Symmetric functions, II:

Rhinofish Facth

Computing Science Institute/

CSI-R9805 January 1998
Symmetric functions, II:

- The Finite Case -

Rhinefish Facth

March 23, 1998

Abstract

An elementary combinatorial version of the symmetric function theorem leads to an interesting Diophantine equation. It is proved that the only finite field admitting such a theorem is GF(2).

Keywords: symmetric functions, finite sets, finite fields, combinational circuit complexity, diophantine equations, data compression.

1 INTRODUCTION.

In this article, we shall strictly distinguish between symmetric functions and symmetric polynomials. In particular, we shall talk about the "Symmetric Polynomial Theorem", meaning the well-known but incorrectly named Symmetric Function Theorem.
The symmetric group on \( n \) symbols acts in a natural way on the polynomials in \( x_1, \ldots, x_n \) over a field \( k \). The one-point orbits are called the symmetric polynomials. Special examples are the elementary symmetric polynomials \( \alpha_i = \sum_{j_1 < j_2 < \cdots < j_i} x_{j_1} x_{j_2} \cdots x_{j_i} \). The Symmetric Polynomial Theorem states that any symmetric polynomial can be written uniquely as a polynomial in the \( \alpha_i \)'s, which thus are algebraically independent [13], [14].

Let the symmetric functions from \( k^n \) to \( k \) be those functions which are invariant under the symmetric group acting as a group of coordinate permutations. Any polynomial defines a function from \( k^n \) to \( k \). If \( k \) is infinite, this is injective; hence in that case the symmetric polynomial functions can be expressed uniquely as polynomial functions of the elementary symmetric functions defined by the \( \alpha_i \)'s. When \( k \) is finite, the situation is very different, as one already sees from the identity \( \alpha_1^2 = \alpha_1 \) as functions mod 2.

We are interested in the unique representation of symmetric functions on finite sets in terms of "elementary" functions. In fact, as we will show, this reduces to a nice Diophantine equation, generalizing known equations incorporating triangular and tetrahedral numbers [4].

There are some connections with Boolean complexity theory [10], and with the fundamentals of data compression ([1], [6]).
2 NOTATIONS AND GENERALITIES.

Let $S_n$ be the symmetric group of $\{1, 2, \ldots, n\}$. Let $k$ be any field, such as $\mathbb{R}; \mathbb{C}$; or $GF(q)$, the finite field of $q$ elements.

$P = P_n = k[x]$ is the ring of polynomials in $x = (x_1, \ldots, x_n)$. $F = F_n$ is the set of all functions $f : k^n \rightarrow k$. There exists a canonical semantic mapping $\text{Sem} : P \rightarrow F$, which interprets polynomials as functions.

For $n$-tuples $y = (y_1, \ldots, y_n)$ and $\pi \in S^n$ let $y^\pi$ be $(y_{\pi(1)}, \ldots, y_{\pi(n)})$. Then define $SP = SP_n = \{\phi \in P|\forall \pi \in S_n \, \phi(x^\pi) = \phi(x)\}$; the set of symmetric polynomials over $k$, and $SF = SF_n = \{f \in F|\forall a \in k^n \forall \pi \in S_n \, f(a^\pi) = f(a)\}$; the set of symmetric functions over $k$.

The elementary symmetric polynomials are $\alpha_i = \sum_{j_1 < j_2 < \ldots < j_i} x_{j_1}x_{j_2} \ldots x_{j_i}, 1 \leq i \leq n$; let $\alpha = (\alpha_1, \ldots, \alpha_n)$. Put $\alpha_0(\underline{x}) \equiv 1$. It is well-known [13] that $\Psi : P \rightarrow SP$, defined by $(\Psi(\phi)) = \phi(\underline{x})$ is a $k$-algebra isomorphism ("Symmetric Polynomial Theorem"); the classic example of invariant theory [12]. One also has (see [3]):

- If $|k| = \infty$ (e.g., $\text{char}(k) = 0$), $\text{Sem}$ is injective, and $\phi \in P$ if and only if $\text{Sem}(\phi) \in SF$.
- If $k = GF(q)$, $\text{Sem}$ is surjective and $\text{Ker}(\text{Sem})$ is the ideal

$I < x_1^q - x_1, \ldots, x_n^q - x_n$.

Let us call a finite tuple $\underline{\phi} = (\phi_1, \ldots, \phi_t)$ from $SP$ a polynomial generating set for $SP$ if $\Psi : P_t \rightarrow SP$, defined by $(\Psi(\phi)) = \phi(\underline{\phi})$ is surjective, and a
polynomial basis if \( \Psi \) is bijective.

Examples of bases are \( \alpha \) (the elementary symmetric polynomials); and Newton’s polynomials \( \nu_i = \sum_{j=0}^{n} x_j^i; 1 \leq i \leq n. \) ([7], pg. 6)

Since by the symmetric polynomial theorem the elementary symmetric polynomials are algebraically independent, \( k[\alpha] \) has transcendence degree \( n \) over \( k. \)

Hence any polynomial generating set possesses at least \( n \) elements \([13]\). (1)

If one considers polynomials as functions, the situation is very different. The elementary symmetric functions are algebraically dependent. In fact, as we shall see in the following example, only a logarithmic number of them suffice to express all elements of \( SF. \)

**EXAMPLE 1.**

Let \( k = GF(2), n = 2^\lambda - 1 \). The *Hamming weight* \( w(a) \) of a vector \( a \) in \( k^n \) is defined as \( \{i | a_i \neq 0\} \).

If one writes \( w(a) \) in binary as \( \sum_{i=0}^{\lambda-1} f_i(a)2^i \), then \( f_i : a \rightarrow f_i(a) \) can be considered an element of \( SF. \) Since any symmetric function obviously depends on the Hamming weight only, and \( 0 \leq w(a) \leq n \), there are \( 2^{n+1} = 2^{2^\lambda} \) symmetric functions, and these can all be expressed in terms of the \( \lambda \) *bit functions* \( f_i. \) Also, there are only \( 2^{2^\lambda} \) functions \( g : k^\lambda \rightarrow k \) which proves that \( \Psi : F_\lambda \rightarrow SF, \) defined by \( (\Psi(g))(\bar{x}) = g(f) \) must in fact be injective.
More explicitly, it is not difficult to prove that $f_i(z) = \alpha_{2^i}(z)$; that $\alpha_k(z) = \prod_{k_i=1} a_{2^k}(z)$ (where $k = \sum_{i \geq 0} k_i 2^i$ in binary); and that every symmetric function is a $GF(2)$-linear combination $\sum_{k=0}^n \epsilon_k \alpha_k(z) = \sum_{k=0}^n \epsilon_k \prod_{k_i=1} a_{2^k}(z)$ with the $\epsilon_k$'s 0 or 1.

If $n = 2^k - 1$ the $2^k$ products $\prod_{k_i=1} a_{2^k}(z)$ (including 1, the empty product) are linearly independent as functions. In the same notation, algebraic dependencies $\alpha_k(z)\alpha_l(z) = \alpha_{k \lor l}(z)$ hold (where $(k \lor l)_i = k_i \lor l_i$ for all $i$, with "\lor" the Boolean "or" operator).

The Hamming weight occurs in a natural way in the theory of error correcting codes [8]. The "Hamming bits" $f_i$ are of great interest in the theory of Boolean complexity. Pippenger used them implicitly in his construction of small tree-type combinational circuits (formulas) for the symmetric Boolean functions [10].

Thus, it seems to make some sense to define "generating function sets" and "function bases" analogously to the polynomial case, and to pose questions like:

- Is there a lower bound like (1) for generating function sets?
- For which finite fields does there exist a function basis?

Since the notion of function does not depend on any algebraic structure, these problems are in fact combinatorial. The first will appear to be equivalent to a very nice diophantine equation, which we shall partially solve. The second
question can be answered fully: in the finite field case, a basis does not exist except for Example 1 (2.1).

3 GENERAL SETTING.

In this section, let \( a, q, n \in \mathbb{N} \); let \( X \) be an \( a \)-set, \( Y \) a \( q \)-set; and let \( SF_n \) be the collection of all symmetric functions \( f : X^n \to Y \), i.e., invariant under coordinate permutations by \( S_n \). Let \( F_Y \) be the set of all functions \( g : Y^t \to Y \).

**Definition 1** A generating function set for \( SF \) is a finite tuple \( f = (f_1, \ldots, f_t) \) from \( SF_n \) such that the mapping \( \Psi : F_Y \to SF \) defined by \( \Psi(g) = g(f) \) is surjective. It is called a function basis if \( \Psi \) is also injective. (Fig. 1).

![Figure 1: the mapping \( \Psi \)](image)

Thus, the elements of \( SF \) are "factorized" by \( \Psi \).

Let \( \rho = (\rho_1, \ldots, \rho_a) \) denote a vector of nonnegative integers with sum \( n \). By elementary combinatorics, the \( \rho \)'s number \( h(q, a, n) = \binom{n+a-1}{a-1} \). Suppose
that $x_1, \ldots, x_a$ are the elements of $X$. The orbits of $S_n$ on $X^n$ are then the ''Hamming spheres" $H_p = \{ v \in X^n | v_i = x_i \text{ exactly } p_i \text{ times; } 1 \leq i \leq a \}$. Thus, $X^n$ is partitioned into $h(q, a, n)$ disjoint Hamming spheres and $|SF| = q^{h(q, a, n)}$.

**Theorem 1 a. (Lower bound for generating function sets).**

If a generating function set exists, $t \geq H = \text{Ref } q \log \left( \frac{n+a-1}{a-1} \right)$.

b. A symmetric function basis exists if and only if equality holds in a.

**Proof.**
a. The substitution map $\Psi : F_Y \rightarrow SF$ is surjective, hence $|F_Y| = q^t$ certainly cannot be less than $|SF| = q^{h(q, a, n)}$.

b. If a basis exists, $\Psi : F_Y \rightarrow SF$ is bijective so equality holds. On the other hand, if equality holds, it suffices to construct a generating set. Let $\Phi$ be any bijection from the $h(q, a, n)$-set $H$ of $S_n$-orbits to $Y_t$. Let $\Phi_i (1 \leq i \leq t)$ be the $i^{th}$ projection of $\Phi$, i.e. $\forall h \in H : \Phi_i(h) = \Phi(h)_i$. We define $f_i : X^n \rightarrow Y$ by putting for all $a : f_i(a) = \Phi_i(h(a))$, where $a \in h(a)$.

The claim is that the $f_i$ form a generating set. Indeed, each $f_i$ is symmetric by definition. Furthermore, each symmetric function $F$ can be written as $F(a) = G \circ h(a)$, $h(a)$ as above and $G : H \rightarrow Y$. But $G \circ h(a) = G \circ \Phi^{-1} \circ \Phi \circ h(a) = G \circ \Phi^{-1}(\Phi_1(h(a)), \ldots, \Phi_t(h(a))) = g(f)$ where $g = G \circ \Phi^{-1}$ is a mapping : $Y^t \rightarrow Y$. So the $f_i$ form a generating set. (Note that this construction is a simple consequence of the elementary observation that a finite set of composite size can be written as a cartesian product.)

**REMARK.**

The representation of the symmetric functions by a basis is an efficient encoding.
of the elements of SF. In this way, this concept is related to the theory of Kolmogorow complexity [6] and that of data compression [1]. An idea from Schalkwijk [11] can be used to find an algorithmically nice example of a function $F$ as used in theorem 1. This $F$ can be seen as the natural generalization of the Hamming bit functions.

To do so, let $\Sigma = \{A, B\}$ be a two-element "source" alphabet. There is a bijection between the vectors $\rho$ describing the Hamming spheres, and the $h(q, a, n) \Sigma$-vectors (messages) containing $a - 1$ $A$’s: let $\rho$ correspond to a vector consisting of $p_i$ runs of $B$’s, $1 < i < a$, separated by $a - 1$ $A$’s.

In this way $\frac{1}{q}$ can be seen (at least asymptotically for $n + a - 1 \rightarrow \infty$) as the entropy of a memoryless channel with $q$ channel symbols and 2 source symbols $A, B$ of fixed frequencies $\frac{n}{n+a-1}$ and $\frac{a}{n+a-1}$ [1], [11].

An optimal and easily computable encoding of these messages (and, thus, of the $\rho$’s) can be constructed directly [11] as follows: order the messages lexicographically, and give each of them their $q$-ary index in this ordering. For this, $t$ digits are sufficient; these again form the basis.

Do there exist any nontrivial examples of bases? The remainder of this paper will be devoted to this question.
4 A DIOPHANTINE EQUATION AND EXPLICIT EXAMPLES.

The equality of Theorem 1 can be written as

\[ \binom{N}{K} = A^B \]

(2)

to be solved in natural numbers. From now on suppose w.l.o.g. that \( k \leq \left\lfloor \frac{N}{2} \right\rfloor \).

Similar equations or subcases have been studied in the literature ([4]: problem B31, D3; [15]: 15, 36, 1225, 19600). It seems difficult to obtain a full solution of (2), and we shall restrict ourselves to some special cases, one of which yields an infinite family of nontrivial solutions, and the other covering the finite field case.

Let us call a solution \((N, K, A, B)\) of (1) trivial whenever

\[ B = 0 : (N, 0, 1, 0), \]
\[ B = 1 : (N, K, \binom{N}{K}, 1), \text{ or} \]
\[ K = 1 : (A^B, 1, A, B). \]

It should be mentioned that these solutions need not be trivial in the sense of symmetric function theory. Indeed, Theorem 1 applies here, e.g. yielding the basis of Example 1 (2.1) (where \( K = 1 \)). First we shall show that this example is unique, if we consider finite fields.
4.1 THE FINITE FIELD CASE.

Consider the case that \( X = Y = GF(q) \), so in part a. of Theorem 1, \( a = q \) and one has to solve \( \binom{n+q-1}{q-1} = q^t \) with \( q \) a prime power. In Ex. 1 we already mentioned the solutions \( q = 2, N = 2^t \) (trivial in the sense of eq. (2); but corresponding to a nontrivial basis).

**Theorem 2** The Diophantine equation \( \binom{N}{q-1} = q^t \), with \( q \) a prime power and \( N \geq q - 1 \), has only the trivial solutions:

- \( N = q - 1, \ t = 0, \ q \) arbitrary;
- \( N = q, \ t = 1, \ q \) arbitrary; and
- \( N = 2^t, \ q = 2. \)

Hence, the only symmetric function theorems for finite fields occur for \( q = 2, \ n = 2^t - 1. \)

**Proof:** Let \( q = p^\lambda, \ p \) a prime; let \( N \geq q + 1. \) Put \( N = n + q - 1, \ n \geq 2. \) For real \( a \) define \( c_{p,n}^a = c_{a,n} = \left\lfloor \frac{a}{p} \right\rfloor + \left\lfloor \frac{a}{p^2} \right\rfloor + \left\lfloor \frac{a}{p^3} \right\rfloor + \ldots + \left\lfloor \frac{a}{p^n} \right\rfloor \) and \( c_0^a = c_a = c_{a,\infty}. \)

It is well-known ([5]) that \( p \) occurs exactly \( c_N \) times in \( N!; \) so \( p \) occurs exactly \( \rho \equiv c_{n-q-1} - c_n - c_{q-1} \) times in \( \binom{n+q-1}{q-1}. \) Note that, since \( \binom{n+q-1}{q-1} = q^t = p^{\lambda}, \) one also has \( \rho = t\lambda. \)

Easily, \( c_{q-1} = S - \lambda \) where \( S = \frac{p^{\lambda-1}}{p^{\lambda-1}}. \)

Using the fact that \( c_a = c_{a,t} + c_{a,\lambda} \) one has \( c_{n-q-1} - c_n = (c_{n-q-1,\lambda} - c_{n,\lambda}) + T \) with \( T = (c_{n+q-1} - c_{n,\lambda}). \)
Now \( \left\lfloor \frac{n+q-1}{p} \right\rfloor + p^{\lambda-i} \) if \( i \leq \lambda \), so \( (c_{n-q-1,\lambda} - c_{n,\lambda}) + T \) reduces to \( c_{n-1,\lambda} - c_{n,\lambda} + S + T \leq S + T \) (here \( S \) is as in (3)).

Therefore and by (3), \( \rho \leq \lambda + T \) which can be regrouped as \( \lambda + \left\lfloor \frac{n+q-1}{p^{\lambda+1}} \right\rfloor + \left\lfloor \frac{n+q-2}{p^{\lambda+1}} \right\rfloor + \left\lfloor \frac{n}{p^{\lambda+1}} \right\rfloor + \ldots \).

Since we need only consider terms with \( p^{\lambda+i} \leq n + q - 1 \), \( i \) is bounded by \( \mu = \mu \log \left( \frac{n-1}{p^\lambda} + 1 \right) \).

For \( i \geq 1 \) each term \( \left\lfloor \frac{n+q-1}{p^{\lambda+1}} \right\rfloor - \left\lfloor \frac{n}{p^{\lambda+1}} \right\rfloor \) is at most \( \left\lfloor \frac{n}{p^{\lambda+1}} + \frac{q-1}{p^{\lambda+1}} \right\rfloor - \left\lfloor \frac{n}{p^{\lambda+1}} - 1 \right\rfloor \leq \frac{1}{p^\lambda} + 1 \).

Hence, \( \rho \leq \lambda + \sum_{i=1}^{\mu} \left\lfloor \frac{1}{p^\lambda} + 1 \right\rfloor = \lambda + \mu + \frac{p^{n-1}}{p^{\mu}(p-1)} < \lambda + \mu + 1 \); so \( \rho \leq \lambda + \mu \).

We also had \( \rho = t\lambda \) so \( q' = p^\rho \leq p^{\lambda+\mu} \leq n + q - 1 \) (by \( \mu \)'s definition); i.e. \( \binom{n+q-1}{q-1} \leq n + q - 1 \) or, equivalently, \( \frac{\binom{n+q-2}{q-2}}{q-1} \leq 1 \).

However, \( \frac{\binom{n+q-2}{q-2}}{q-1} \) is strictly increasing as a function of \( q \) (for fixed \( n \geq 2 \)) with value 1 for \( q = 2 \). So this is the only possibility. Then \( \binom{n+q-1}{q-1} = n + 1 = 2^t \) and we are done.

Finally we investigate some nontrivial cases.

### 4.2 OTHER SOLUTIONS.

**K =2, B even.**

In (2), let \( K = 2 \) and \( B \) even. W.l.o.g, let \( B = 2 \). Then (2) is in fact an ancient question: which numbers are simultaneously triangular and square?
This easily reduces to the Pellian equation \( 1 + 8y^2 = x^2 \) (see [15], 15; and [5]). The infinite set of solutions is given in the standard way in terms of the fundamental unit \( 3 + \sqrt{8} \) of \( \mathbb{Q}(\sqrt{8}) \) as \( x_n + y_n\sqrt{8} = (3 + \sqrt{8})^n \), yielding the parametrization \( (N, K, A, B) = (\frac{x_{i+1}}{2}, 2, y_i, 2), \ i \geq 2 \), with \( x_1 = 3, \ y_1 = 1, \ x_{n+1} = 3x_n + 8y_n, \ y_{n+1} = x_n + 3y_n \ (n \geq 2) \).

The smallest solution is \( (N, K, A, B) = (9, 2, 6, 2) \), corresponding to \( |X| = 3, |Y| = 6, n = 7, t = 2 \) in Thm. 1. This means that all symmetric functions in 7 variables on 3 symbols, taking values in a 6-set, can be uniquely written in terms of only 2 "elementary symmetric" basis functions. The next solution is \( |X| = 3, |Y| = 35, n = 48, t = 2 \); etc.

**K=2.** B odd.

This is considerably more difficult. The cases \( N = 2t \) and \( N = 2t + 1 \) taken together yield the equation \( y^B - 2x^B = \pm 1 \) (where \( t = x^B, xy = A \)). Deep results of various authors are cited in Mordell’s book ([9], Ch. 23 Thm 5, Ch. 28 Thms. 11, 13, 17, 20). For \( B = 3 \) and 5 there are no nontrivial solutions. For larger \( B \) there is at most one.

**K=3**

In this case a computer program (below) only found \( (50, 3, 140, 2) \). As it is reported in Wells [15], 19600 is the only square tetrahedral number, which implies that this is the only solution having \( K = 3, B \) even.

**OTHER**

For fixed \( K \) and \( B \) larger than 2, an elementary analysis in the same vein as
that for $K = 2$ leads to higher degree equations with very few solutions, as studied by Siegel, Baker and others [9]. It is easy to find ranges for $K$ and $N$ where only trivial solutions exist. For example, if $P$ is a prime number between $N$ and $N - K$ then in $\binom{N}{K}$ a single factor $P$ occurs; hence the only possible solutions are trivial ($K = 1$). Elaborating on this idea, many pairs $(N, K)$ can be eliminated by the following observation. Let $c_0^P$ be as in the proof of Thm. 2. Then one obviously has:

**Lemma 1** Let $a = \binom{N}{K}$. If a nontrivial solution of Eq. (1): $\binom{N}{K} = A^B$ exists, then $B | \gcd(c_N^P - c_K^P - c_{N-K}^P)$ where $P$ ranges over all prime numbers $\leq N$.

Using a simple Maple implementation of this lemma, all values $N \leq 100$, $K \leq 50$ were investigated; and by a faster Pascal program all values up to $N = 1000$, $K \leq 500$ (the code is given in the Appendix). We found that the only nontrivial solutions in this range were those already mentioned:

**RESULT**

The only nontrivial solutions of $\binom{N}{K} = A^B$ below $N = 1000$, $K = 500$ are $(N, K, A, B) = (9, 2, 6, 2), (50, 2, 35, 2), (289, 2, 204, 2)$ and $(50, 3, 140, 2)$.

One is tempted to formulate the (facile) conjecture that solutions with $B \geq 3$ do not exist.
5 CONCLUSIONS.

We investigated the combinatorial version of the Symmetric Function Theorem. This led to a lower bound for generating sets for the symmetric functions. Equality in this bound is equivalent to the interesting Diophantine equation \( \binom{N}{K} = A^B \), possessing an infinite set of nontrivial solutions, the smallest being \((N, K, A, B) = (9, 2, 6, 2)\). The corresponding symmetric function theorem expresses the symmetric functions in 7 variables on 3 symbols, taking values in a 6-set, uniquely in terms of 2 basis functions.

Finally, the only finite field with a symmetric function theorem (not to be confused with the symmetric polynomial theorem) is \( GF(2) \), in dimension \( 2^\lambda - 1 \).

6 REFERENCES.


7 APPENDIX

7.1 A simple Maple program

This may be used to investigate the binomial coefficients \( \binom{m}{k} \) for \( m \) up to a given number \( n \).

- the following procedure generates a list of the first \( n \) primes:

\[
\text{priemen} :=
\]
\[
\text{proc}(n)
\]
\[
\text{local } i, p, \text{priemlijst};
\]
\[
\text{priemlijst} := []; \text{for } i \text{ to } n \text{ do } p := \text{numtheory} \[ \text{ithprime} \](i); \text{if } p > n \text{ then break fi; }
\]
\[
\text{priemlijst} := [\text{op} \( \text{priemlijst} \), p] \text{ od;}
\]
\[
\text{priemlijst}
\]
\[
\text{end;}
\]

- the following procedure computes the exponent to which the prime \( p \) divides \( m! \), and stores these for \( m \) from 1 to \( n \) in a table "cpm":

\[
\text{prifac} :=
\]
\[
\text{proc}(n, p)
\]
\[
\text{local } i, m, h, \text{cpm}, \text{lijst};
\]
\[
\text{cpm} := \text{table}();
\]
\[
\text{for } m \text{ to } n \text{ do } \text{cpm}[m] := 0; \text{for } i \text{ to } m \text{ do } h := \text{floor}(m/p^i);
\]
\[
\text{if } h = 0 \text{ then break fi; } \text{cpm}[m] := \text{cpm}[m] + h \text{ od od; }
\]
\[
\text{lijst} := []; \text{for } m \text{ to } n \text{ do } \text{lijst} := [\text{op} \( \text{lijst} \), \text{cpm}[m]] \text{ od;}
\]
\[
\text{lijst}
\]
end;

- given \( k \) and \( n \), the following procedure computes, for \( m \) from \( 2 \cdot k \) to \( n \) and for each prime \( p \) the exponent to which the prime \( p \) divides \( \binom{n}{k} \). It uses the above procedure "prifac". For each \( m \), the \( \gcd \) of these exponents is stored in the array "ggdlijst", and the numbers \( m \) for which the stored \( ggd \) is greater than 1 are put into a list "candidaten":

```
zeefmet:=
proc(k)
local i,p,m,cmp,candidaten,ggdlijst,hulp;
    ggdlijst:=array(2*k..n);candidaten:=[ ];
cmp:=prifac(n,2); for m from 2*k to n do ggdlijst[m]:=cmp[m]-cmp[k]-cmp[m-k] ;
    for i from 2 to nrpr do if ggdlijst[m]=1 then break fi; p:= pr[i];
    cmp:=prifac(n,p);ggdlijst[m]:= igcd(ggdlijst[m], cmp[m]-cmp[k]-cmp[m-k]) od od;
    for m from 2*k to n do if ggdlijst[m]>1 then
        candidaten:=[op(candidaten),[m,k,ggdlijst[m]]] fi od;
candidaten
end;
```

- In the main program, "zeefmet(k)" is executed for given \( n \) (here, \( n = 100 \)) and all \( k \) with \( 2 \cdot k \leq n \)

```
n:=100; pr:=priemen(n); nrpr:=nops(pr);
interface(quiet=true);
for k from 2 to floor(1/2*n) do print(zeefmet(k)) od;
```
7.2 A faster Turbo Pascal program.

A faster Turbo Pascal program. - This program works analogously to the above Maple attempt. The constant bomax equals the "n" of our Diophantine equation; pmax is at least the number of primes \( \leq \) bomax. -

program zeprobeter;

const bomax = 7600; pmax = 965;
var nrprs: integer; p: array[1..pmax] of integer;

const bomax = 7600; pmax = 965;
var nrprs: integer; p: array[1..pmax] of integer;

const bomax = 7600; pmax = 965;
var nrprs: integer; p: array[1..pmax] of integer;

noverkoutbeter: text;

- A standard sieve to store the primes \( \leq \) bomax in an array p[1..pmax] -

procedure primes;

var i,incr:integer;z:array[1..bomax] of boolean;

begin
  for i:=1 to bomax do z[i]:=true;
  z[1]:=false;
  incr:=0;
  repeat
    i:=incr+1;
    while not z[i] do i:=i+1;
    incr:=i;
    i:=i+incr;
end;
while i<=bomax do
    begin
        z[i]:=false;
        i:=i+incr
    end
until incr>=bomax;

for i:=1 to pmax do p[i]:=-1;
i:=1;
incr:=0;
repeat
    i:=i+1;
    if z[i] then
        begin
            incr:=incr+1;
            p[incr]:=i
        end
until (i>=bomax) or (incr>=pmax);
nrprs:=incr
end;

- A standard gcd procedure -

function ggd(a,b:integer):integer;
var ao,bo,r,ro,v:integer;


label 1;
begin
  ao:=a;
  bo:=b;
  r:=bo;
  if bo=0 then begin ro:=ao; goto 1 end else
  if ao=0 then begin ro:=bo; goto 1 end;
  while not(bo=0) do
    begin
      v:=ao div bo; ro:=r; r:=ao mod bo; ao:=bo; bo:=r
    end;
1:  ggd:=ro
end;

- the following procedure fills, for \( n \) from 1 to \( \text{bomax} \), the array entry \( \text{cnp}[n] \) with the exponent to which the \( p \)-th prime (i.e., \( p[pindex] \)) occurs in \( n! \):

procedure vulcnp(pindex: integer);
var n,j,prime: integer; pexpj: longint;
begin
  prime:=p[pindex]; write(' ',prime);
  for n:=1 to \( \text{bomax} \) do
    begin
      cnp[n]:=0; j:=0; pexpj:=1;
      while pexpj <= n do

begin
  pexpj := pexpj * prime;
  cnp[n] := cnp[n] + n div pexpj
end
end
end;

- The following procedure fills the array entry "expo[n]" with the \(gcd\) of the exponents of the primepowers occurring in \(\binom{n}{k}\). This is done for \(2 \cdot k \leq n \leq \text{bomax}\). If the \(gcd\) is already one, the entry is ignored in further calculations.

procedure maakexp(k: integer);
var a, b, n, pindex: integer; alleseen: boolean;
label 1, 2;
begin
  for n := 2 * k to \text{bomax} do expo[n] := 0;
  alleseen := false; pindex := nrprs;
  while not(alleseen) and (pindex > 0) do
  begin
    vcnp(pindex); n := \text{bomax} + 1;
    1: n := n - 1;
    if n < 2 * k then goto 2; if (expo[n] = 1) then goto 1;
    a := expo[n]; b := cnp[n] - cnp[k] - cnp[n - k];
    expo[n] := ggd(a, b); goto 1;
    2: pindex := pindex - 1
  end
end;
procedure check(a,b:integer);
var k,n:integer;bo:boolean;
begin
    primes; rewrite(noverkoutbeter,'noverkoutbeter');
    for k:=a to b do
begin
        writeln(noverkoutbeter);
        writeln;writeln;writeln('Bezig aan k = ' ,k,' tot n maximaal ',bomax);
        writeln(noverkoutbeter,'Geval k = ' ,k,' tot n maximaal ',bomax);
        maakexp(k);bo:=false;
        for n:=2*k to bomax do
            if expo[n]>1 then
                begin
                    bo:=true;
                    writeln(noverkoutbeter,'Opl.: n = ',n,', k = ',k,', macht: ',expo[n])
                end;
        if bo = false then
        begin

- The following procedure applies the procedure "check" for $k$ from one to $k_{max}$ and finds solutions of our Diophantine equation in the range $n \leq b_{max}$, $k \leq k_{max}$ -

```
procedure totalchecktotkis(kmax: integer);
var grens: integer;
begin
  if 2*kmax <= b_{max} then grens := k_{max}
  else grens := b_{max} div 2;
  check(2, grens)
end;
```

- This is (a form of) the main program: -

```
begin
  totalchecktotkis(2)
end.
```