a permutation that touches both, then $z = y_1ay_2$ is a cycle of length $l_1 + l_2$. In particular, if y is a cycle of length $n-1$ and a is a transposition that touches y at only one point, then $z = ya$ is a cycle of length n.*

The proof of Lemma 3 is straightforward. These lemmas imply the correctness of our algorithm.

4. Finding the Image of an Arbitrary Element of G

We now present an algorithm that takes as input an arbitrary element $x \in G$, elements $a, b \in G$ above, as well as the number n and the set of generators of G, and outputs the permutation $\omega = \phi(x)$ which is the image of x under the isomorphism ϕ fixed by the condition that $\phi(a) = (1, 2)$ and $\phi(b) = (1, \dots, n)$. The algorithm takes $O(\mu n \log n)$ time and requires storing $2n$ elements of G. The latter are pre-computed and afterwards used by each instance of the procedure.

Recall that we say that two elements of $G \cong S_n$ *touch* one another if one of them moves (does not fix) a point from the support of the other, i.e. the intersection of their supports is non-empty. In simple cases, such as the case of both elements being transpositions, checking if the elements commute (or are identically equal) is enough to decide if they touch. The same is true if one element is a transposition and the other is a (single) cycle, in which case it is enough to check the commutation of the original transposition and its conjugates by the cycle and the square of the cycle.

We will write $a \sim (1, 2)$, $b \sim (1, 2, \dots, n)$ instead of $\phi(a) = (1, 2)$, $\phi(b) = (1, 2, \dots, n)$, and similarly for other elements of G.

As stated above, our algorithm will be nearly linear in n. It is based on two simple ideas. The first idea is to conjugate a known transposition (pre-computed by multiplying a, b) with the element x and establish what transposition results from it by checking how it touches other known ones. In this fashion it is possible to establish the images of each point under x, i.e. the permutation $\phi(x)$ that corresponds to x. Straightforward application of this idea will yield the "naive" Method A, requiring quadratic time. The second method is to improve performance by implementing an analog of binary search on touching permutations.

Let us now outline the "naive" Method A, and its improvement, Method B.

METHOD A. (QUADRATIC IN n) Given $a \sim (1, 2)$ and $b \sim (1, 2, \dots, n)$ as above, pre-compute elements of G corresponding to $a_2 \sim (2, 3), \dots, a_{n-1} \sim (n-1, n), a_n \sim (n, 1)$, the transpositions of neighboring points.

Given an $x \in G$. Denote $\bar{x} = \phi(x)$, the permutation to which $x \in G$ maps under our isomorphism. We will subsequently find the images of the points $1, 2, \dots, n-1$ under the permutation $\bar{x}$.

To find the image of 1 under $\bar{x}$, compute $x^{-1}ax$. It is a transposition, and in fact, $\phi(x^{-1}ax) = (\bar{x}(1), \bar{x}(2))$. By checking which of the n pre-computed transpositions above touch it (i.e. have supports that intersect with the set $\{\bar{x}(1), \bar{x}(2)\}$), we will know the set $\{\bar{x}(1), \bar{x}(2)\}$. Next, compute $x^{-1}a_2x$. We know that $\phi(x^{-1}a_2x) = (\bar{x}(2), \bar{x}(3))$. Check if $x^{-1}a_2x$ touches our pre-computed elements $a_i \in G$ corresponding to transpositions of neighboring points. Thus we will know both $\bar{x}(1)$ and $\bar{x}(2)$ precisely. Continue for all other points $3, \ldots, n-1$.

This process will furnish the permutation $\bar{x} = (\bar{x}(1), \bar{x}(2), \ldots, \bar{x}(n))$ in $O(n^2)$ multiplications and comparisons, $O(n)$ steps being required to find the image of each point under $\bar{x}$. Pre-computation of necessary transpositions will take no more than $O(\mu n \log n)$ time even if done in the most naive way. The storage requirement is that of storing these permutations.

Method B. (nearly linear in n) Given $a \sim (1,2)$ and $b \sim (1,2,\ldots,n)$ as above, pre-compute elements of G corresponding to $a_2 \sim (2,3), \ldots, a_{n-1} \sim (n-1,n), a_n \sim (n,1)$. To find the image of a point i under $\bar{x} = \phi(x)$ for some $x \in G$, compute $x^{-1}a_ix$, for which we know that $\phi(x^{-1}a_ix) = (\bar{x}(i), \bar{x}(i+1))$. As before, the actual identity of the transposition $\phi(x^{-1}a_ix)$ is found by checking which points it touches, via checking commutation relations with pre-computed elements of the group. We now perform a binary search on the set of all n points on which G acts, as follows.

Divide all points into two sets of lengths $k = n/2$ if $n = 2k$, or k and $k+1$ if $n = 2k+1$. Likewise, divide each one of these sets into two halves, and continue recursively until you have n single point sets. We have now formed the *search tree* for our search. Order the points in each set and pre-compute (starting from the elements a and b) the cycles that correspond to each set.

For $n = 2^m$ these are the following permutations (in the cycle notation):

$$\left(1, 2, \ldots, \frac{n}{2}\right), \qquad \left(\frac{n}{2}+1, \ldots, n\right),$$

$$\left(1, \ldots, \frac{n}{4}\right), \qquad \left(\frac{n}{4}+1, \ldots, \frac{n}{2}\right), \qquad \left(\frac{n}{2}+1, \ldots, \frac{3n}{4}\right), \qquad \left(\frac{3n}{4}+1, \ldots, n\right),$$

$$\cdots$$

$$(1,2), \qquad (3,4), \qquad \ldots, \qquad (n-1,n), \qquad (n,1).$$

Pre-computation of the lowest row will take soft $O(n)$ multiplications, computing the rest from the lowest row up (each time multiplying elements from the preceding row and transpositions from the lower row) will again take soft $O(n)$ multiplications. Note that we do not need these exact cycles, only the cycles with the same supports as those in the table. We now have a binary tree of permutations.

As before, take the conjugate $x^{-1}ax$, and, knowing that $\phi(x^{-1}ax) = (\bar{x}(1), \bar{x}(2))$, proceed with checking if it touches the group elements corresponding to one of the two cycles in the first row of the above table (the binary tree of cycles). For the cycle(s) it does touch check their descendants in the next row below. For each row the given permutation can touch no more than two of its cycles, therefore we will know the set $\{\bar{x}(1), \bar{x}(2)\}$ in at most $2 \log n$ checks, i.e. $O(\log n)$ multiplications and comparisons.

Repeat this procedure with $x^{-1}a_2x$ to find both $\bar{x}(1)$ and $\bar{x}(2)$, and continue for $3, \ldots, n-1$. The process of finding the permutation $\bar{x}$ corresponding to x will take $O(n \log n)$ steps, i.e. $O(\mu\, n \log n)$ time, with the requirement of simultaneously storing $2n$ elements of G.

5. Note on Provable Complexity

First, let us state the reasoning behind the EGC. The first part of the EGC is a slight extension of the GC as stated by Goldbach in his letter to Euler (June 7, 1742). Conjectures similar to the second part were made by Sylvester (1871), and subsequently refined by others, including Hardy and Littlewood (1922), see also Brun (1915) and Wang Yuan (1984), who conjectured the following asymptotic formula for $\pi_2(n)$:

$$\pi_2(n) \sim 2C_2 \frac{n}{\log(n)^2} \prod_{p|n} \frac{p-1}{p-2},$$

where the product is over the odd prime divisors p of n, and

$$C_2 = \prod_{p=3}^{\infty} \left(1 - \frac{1}{(p-1)^2}\right) \approx 0.6601618158,$$

where the product is over all odd primes. Thus asymptotically the lower bound in the EGC should be about $2\,C_2 \approx 1.320323632$ (see Hardy and Littlewood, 1922). Here we assume that $\pi_2(n)$ does not have large deviations. A similar result has been proved for decompositions of odd numbers into triples of primes (see below).

It is also known (see Tenenbaum, 1995, p. 78), that

$$\pi_2(n) \leq (8 + o(1))C_2 \frac{n \log\log n}{\log^2 n}.$$

We will need this bound in Section 6.

The reasoning behind the WGC is more speculative. It is based on a heuristic for the distribution of primes. In the Appendix we compute the average behavior of P_n and find lower bounds on P_n under a certain probabilistic model.

Now we would like to elaborate on the nature of our algorithm's dependence on the versions of the GC. The GC has been verified on numerous occasions, most recently on the Cray series, up to astronomically large[†] even numbers.

Further, it has been proved for "almost all" even n in the following Further, it has been proved for "almost all" even n in the following sense: for large N the number of even integers less than N for which the conjecture does not hold is of the order $N^{1-\epsilon}$ (Montgomery and Vaughan, 1975). Recently, the admissible value $\epsilon = 1/20$ was proved in Chen and Liu (1972). The original Hardy and Littlewood's argument gives $\epsilon = 1/2$, but it relies on the General Riemann Hypothesis (GRH), which remains unproven. For similar results for the asymptotic behavior of $\pi_2(n)$ and history of the GC see Hardy and Littlewood (1922), Wang Yuan (1984) and Ribenboim (1995).

Provided that the GC holds, our algorithm will work for all values of n, while its actual complexity will depend on the number and the distribution of pairs of primes p_1, p_2 such that $p_1 + p_2 = n, n-2$ (even n), or $p_1 + p_2 = n-1, n-5$ (odd n).

Assuming the EGC, the number of such pairs is no less than $\frac{1}{3}n/\log^2 n$, hence the complexity of the algorithm will be $O([\rho+\mu+\nu]n\log^2 n)$. Moreover, assuming the WGC, we obtain $O([\rho+\nu+\mu]\, n\log n/\log\log n)$ time complexity. According to the result in the Appendix, this is also the lower bound for the *average case* work of the algorithm.

[†]Jean-Mark Deshouillers, Yannick Saouter and Herman de Riele have verified this up to 10^{14} with the help of a Cray C90 and various workstations. See Ribenboim (1995) and Wang Yuan (1984) for more information.

It was proved by Vinogradov that every large enough odd number can be represented as a sum of three primes (see Vinorgadov, 1937; Wang Yuan, 1984). It has been recently demonstrated that the claim holds for all odd $n > 10^{7200}$ (see Chen and Wang, 1996). Assuming the GRH, the result was recently *proved* for all odd $n > 5$ (see Zinoviev, 1997; Saouter, 1998; Deshouillers *et al.*, 1997). In a different direction, it has been shown (see Vinorgadov, 1954, and Karatsuba, 1992) that for any $\epsilon > 0$ the number $\pi_3(n)$ of such presentations satisfies

$$\pi_3(n) = C_3 \frac{n^2}{\log^3 n} \prod_{p|n} \frac{(p-1)(p-2)}{p^2-3p+3} + O\left(\frac{n^2}{\log^{3.5-\epsilon}}\right),$$

where

$$C_3 = \prod_{p=3}^{\infty}\left(1 - \frac{1}{(p-1)^3}\right) = 0.8553921037.$$

The main term of the above asymptotic formula was first discovered and proved by Hardy and Littlewood, but their proof relies on the GRH. Now observe that

$$\prod_{p|n} \frac{(p-1)(p-2)}{p^2-3p+3} > \prod_{p=3}^{\infty} \frac{(p-1)(p-2)}{p^2-3p+3} \approx 0.5738139342.$$

This gives us the following result.

Theorem. (Vinogradov) For sufficiently large odd n we have

$$\pi_3(n) > 0.49 \frac{n^2}{\log^3 n}.$$

If we modify our algorithm so that instead of Goldbach elements of cycle forms (p_1, p_2) it uses elements of cycle forms (p_1, p_2, p_3), we obtain an algorithm which, for large enough n, has time complexity $O([\rho + \nu + \mu]\, n \log^3 n)$. Note, however, that it is unclear whether such an algorithm would work for *all* odd n.

Going one step further, for six primes it *is* known (see Ramaré, 1995) that for all even $n > 10$ there is a presentation $n = p_1 + \cdots + p_6$. Consequently, another modification of our algorithm will give an algorithm *provably* succeeding in $O(n^6)$ time. One can use this modification for n smaller than the constant predicted by Vinogradov's theorem. For larger n one can use the previous version. This proves the following theoretical result.

Theorem 3. *Provided that n is known, there exists a Las Vegas algorithm, which in time $O([\rho + \nu + \mu] n \log^3 n)$ can produce a data structure which can then be used to compute an isomorphism $\phi : G \to S_n$.*

Note that the constant implied by $O(\cdot)$ notation is expected to be very large, so the theorem is only of theoretical interest.

We can combine the above arguments into a single algorithm, in which for the given n we find its representation as a sum of a number of distinct primes, thus locating an element consisting of primes cycles of different lengths (Goldbach elements are a specific case of these). Using these elements instead of only the Goldbach elements, we can expect a somewhat faster algorithm. However, for the sake of brevity, we will not go into details of this, since the latter should not pose any significant conceptual problems over the presented algorithm.

To summarize, we have chosen the venue which is at the same time efficient and easy to present, although not formally proven.

6. Computing the Orders of Elements

Observe that throughout the algorithm we use the order $r(x)$ of a random element $x \in G$ for the sole purpose of verifying whether $r(x) = p_1 p_2$, where $p_1 + p_2 = n$ for $n = 2k$, or $p_1 + p_2 = n - 1$ for $n = 2k + 1$, and for verifying two similar properties. All these properties revolve around the Goldbach pairs of primes p_1, p_2 for n and $n - 1$ (n even), or $n - 1$ and $n - 5$ (n odd).

If the Goldbach pairs p_1, p_2 are pre-computed for these integers, these properties of $x \in G$ can be tested by computing $x^{p_1 p_2}$ and testing it for being the identity, together with its component cycles x^{p_1} and x^{p_2}. Each such test involves $O(\log p_1 + \log p_2)$ multiplications for computing the power and a black box identity test (which is assumed to take time comparable with a multiplication), i.e. no more than $O(\mu \log n)$ complexity per pair (cf. Pak, 1998).

Recall from Section 5 that there are at most $O(n \log\log n / \log^2 n)$ Goldbach pairs available for any n. Therefore we need $O(\mu\, n \log\log n / \log n)$ time for testing each random element for the Goldbach property. This estimate takes the place of ν in the algorithm's overall complexity when no efficient trick for order computation is available. In other words, one can always assume that

$$\nu = O\left(\frac{\mu n \log\log n}{\log n}\right).$$

7. Modification for the Alternating Group A_n

It turns out that the Goldbach conjecture approach can be applied to the recognition of a gray box group G known to be isomorphic to A_n. Indeed, A_n for n odd is generated by a cycle of length 3 and a long cycle. For all odd primes p_1, p_2, the corresponding Goldbach elements of S_n actually lie inside A_n. Therefore we can first locate a Goldbach element in A_n and break it down into two prime cycles by raising it to corresponding powers.

Then we find a 3-cycle using a technique similar to that of Section 2.1. Once a 3-cycle is found, we can obtain any number of (nearly) uniformly distributed random 3-cycles by conjugating it with random elements of the group G. We then find a 3-cycle that touches both cycles of the Goldbach element. The product of these and the found 3-cycle gives us a long cycle in G. Re-numbering the points accordingly, we arrive at two elements $a, b \in G$ such that there is an isomorphism $\psi : G \to A_n$ carrying a to $(1, 2, 3)$ and b to $(1, 2, \ldots, n)$.

The case of $G \cong A_{2k}$ involves some additional work. In this case we need to consider not one but two fixed points and connect them with our Goldbach cycles of lengths p_1, p_2. Let us be more careful. First, locate an element x of order $p_1 \cdot p_2$, where $p_1 + p_2 + 2 = 2k$, $p_1 > p_2 \geq 5$. Then $x_1 = x^{p_2}$ and $x_2 = x^{p_1}$ are two cycles of lengths p_1 and p_2, respectively. Similarly locate a 3-cycle y. Consider random conjugates y^g of y until we find a 3-cycle z which does not commute with both cycles x_1 and x_2 (and therefore touches both of them). Now proceed with constructive recognition of A_{2k-1} generated by x_1, x_2 and z (remember that they all fix one point). After this is done, consider again random conjugates y^g until we find a 3-cycle that touches the remaining point. Use version of the algorithm in Section 4 to determine the remaining two vertices in the 3-cycle and complete the recognition. We omit the technical details.

8. Checking the Isomorphism between G and S_n for a Known n

In this section we present a one-sided Monte Carlo version of our algorithm for verifying that G is isomorphic to S_n for a known n. A positive answer is guaranteed to be correct, while a negative answer may be incorrect, with probability of error $\varepsilon > 0$ that can be made as small as desired.

Assume that $G \cong S_n$, and start, as before, the search for a Goldbach element. If no Goldbach element is found after a sufficient number $N(\varepsilon, n)$ (discussed below) of random elements have been tried, the algorithm stops and reports a negative answer. The same happens during the search for a transposition. If both searches are successful, we have elements $a, b \in G$.

Check if a, b satisfy the Moore's defining relations for S_n (see Coxeter and Moser, 1972), namely

$$b^n = a^2 = (ba)^{n-1} = (ab^{-1}ab)^3 = (ab^{-j}ab^j)^2 = id, \qquad 2 \le j \le n-2.$$

This can be done in $O(\mu n)$ time. If this is the case, the mapping $\psi : S_n \to G$ defined by $\psi(1\ 2) = a$, $\psi(1,\ 2, \ldots,\ n) = b$ defines a homomorphism of S_n onto its image $G_0 = \langle a, b\rangle \subset G$. Being a homomorphic image of S_n, G_0 can only be one of S_n, Z_2 or $\{id\}$, and excluding the latter two cases for $G = \langle a, b\rangle$ is trivial. Therefore $G_0 \cong S_n$.

To demonstrate $G \cong S_n$ we need to check that $G_0 = G$. Given $G = \langle g_1, \ldots, g_s\rangle$, we need to express the original generators $g_1, \ldots, g_s$ as words in a, b. Note that the cycle tree used in the second phase of our algorithm is constructed entirely inside G_0. Running the second phase for g_i, $i = 1, \ldots, s$ we obtain some permutations $\gamma_i \in S_n$, $i = 1, \ldots, s$. Represent these permutations as words in the generators $(1\ 2)$, $(1,\ 2,\ \ldots,\ n)$ of S_n and construct the images $\psi(\gamma_i)$, $i = 1, \ldots, s$. If $\psi(\gamma_i) = g_i$, we have the desired isomorphism, if not, we can claim that $S_n \cong G_0 \subsetneq G$. This step of verification procedure takes $O(\mu\, s\, n \log n)$ time.

Let us estimate the number $N(\varepsilon, n)$ of trials that fail to produce a Goldbach element, after which we can claim, with probability of error less than ε, that the group G is not isomorphic to S_n. We need at most

$$N(\varepsilon, n) = 2 \left\lceil \log \frac{1}{\varepsilon} \right\rceil \frac{1}{P_n}$$

trials. Indeed, if $G \cong S_n$, then after $2/P_n$ trials the probability of finding a Goldbach element is at least $1/2$. Therefore after $2C/P_n$ trials the latter probability is at least $1 - 1/2^C$. Taking $C = \lceil \log \frac{1}{\varepsilon} \rceil$ proves the claim. A similar estimate holds for the number of unsuccessful trials while searching for a transposition in G. Using the estimate of P_n from the EGC and WGC, we obtain the following result.

THEOREM 4. *Let $G = \langle g_1, \ldots g_s\rangle$ be a gray box group. Under the definitions of Theorem 1, and provided that EGC holds, a one-sided Las Vegas algorithm exists which, in* *time*

$$O\left([\rho+\mu+\nu]\log\left(\frac{1}{\varepsilon}\right) n\,\log^2 n + \mu\, s\, n \log n\right)$$

either proves that G is isomorphic to S_n or rejects this hypothesis with probability of error ε.

Moreover, provided that WGC holds, the above algorithm completes in

$$O\left([\rho+\mu+\nu]\log\left(\frac{1}{\varepsilon}\right) \frac{n\log n}{\log\log n} + \mu\, s\, n \log n\right).$$

9. What To Do If n is Not Known?

The algorithm presented so far is Las Vegas only if we know the value of n, *a priori.* Let us present a Monte Carlo way of finding an initially unknown n (which can be made Las Vegas at the cost of admitting large storage requirements and significant increase in complexity). We still assume that our gray box group G is isomorphic to S_n for some n.

We assume that there exists a natural upper bound for n. Such an upper bound arises when the elements of the gray box group are encoded with bit strings of equal (finite) length M. In that case we know that $n < N := f(2^M)$, where f is the inverse factorial. For simplicity assume N is even.

Denote by $\kappa(N)$ the maximum of $1/P_{2i}$ over all $i = 1,\dots,N/2$. Take a sequence of $2\kappa(N)$ random elements of G, and for every random element x compute its order

$$r(x) = p_1^{k_1}\cdot p_2^{k_2}\cdot\,\cdots\,\cdot p_m^{k_m}.$$

Take the maximum of sums

$$S = p_1 + p_2 + \cdots + p_m$$

over all these x. It is easy to see that $S \le n$, and equals n exactly when the element x consists of cycles of distinct prime lengths $p_1, p_2, \dots, p_m$ (and no fixed points). In particular, if x is a Goldbach element and n is even, we obtain equality, and if n is odd, then $S = n-1$. We claim that after $2\kappa(N)$ steps the probability of obtaining $S = n$ or $S = n-1$ is at least $1/2$. Indeed, the expected number of steps before a Goldbach element is encountered is at most $1/P_n$ when n is even, or $1/P_{n-1}$, when n is odd. This, and the Markov bound immediately imply the above estimate.

Deciding whether S, the estimated degree of the group, is equal to n or $n-1$ requires more time and involves looking for elements of order $3p_1p_2$, where $p_1+p_2+3 = S+1$. If no such elements were found after $6\kappa(S)$ steps, we can conclude, with the probability of error $< 1/2$, that $S = n$. Repeating this k times will improve the probability of error to $1/2^k$.

Now recall that by the EGC we have $\kappa(N) = O(N\log^2 N)$. Analogously by the WGC we obtain $\kappa(N) = O(N\log N/\log\log N)$. This finishes the proof of the first part of Theorem 2.

The above Monte Carlo algorithm outputs a number S which is smaller or equal to the actual degree n of the group. Starting with this number, we obtain an element a of G which is conjectured to be a transposition (if $S = n$, it is provably a transposition). Now we can use original generators of our group G to generate the entire conjugacy class of a, which is supposedly that of all transpositions in $G \cong S_n$. The conjugacy class formed by transpositions in S_n is the smallest non-trivial class, containing $\binom{n}{2}$ elements. If the class we have generated is not of this size, then either a is not a transposition, or we have

Table 1.

n	R_{average}	$\pi_2(n)$	$\pi_2(n-2)$	t, s
20	121	2	2	0.045
30	224	3	2	0.135
40	468	3	2	0.375
50	201	4	5	0.208
60	377	6	4	0.459
70	847	5	2	1.207
80	474	4	7	0.775
90	558	9	4	1.040
100	1625	6	3	3.338
110	527	6	8	1.193
120	684	12	6	1.704
130	1971	7	3	5.310
140	923	7	8	2.682
150	1618	12	5	5.069
160	1507	8	5	5.055
170	849	9	13	3.020
180	1072	14	7	4.042
190	1627	8	5	6.521
200	779	8	13	3.302

Table 2.

n	R_{average}	$\pi_2(n)$	$\pi_2(n-2)$	t, s
25	586	3	2	0.312
35	422	4	3	0.309
45	1102	3	3	1.028
55	1111	5	4	1.240
65	841	5	6	1.131
75	1042	5	5	1.609
85	1506	8	4	2.624
95	1140	5	9	2.209
105	2011	5	6	4.344
115	2749	10	6	6.537
125	1267	5	12	3.271
135	3937	6	7	11.018
145	3659	11	7	11.082
155	2011	8	12	6.498
165	3650	5	8	12.64
175	2561	11	9	9.417
185	1793	8	14	6.992
195	3730	7	8	15.34

$S < n$. In either case, the conjugacy class of a is larger than $\binom{n}{2}$. This computation is hardly practical, as it will take more than quadratic time in n.

10. Implementation of the Algorithm

The algorithm has been implemented in the computational group theory system GAP, and can be obtained from the authors. We have tested the performance of our GAP im-

Table 3.

n	R_{average}	$\pi_2(n)$	$\pi_2(n-2)$	t, s
2310	3309	114	34	369.13
2860	11367	68	38	1793.7

plementation on a Pentium II 400 MHz PC with 128 M RAM, running Linux 2.0.33. The algorithm was tested on a family of symmetric groups of various degrees and representations, as well as on sporadic examples of groups non-isomorphic to S_n. A full discussion of implementation will be given in Bratus and Pak (1999).

In Tables 1 and 2 we present partial test run results for two families of symmetric groups. For each group we give the number R_{average} of times a random element of the black box group was sampled before the long cycle and the transposition were found, and the CPU time in seconds, averaged over 100 runs. The former statistic is central to the randomized part of the algorithm, as it clearly dominates the complexity. The running time of the randomized part for S_n, $n = 2k$, is strongly affected by the numbers $\pi_2(n)$ and $\pi_2(n-2)$ of Goldbach pairs, and therefore we give these numbers for the groups in Table 1. For Table 2, i.e. for the family S_n, $n = 2k+1$, the same role is played by $\pi_2(n-1)$ and $\pi_2(n-5)$. These numbers, connected with the probability of finding a Goldbach element, explain why the algorithm may take a longer time to complete for a smaller value of n. We remark here that the odd case algorithm is normally slower by a factor of about 2. This factor can be explained by the fewer numbers of Goldbach elements, and, consequently, the larger number of times the random sampling procedure was called.

In Table 3 we present similar statistics for two large examples. The latter examples were run only once, because of their larger size. The numbers $2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$ and $2860 = 2^2 \cdot 5 \cdot 7 \cdot 11$ were chosen because of the large number of their prime factors. We should warn the reader that the running time of the algorithm is subject to high deviations, so the data in Table 3 may not represent the average case.

We chose not to report here the results of test runs on groups non-isomorphic to S_n (in which the groups were rejected by the algorithm). We refer the reader to Bratus and Pak (1999) for details.

Acknowledgements

We are grateful to Gene Cooperman and Larry Finkelstein for introducing us to Symbolic Algebra in general and to the group recognition problem in particular. Their attention and helpful discussions were invaluable. Special thanks to Gene Cooperman for reading the manuscript and help with several sections of the paper. We would also like to thank Bill Kantor for stimulating conversations, Serge Lang and Andrew Odlyzko for help with the references. We thank the referees for many suggestions on the subject matter and the presentation of the paper.

The Appendix was added when both authors attended the Summer Probability Program at the University of Wisconsin. We are thankful to Eric Bach for his heuristic idea, which led us to Theorem 4 and the WGC.

The first author was partially supported by NSF Grant CCR-9509783. The second author would like to thank his graduate advisor Persi Diaconis for giving him a feel for randomness, and to the Fannie and John Hertz Foundation for the financial support during his graduate study at Harvard University.

References

Babai, L. (1991). Local expansion of vertex-transitive graphs and random generation in finite groups. In *Proc. 23rd ACM Symposium on the Theory of Computing*, pp. 164–174.

Babai, L. (1997). Randomization in group algorithms: conceptual questions. In *Groups and Computation II*, DIMACS Series in Discrete Mathematics and Computer Science Finkelstein, L., Kantor, W. M. eds., **28**, pp. 1–17. Providence, RI, American Mathematical Society.

Beals, R., Babai, L. (1993). Las Vegas algorithms for matrix groups. In *Proceeding 24th IEEE FOCS*, pp. 427–436.

Bratus, S., Pak, I. (1999). Implementing the Goldbach algortihm for recognition of symmetric groups, in preparation.

Brun, V. (1915). Uber das Goldbachshe Gesatz und die Anzahl der Primzahlpaare. *Arch. Math. Naturvid B* **34**, 1–15.

Cameron, P. J., Cannon, J. (1991). Fast recognition of doubly transitive groups, computational groups theory, part 2. *J. Symb. Comput.*, **12**, 459–474.

Celler, F., Leedham-Green, C. R. (1997a). A non-constructive recognition algorithm for the special linear and other classical groups. In *Groups and Computation I*, DIMACS Series in Discrete Mathematics and Computer Science Finkelstein, L., Kantor, W. M. eds., **28**, pp. 61–68. Providence, RI, American Mathematical Society.

Celler, F., Leedham-Green, C. R. (1997b). Calculating the order of an invertible matrix. In *Groups and Computation II*, DIMACS Series in Discrete Mathematics and Computer Science Finkelstein, L., Kantor, W. M. eds., **28**, pp. 55–60. Providence, RI, American Mathematical Society.

Celler, F., Leedham-Green, C. R., Murray, S. H., Niemeyer, A. C., O'Brien, E. A. (1995). Generating random elements of a finite group. *Commun. Algebra*, **23**, 4931–4948.

Chandrasekharan, K. (1968). *Introduction to Analytic Number Theory*, Berlin, Springer.

Chen, J. R., Liu, J. (1989). The exceptional set of Goldbach-numbers. III. *Chin. Q. J. Math.*, **4**, 1–15.

Chen, J. R., Wang, T. Z. (1996). The Goldbach problem for odd numbers. (Chinese, English summary). *Acta Math. Sin.*, **39**, 169–174.

Cooperman, G., Finkelstein, L. (1998). Topics in computing with large groups of matrices, preprint.

Cooperman, G., Finkelstein, L., Linton, S. (1997). Constructive recognition of a black box group isomorphic to $GL(n, 2)$. In *Groups and Computation II*, DIMACS Series in Discrete Mathematics and Computer Science Finkelstein, L., Kantor, W. M. eds., **28**, pp. 85–100. Providence, RI, American Mathematical Society.

Coxeter, H. S. M., Moser, W. O. J. (1972). *Generators and Relations for Discrete Groups*, 3rd edn. New York, Springer.

Deshouillers, J.-M., Effinger, G., te Riele, H., Zinoviev, D. (1997). A complete Vinogradov 3-primes theorem under the Riemann hypothesis. *ERA Am. Math. Soc.*, **3**, 99–104.

Diaconis, P., Saloff-Coste, L. (1998). Walks on generating sets of groups. *Invent. Math.*, **134**, 251–299.

Hardy, G. H., Littlewood, J. E. (1922). Some problems of "Partitio Numerorum"; III: on the expression of a number as a sum of primes. *Acta Math.*, **44**, 1–70.

Hardy, G. H., Wright, J. E. (1960). *Basic Analytic Number Theory*, 4th edn. Oxford, Clarendon Press.

Kantor, W., Seress, A. (1997). Black Box classical groups, preprint.

Karatsuba, A. A. (1992). *Basic Analytic Number Theory*, Berlin, Springer.

Montgomery, H. C., Vaughan, R. C. (1975). The exceptional set in Goldbach problem. *Acta Arith.*, **27**, 353–370.

Neumann, P. M., Praeger, C. E. (1992). A recognition algorithm for special linear groups. *Proc. London Math. Soc.*, **65**, 555–603.

Niemeyer, A. C., Praeger, C. E. (1997). Implementing a recognition algorithm for classical groups. In *Groups and Computation II*, DIMACS Series in Discrete Mathematics and Computer Science Finkelstein, L., Kantor, W. M. eds., **28**, pp. 273–296. Providence, RI, American Mathematical Society.

Niemeyer, A. C., Praeger, C. E. (1998). A recognition algorithm for classical groups over finite fields. *Proc. London Math. Soc. (3)*, **77**, 117–169.

Pak, I. (1998). When and how n chose k. In *Randomization Methods in Algorithmic Design*, DIMACS Series in Discrete Mathematics and Computer Science, Pardalos, P., Rajasekaran, S., Roloim, J. eds., **43**, pp. 191–238. Providence, RI, American Mathematical Society.

Ramaré, O. (1995). On Schnirelman's constant. *Ann. Scuola. Norm. Sup. Pisa Cl. Sci.*, **22**, 645–706.

Ribenboim, P. (1995). *The New Book of Prime Numbers*, New York, Springer.

Rosser, J. B., Schoenfeld, L. (1962). Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, **6**, 64–94.

Saouter, Y. (1998). Checking the odd Goldbach conjecture up to 10^{20}. *Math. Comput.*, **67**, 863–866.

Schönert, M. *et al.* (1995). *GAP—Groups, Algorithms, and Programming*, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, 5th edn. Germany, Aachen.

Sims, C. C. (1971). Computation with Permutation Groups. In *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, Petrick, S. R. ed., pp. 23–28. New York, ACM.

Sylvester, J. J. (1871). On the partition of an even number into two primes. *Proc. London Math. Soc.* Series 1, **4**, 4–6.

Tenenbaum, G. (1995). *Introduction to Analytic and Probabilistic Number Theory*, Cambridge, UK, Cambridge University Press.

Vinogradov, I. M. (1937). Representation of an odd number as a sum of three primes. *Dokl. Akad. Nauk SSSR*, **15**, 291–294.

Vinogradov, I. M. (1954). *The Method of Trigonometrical Sums in the Theory of Numbers*, New York, Interscience Publishers.

Whittaker, E. T., Watson, G. N. (1927). *A Course of Modern Analysis.* 4th edn, Cambridge, UK, Cambridge University Press.

Yuan, W. (1984). *Goldbach Conjecture*, Singapore, World Scientific Publishing Co.

Zinoviev, D. (1997). On Vinogradov's constant in Goldbach's ternary problem. *J. Number Theory*, **65**, 334–358.

11. Appendix:[†] On the Fraction of Goldbach Elements in S_n

Throughout the paper we assumed that finding the Goldbach elements can be done fairly fast. The EGC implies that this can be done in time $O([\rho+\nu+\mu]\, n\log^2 n)$, which is the dominating term in the running time of our algorithm. The purpose of this Appendix[†] is to give a basis for the WGC, which implies that the running time of the algorithm is $O([\rho+\nu+\mu]\, n\log n/\log\log n)$.

11.1. THE LOWER BOUND

Recall that

$$P_n = \sum_{p+q=n} \frac{1}{p\,q}.$$

Everywhere in this section, by p, q we mean prime numbers. In the algorithm we also required $p \neq q$ in the summation, but for large n the difference in P_n is marginal and will be ignored.

THEOREM 5. *For even $N \to \infty$ we have*

$$\frac{1}{N/2}(P_{N+2} + P_{N+4} + \cdots + P_{2N}) \sim \frac{\log 2 \cdot \log\log N}{N \log N}.$$

This shows that for *average* even $n \in [N+2, 2N]$, the expected number of tries to find a Goldbach element is $\Omega(n\log n/\log\log n)$. This gives a matching lower bound on the running time of the algorithm.

PROOF. We need several definitions and known results. Define

$$S(a,b) = \sum_{a/2<i\le b/2} P_{2i} = \sum_{a<p+q\le b} \frac{1}{pq}.$$

[†]The Appendix is due to the second author.

We need to estimate $S(N/2,N)$. If we show that

$$S(N/2,N) \sim \frac{\log 2 \log\log n}{\log n}$$

this would immediately imply the result.

Recall the Euler–Mascheroni constant

$$\gamma = \lim_{N\to\infty}\left(1+\frac{1}{2}+\cdots+\frac{1}{N}\right) - \log N \approx 0.5772156649$$

(see e.g. Hardy and Wright, 1960, and Tenenbaum, 1995), and Mertens constant

$$\beta = \gamma + \sum_p \left(\log\left(1-\frac{1}{p}\right)+\frac{1}{p}\right) \approx 0.2614972128.$$

Versions of the following classical result go back to Euler, Mertens, Hadamard and de la Vallée Poussin (see Hardy and Wright, 1960, and Tenenbaum, 1995):

$$\sum_{1<p\le N}\frac{1}{p} = \log\log N + \beta + o(1).$$

Furthermore, for $N \ge 286$ we have

$$\sum_{1\le p\le N}\frac{1}{p} = \log\log N + \beta + \frac{\tau(N)}{\log^2 N},$$

where $-1/2 < \tau(N) < 1/2$. The latter result is due to Rosser and Schoenfeld (1962).

By analogy with $S(a,b)$, define

$$T(a,b) = \sum_{a<p\le b}\frac{1}{p}.$$

For any $\alpha, \epsilon > 0$ and N large enough we have

$$\begin{aligned} T(\alpha N, (\alpha+\epsilon)N) &= \log\log(\alpha+\epsilon)N - \log\log\alpha N + O\left(\frac{1}{\log^2 N}\right) \\ &= \int_{\alpha N}^{(\alpha+\epsilon)N}\frac{dx}{x\log x} + O\left(\frac{1}{\log^2 N}\right). \end{aligned}$$

Now observe that given $\alpha_1, \alpha_2 > 0$ and $\epsilon \to 0$ we have

$$\sum_{\substack{\alpha_1 N<p\le(\alpha_1+\epsilon_1)N \\ \alpha_2 N<q\le(\alpha_2+\epsilon_2)N}}\frac{1}{pq} = T(\alpha_1 N, (\alpha_1+\epsilon_1)N)\cdot T(\alpha_2 N, (\alpha_2+\epsilon_2)N).$$

This gives us

$$S(a,b) = \int\int_D \frac{dx}{x\log x}\frac{dy}{y\log y} + O\left(\frac{S(1,N)}{\log^2 N}\right)$$

where $D = D(a,b) = \{(x,y)\,|\,2\le x,y\le N, a<x+y\le b\}$.

From Green's formula we have

$$\begin{aligned} \int\int_D \frac{dx}{x\log x}dxx\log x &= \frac{1}{2}\int_{\partial D}\log\log x\, d\log\log y - \log\log y\, d\log\log x \\ &= A_1 + A_2 + A_3 + A_4. \end{aligned}$$

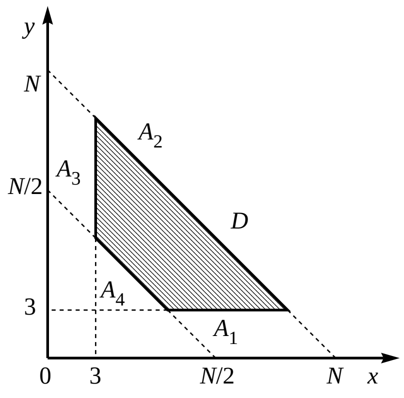

Figure 1.

(See Figure 1 below.) An explicit computation gives

$$A_1 = A_3 = O\left(\frac{1}{\log N}\right)$$
$$A_2 + A_4 \sim \frac{\log 2 \log\log N}{\log N}.$$

For the error term we have

$$S(1,N) = \sum_{p+q\le N} \frac{1}{pq} < \sum_{1<p,q\le N} \frac{1}{pq}(\beta + \log\log N)^2 + O\left(\frac{\log\log N}{\log^2 N}\right) = O((\log\log N)^2).$$

This finishes proof of the theorem.□

11.2. PROBABILISTIC MODEL

The probabilistic method in number theory dates back to Brun and perhaps even earlier. While sometimes informal, this approach often gives answers of the right order of magnitude. In this century, a spectacular progress was made by Erdös, Turán and subsequent investigators, who proved a number of rigorous results (see Tenenbaum, 1995).

Consider now the following simple randomized model for prime numbers. For all $k = 2, 3, \ldots$ let $X_k \in \{0,1\}$ be independent, and $X_k = 1$ with probability $1/\log k$. Denote $S_n = X_2 + \cdots + X_n$. Heuristically, $X_n = 1$ if n is a prime. We have

$$E(S_n) = \sum_{k=2}^{n} E(X_k) = \sum_{k=2}^{n} \frac{1}{\log n} = \mathrm{Li}\, x + O(1),$$

where

$$\mathrm{Li}\, x = \int_2^n \frac{dx}{\log x} \sim \frac{n}{\log n}.$$

Thus $E(S_n)$ roughly agrees with the asymptotic law of prime numbers.

Since X_i are chosen to be independent, we can compute the variance

$$\mathrm{Var}(S_n) = \sum_{k=2}^{n} \mathrm{Var}(X_k) < \sum_{k=2}^{n} \frac{1}{\log n} = \mathrm{Li}\, x + O(1).$$

This gives an error term of the order $\sqrt{n/\log n}$, which roughly agrees with the error term in the asymptotic law of prime numbers predicted by Riemann Hypothesis (see Tenenbaum, 1995).

Here is a heuristic behind the asymptotic formula for the $\pi_2(n)$. We need to estimate the behavior of the following random variable:

$$Z_n = X_2 \cdot X_{n-2} + X_3 \cdot X_{n-3} + \cdots + X_{n/2-1} \cdot X_{n/2+1},$$

where n is even. We have

$$\begin{aligned} E(Z_n) &= \sum_{k=2}^{n/2-1} E(X_k \cdot X_{n-k}) = \sum_{k=2}^{n/2-1} \frac{1}{\log k \log(n-k)} \\ &\sim \frac{1}{\log(n)} \sum_{k=2}^{n/2-1} \frac{1}{\log k} = \frac{n}{2\log^2 n}(1+o(1)). \end{aligned}$$

One can use Chernoff bounds to show that Z_n is concentrated around its mean. Similar analysis can and has been done for $\pi_3(n)$, etc. (see Brun, 1915).

To justify the WGC, we need to find similar bounds for P_n. We are interested in the behavior of the following random variable:

$$R_n = \frac{1}{2(n-2)} X_2 \cdot X_{n-2} + \frac{1}{3(n-3)} X_3 \cdot X_{n-3} + \cdots + \frac{1}{(n/2-1)(n/2+1)} X_{n/2-1} \cdot X_{n/2+1},$$

where n is even. Heuristically, R_n is the analog of P_n, i.e. a proportion of "Goldbach elements" in our randomized model. We have

$$E(R_n) = \sum_{k=2}^{n/2-1} \frac{E(X_k \cdot X_{n-k})}{k\,(n-k)} = \sum_{k=2}^{n/2-1} \frac{1}{k\,(n-k) \log k \log(n-k)}.$$

Using $n/2 < n-k < n$ we obtain

$$E(R_n) = \theta\left(\frac{1}{n\log n}\right) \cdot \sum_{k=2}^{n/2-1} \frac{1}{k\log k} = \theta\left(\frac{\log\log n}{n\log n}\right),$$

which gives the same expected order of R_n as of P_n in the WGC.

It is important to note, however, that the average running time of the algorithm in this model is given by $E(1/R_n) = \infty$ since with non-zero probability we have $X_2 \cdot X_{n-2} = X_3 \cdot X_{n-3} = \cdots = 0$. The probability of this event is $c\,2^{-n/2}$. In all other cases we have $1/R_n = O(n^2)$. The analog of the EGC in this case is the claim that $1/R_n < 3/4n\log^2 n$ for most choices $X_1, \ldots, X_n$. Indeed, if with high probability there are at least $n/3\log^2 n$ choices of i such that $X_i \cdot X_{n-1} = 1$, then we always have $1/i(n-i) > 4/n^2$. This gives us $R_n > 4/n^2 \cdot n/3\log^2 n > 3/4n\log^2 n$. On the other hand, in the best possible case we have $X_2 = \cdots = X_{n-2} = 1$ and thus

$$R_n \geq \sum_{i=2}^{n/2-1} \frac{1}{i(n-i)} = \frac{1}{n} \sum i = 2^{n/2-1} \frac{1}{i} + \frac{1}{n-i} = \frac{\log n + O(1)}{n}.$$

This shows that $1/R_n > n/\log n\,(1+o(1))$ for any choices of X_i.

THEOREM 6. *If R_n is as above, for sufficiently large n we have*

$$R_n > \frac{\log\log n}{8\,n\log n}$$

with probability $> 1 - e^{-e^{\sqrt{\log n}/2}}$.

PROOF. For simplicity we will omit the floor/ceiling notation. Break an interval $[2, n/2]$ into $r = n/2m$ intervals of length m each. Call these intervals $I_1, \ldots, I_r$. For all $k = 1, \ldots, r$ consider the number a_k of $i \in I_k$ such that $X_i = X_{n-i} = 1$. By definition, $Y_i = X_i \cdot X_{n-i}$, $i = 1, \ldots, r$ are independent Bernoulli trials with

$$P(Y_i = 1) = \frac{1}{\log i \log(n-i)} > \frac{1}{\log n \log km}$$

given $i \in I_k$. Therefore

$$E(a_k) > \frac{m}{\log n \log km}.$$

The Chernoff bound gives

$$P(a_k > E(a_k)/2) > 1 - e^{-m/8(\log n)^2}.$$

Therefore with probability $Q > 1 - ne^{-m/8(\log n)^2}$ we have $a_i > E(a_k)/2$ for all $i = 1, \ldots, n/2$. But in this case

$$\begin{aligned} R_n &= \sum_{j=2}^{n/2-1} \frac{X_j \cdot X_{n-j}}{j(n-j)} > \sum_{k=1}^{r} \frac{E(a_k)/2}{km\,(n-km)} \\ &> \sum_{k=1}^{r} \frac{1}{2\log(km)\log n\,k\,n} = \frac{1}{2\,n\log n}\sum_{k=1}^{r}\frac{1}{k(\log k + \log m)}. \end{aligned}$$

Given $m < r$ the latter sum is bounded by

$$\sum_{k=m}^{r} \frac{1}{k(\log k + \log m)} > \sum_{k=m}^{r} \frac{1}{2k\log k} = \frac{1}{2}(\log\log r - \log\log m).$$

Recall that $r = n/2m$. Let $m = e^{\sqrt{1/2\log(n/2)}}$. We have $\log\log m < 1/2 \log\log r$ then, and

$$R_n > \frac{1}{2\,n\log n}\frac{1}{4}\log\log n = \frac{1}{8}\frac{\log\log n}{n\log n}$$

with probability

$$Q > 1 - ne^{-m/8(\log n)^2} > 1 - ne^{-e^{\sqrt{1/2\log(n/2)}}/8(\log n)^2} > 1 - e^{-e^{\sqrt{\log n}/2}}$$

when n is large enough. This proves the result.□

REMARK. Let us conclude by saying that not being number theorists, we do not attempt to make any progress in the WGC. We believe, however, that the following two weaker versions are probably doable by the current techniques:

- The WGC holds for all but N^ϵ even numbers $n < N$.

- The analog of the WGC for triples of primes $p_1 + p_2 + p_3 = n$ holds for all large enough odd n.

Originally Received 29 September 1997
Accepted 27 September 1999