Acta Cybernetica **22** (2015) 423–433.

On a Parity Based Group Testing Algorithm*

Sándor Z. Kiss†, Éva Hosszu‡, Lajos Rónyai§, and János Tapolcai‡

This paper is dedicated to the memory of Professor Ferenc Gécseg

Abstract

In traditional Combinatorial Group Testing the problem is to identify up to d defective items from a set of n items on the basis of group tests. In this paper we describe a variant of the group testing problem above, which we call *parity group testing*. The problem is to identify up to d defective items from a set of n items as in the classical group test problem. The main difference is that we check the parity of the defective items in a subset. The test can be applied to an arbitrary subset of the n items with two possible outcomes. The test is positive if the number of defective items in the subset is odd, otherwise it is negative. In this paper we extend Hirschberg et al.'s method to the parity group testing scenario.

Keywords: combinatorial group testing

1 Introduction

1.1 Motivation

Dealing with errors during transmission has been a long-standing problem of communication theory. Numerous error scenarios have been considered, mostly focusing on cases when the channel is unreliable. In [8] Hachem et al. proposed a novel possibility: what if the encoder itself is introducing uncertainty?

There are several causes as to why an encoder might behave in a faulty manner [8]. First, the physical device implementing the encoder might be faulty, causing the encoder to have faults itself. Second, due to ever-reducing chip size, soft errors

*The work of János Tapolcai was partially supported by the Hungarian Scientific Research Fund (grant No. OTKA 108947). The work of Lajos Rónyai was supported by the Hungarian Scientific Research Fund (grant No. OTKA NK 105645).

†Budapest University of Technology and Economics, Department of Algebra E-mail: kisspest@cs.elte.hu

‡MTA-BME Future Internet Research Group, Budapest University of Technology and Economics (BME) E-mail: {hosszu,tapolcai}@tmit.bme.hu

§Computer and Automation Research Institute Hungarian Academy of Sciences and Budapest University of Technology and Economics, Department of Algebra, E-mail: ronyai@sztaki.hu

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

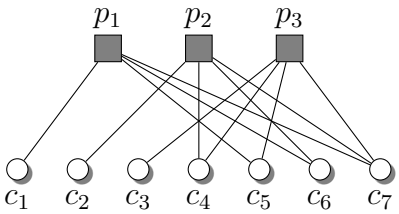

Figure 1: The generator matrix and the Tanner graph of the $(7, 3)$ dual Hamming code.

in processing and storage are becoming more and more frequent [13]. Third, with the scaling of technology, device degradation and variability in transistor design may also cause unreliable behaviour [3]. Lastly, errors might happen during distributed encoding when physically separated devices are connected through a noisy channel, as in sensor networks [2].

In this work we adapt their fault model. Let us consider the Tanner-type factor graph defined as below. For a given $k \times n$ linear code G and $(n-k) \times n$ parity check matrix H the *Tanner graph* is the following. The Tanner graph $T = (\{V_1, V_2\}, E)$ consists of node set $V_1 \dot\cup V_2$, where $|V_1| = k$ and $|V_2| = n$; and for $i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}$ edge set $E = \{\{v_1^i, v_2^j\} : v_1^i \in V_1, v_2^j \in V_2, G[i, j] = 1\}$.

Note that as the Tanner graph is defined by the generator matrix, it is not necessarily unique to the code.

Let us model the faults as edges getting erased in the factor graph of G, which reveal themselves as bits getting flipped $1 \to 0$. It is assumed that due to the edge erasures every bit that is 1 may get flipped to 0 independently from each other with probability p.

Assuming that the original generator matrix is known to both the receiver and the transmitter, an easy way to check against erasures would be to send the unit vectors of length k as test messages. In this case when sending the i^{th} unit vector the receiver would receive the i^{th} row of the generator thus enabling to detect any number of faults after getting all the messages – as many as the number of rows in the generator. The natural question follows: can one do better?

In this work we investigate what one can do to check whether the encoder itself is introducing uncertainty. Hachem et al. [8] considered the problem of introducing enough redundancy so as to counteract the effects of a faulty encoder. The problem we address in this paper is how one would go about discovering the locations of these erasures.

1.2 Introducing Parity Group Testing

The traditional problem in group testing is the following. Let S be a set of items with n elements, some of them (say, at most d) are possibly defective. For simpler notation we assume that $S = \{1, 2, \ldots, n\}$. We intend to find the defective items via *group tests*. A group test is a subset T of S; testing T has two possible outcomes.

It is positive if there is at least one defective item in T and negative otherwise. The tests may be executed either in an adaptive manner, taking the preceding tests' outcome into account when designing the next one, or non-adaptively, when all tests are to be determined at the start. In this paper we consider the non-adaptive version of the problem.

The main objective of any combinatorial group testing (CGT) scheme is to find the defective elements via such group tests efficiently. Efficiency may be measured in different ways, a prevalent goal is to try and minimize the number of subsets T to be tested. There is rich literature on the subject, for further details we refer the reader to [4, 10, 11].

Translating this concept to binary linear encoders goes as follows. The set of items are all the bits that could get erased, the 1s in the generator matrix. A test would be a message, which gets evaluated based on whether it differs from what we were supposed to receive or not – assuming that the generator matrix of the code is known to both the receiver and the transmitter. The items included in a test are the ones from every row where there is a 1 in the test message, so individual testing of the items would be to send messages that contain only a single 1 in them, i.e. the unit vectors. Testing a pool of potential erasures is to send a message that contains more than just one bit that is 1.

Let us present an illustrative example. Let G be the generator matrix for the (7,4)–Hamming code, known to both the transmitter and the receiver. Suppose the erasures denoted by bold **0**'s on Figure 2 happen. Sending the unit vectors of length 4 would display the current state of G row-by-row on the receiver side, making it possible to diagnose any number of faults using 4 messages.

However, the erasures cancel each other out if we send a message containing more than just one bit that is 1 and they hit an even number of erasures. For example let us send the message $(1, 1, 0, 0)$ using G' depicted on Figure 2b. The received word would be $(0, 0, 0, 1, 0, 1, 1)$ whereas the correct word we should receive with an erasure-free received word is $(1, 0, 0, 0, 0, 1, 1)$. This reveals that there are erasures in the first and fourth column but the two erasures in the second column don't show up.

Motivated by this observation, we define *parity group testing* as follows. In the parity group testing problem the aim again is to find at most d defectives in an n-element set S. However, the two outcomes of a test $T \subseteq S$ are changed: instead of revealing the presence of defectives in T the result of a test will now show whether there is an *odd* or *even* number of defective items in T, hence the name parity testing. Our aim is for given set size n and maximum number of defectives d identify all the defective items such that the number of necessary parity group tests is small.

$$\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \quad \begin{pmatrix}1&1&0&1&0&0&1\\0&1&0&1&0&1&0\\1&0&0&1&1&0&0\\1&1&1&0&0&0&0\\1&0&0&0&0&1&1\end{pmatrix}$$

(a) The generator matrix G sending the message $(1, 1, 0, 0)$.

$$\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \quad \begin{pmatrix}\mathbf{0}&\mathbf{0}&0&\mathbf{0}&0&0&1\\0&\mathbf{0}&0&1&0&1&0\\\mathbf{0}&0&0&1&1&0&0\\1&1&1&0&0&0&0\\0&0&0&1&0&1&1\end{pmatrix}$$

(b) The erasure-stricken matrix G' sending the message $(1, 1, 0, 0)$.

$$\begin{array}{rc} & \begin{pmatrix}1&0&0&0&0&1&1\end{pmatrix}\\ \text{XOR} & \begin{pmatrix}0&0&0&1&0&1&1\end{pmatrix}\\ \hline & \begin{pmatrix}1&0&0&1&0&0&0\end{pmatrix}\end{array}$$

(c) The resulting group test reveals an error in the first column by taking the XOR of the received words.

Figure 2: An example of group tests translated to linear encoders.

2 A Chinese Remainder Theorem based CGT Algorithm

In this section first we recap a previous CGT algorithm our parity group testing constructions are based on, then we describe our algorithms for identifying faulty items in the parity setting. We assume the underlying set S to be $\{1, \ldots, n\}$ and that there are at most d faulty items (unless stated otherwise).

Eppstein, Goodrich and Hirschberg [5] provided a non-adaptive combinatorial group testing algorithm based on the Chinese Remainder Theorem. First a sequence of pairwise coprime positive integers $\{p_1, p_2, \ldots, p_k\}$ is selected such that

$$n^d \leq P = \prod_{i=1}^{k} p_i.$$

In this setting the the total number of tests would be

$$t(n, d) = \sum_{i=1}^{k} p_i.$$

We may assume that $p_1 < p_2 < \cdots < p_k$. The first group test X contains the numbers a where $a \equiv 0 \pmod{p_1}$ holds, while the second contains the numbers b satisfying $b \equiv 1 \pmod{p_1}$, and so on, till all remainders for each p_i are taken for $i = 1, \ldots, k$.

2.1 Constructive Algorithm to Find the Solution for Single Defective items

Note that if there is at most one defective item, then parity group testing is the same as the classical group testing problem, i.e., if the set X contains odd number of defective items, then it follows that the only defective item is in X, otherwise X does not contain the defective item.

Let a_i denote the remainder of a single item $x \in S$ for p_i. The task is to find the number x which satisfies the following system of congruences:

$$x \equiv a_i \pmod{p_i} \tag{1}$$

for $i = 0, \ldots, k$.

For each i the integers p_i and $\prod_{j \neq i} p_j$ are relatively prime. Using the extended Euclidean algorithm we can find integers r_i and q_i such that

$$r_i p_i + q_i \prod_{j \neq i} p_j = 1.$$

Then, choosing $e_i = q_i \prod_{j \neq i} p_j$, x can be reconstructed by

$$x = \sum_{i=1}^{k} a_i e_i \pmod{\prod_j p_j} \tag{2}$$

which satisfies (1). This well known scheme of reconstruction from Chinese Remainders can be summarized as follows.

Algorithm 1 Chinese Remainder

Input: $(p_1, \ldots, p_k)$, $(a_1, \ldots, a_k)$

for $i = 1$ to k **do**

Compute

$$N_i = \prod_{j \neq i} p_j,$$

$$q_i = N_i^{-1} \pmod{p_i}.$$

end for

Compute

$$x = \sum_{i=1}^{k} a_i q_i N_i \pmod{p_1 p_2 \cdots p_k}.$$

2.2 Constructive Algorithm to Find the Solution for d defective items in the parity setting

Let $x_1, \ldots, x_d$ denote the defective items, where $d > 1$.

The following simple fact shows that the defective items can be well separated in the different residue classes.

Claim 1. *Let $e_1, \ldots, e_v$ be pairwise coprime positive integers. If $v \geq \binom{d}{2} \log_2 n$, then there exists an e_i, where $1 \leq i \leq v$ such that $x_1, \ldots, x_d$ lie in different residue classes modulo e_i.*

Proof. We prove the statement by contradiction. Assume that $1 \leq x_1 < \ldots < x_d \leq n$, and for any $1 \leq i \leq v$ there are at least two elements among $x_1, \ldots, x_d$ such that they are in the same residue classes modulo e_i. In other words for all $1 \leq i \leq v$, there exist $1 \leq l < m \leq d$ such that $e_i | x_m - x_l$. There may be at most $\binom{d}{2}$ pairs of the last type, hence by the pigeonhole principle there exist $1 \leq r < s \leq d$ such that for at least $c \geq \log_2 n$ different indices j we have $e_j | x_s - x_r$. As e_i's are pairwise coprime, it follows that $\prod e_j | (x_s - x_r)$, but $n \leq 2^c \leq \prod e_j | (x_s - x_r) < n$ which is a contradiction. (Here the product is over the indices j such that $e_j | x_s - x_r$.) □

If we set $k \geq \binom{d}{2} \log_2 n + d \log_2 n + 1$, it follows from the above Claim that there exists pairwise coprime numbers $p_1, \ldots, p_t$ among the numbers $p_1, \ldots, p_k$ such that $p_1 \cdots p_t \geq n^d$ and $x_1, \ldots, x_d$ lie in different residue classes modulo p_i, where $1 \leq i \leq t$. This means that parity testing with the integers $p_1, \ldots, p_t$ the positive outcome (i.e., when the parity of the defective items is odd in a residue class modulo p_i) implies that there is exactly one defective item in the corresponding residue class. Please note that such a collection $p_1, \ldots, p_t$ can be efficiently selected from $p_1, \ldots, p_k$.

Let $y_i^{(1)}, \ldots, y_i^{(d)}$ denote the remainders of the d defective items $x_1, \ldots, x_d \in S$ modulo p_i. Recall that we selected the moduli p_i in such a way that

$$n^d \leq P = \prod_{i=1}^{t} p_i.$$

The task is to find the numbers $x_1, \ldots, x_d$ which satisfy the following system of congruences:

$$x_1 \equiv y_i^{(1)} \pmod{p_i}, \quad \ldots, \quad x_d \equiv y_i^{(d)} \pmod{p_i}$$

for all $1 \leq i \leq t$.

Please note that for an i the residues $y_i^{(j)}$ are pairwise different for $j = 1, \ldots, d$. Having the numbers $y_i^{(j)}$ at hand, we can calculate the residues of the elementary symmetric polynomials[1] of $x_1, \ldots, x_d$ modulo all the p_i by using Algorithm 3:

$$\sigma_1(x_1, \ldots, x_d) \equiv a_1^{(1)} \pmod{p_1}, \ldots, \sigma_1(x_1, \ldots, x_d) \equiv a_t^{(1)} \pmod{p_t};$$

$$\vdots$$

$$\sigma_d(x_1, \ldots, x_d) \equiv a_1^{(d)} \pmod{p_1}, \ldots, \sigma_d(x_1, \ldots, x_d) \equiv a_t^{(d)} \pmod{p_t};$$

By using the Chinese remainder theorem we can calculate

$$\sigma_1(x_1, \ldots, x_d) \equiv A_1 \pmod{P}, \ldots, \sigma_d(x_1, \ldots, x_d) \equiv A_d \pmod{P}.$$

[1] For details, see the Appendix.

As $P \geq n^d$ and

$$0 < \sigma_1(x_1, \ldots, x_d), \ldots, \sigma_d(x_1, \ldots, x_d) < n^d$$

the following equalities hold.

$$\sigma_1(x_1, \ldots, x_d) = A_1, \ldots, \sigma_d(x_1, \ldots, x_d) = A_d \ .$$

It is easy to see that the roots of the polynomial

$$f(w) = w^d - \sigma_1 w^{d-1} + \sigma_2 w^{d-2} - .... + (-1)^d \sigma_d$$

are $x_1, \ldots, x_d$. We can find the roots of f by using the root finder method [9]. The essence of this method is to isolate the roots by using the Sturm theorem and we can find the roots applying the bisection method (binary search). More formally we have the following algorithm.

Algorithm 2 Parity based Chinese Remainder Sieve algorithm

Input: $y_i^{(1)}, \ldots, y_i^{(d)}$ for all $1 \leq i \leq t$, $p_1, \ldots, p_t$

1: **for** $j = 1$ to d **do**
2: **for** $i = 1$ to t **do**
3: $\sigma_j(y_i^{(1)}, \ldots, y_i^{(d)}) = a_i^{(j)} \pmod{p_i}$
4: **end for**
5: $A_j = \text{ChineseRemainder}(a_1^{(j)}, \ldots, a_t^{(j)}, p_1, \ldots, p_t)$
6: **end for**
7: Set $f(z) = z^d + \sum_{l=1}^{d} (-1)^l A_l z^{d-l}$
8: Compute $(x_1, x_2, \ldots, x_d) = \text{Root Finder } (f(z))$

3 Analysis

In this section we will give a brief analysis of the running time of our algorithm and an upper bound for the number of test required to identify the defective items as well. Throughout the remaining part of this section $\log n$ denotes the natural logarithm i.e., the logarithm to the base e.

3.1 Number of tests

Let $t(n, d)$ denote the number of tests constructed in the Chinese Remainder Sieve discovered by Hirschberg et al. They proved that the d defective items could be identified using the number of tests

$$t(n, d) < \frac{\lceil 2d \log n \rceil^2}{2 \log \lceil 2d \log n \rceil} \Big(1 + \frac{1.2762}{\log \lceil 2d \log n \rceil} \Big).$$

As noted in the introduction, in our case the number of required tests is

$$t(n, d) = \sum_{i=1}^{k} p_i.$$

To simplify the calculations we can assume that the p_i's are primes. Let q_i denote the ith largest prime. It follows that we have to estimate

$$\sum_{i=1}^{k} q_i.$$

It is well known [7] that $q_k = O(k \log k)$ which implies that

$$\sum_{i=1}^{k} q_i = O(k^2 \log k).$$

In our case we can choose $k = \binom{d}{2} \log_2 n + d \log_2 n + 1 = \frac{d(d+1)}{2} \log_2 n + 1$, thus we have the following upper bound to the number of tests in the parity case:

$$t(n,d) = O\Big(d^4 \log^2 n \cdot \log d + d^4 \log^2 n \cdot \log\log n\Big).$$

3.2 Running time

Claim 2. *The Parity based Chinese Remainder Sieve algorithm finds the defective items by using $O(d^{10} \log^3 n)$ bit operations. This is in addition to the cost of the tests.*

Proof. The Parity based Chinese Remainder Sieve algorithm contains four steps. In the first step it determined the residues y_i^j. They are essentially the outcomes of the tests. In the second step, it computes the elementary symmetric polynomials, in the third step it uses the Chinese remainder theorem, and finally it determines the roots of the corresponding polynomial.

In Algorithm 3 we compute the symmetric polynomials recursively. In the rth step there are $r-1$ additions and $r-1$ multiplications, thus we can compute all symmetric polynomials by using $1 + \ldots + (d-1)$ additions and multiplications. As $1 \le x_1, \ldots, x_d \le n$, one addition needs $O(\log n)$ bitoperations, and one multiplication requires $O(\log^2 n)$ bit operations, thus the total cost of Algorithm 3. is $O(d^2 \log^2 n)$ bit operations.

In this paragraph we analyze the Chinese remaindering process (Algorithm 1.) It is well known [1] that Chinese remaindering requires $O(\log^2 P)$ bitoperations. It is easy to see [16] that

$$\log P \le \sum_{i=1}^{k} \log q_i \le \pi(q_k) \log q_k = k \log q_k,$$

where $\pi(x)$ denotes the number of primes up to x. It is well known [7] that the kth prime number is $O(k \log k)$, thus we have

$$\log P = O(k(\log k + \log\log k)) = O(k \log k).$$

We know that

$$k = O(d^2 \log n),$$

which implies $k \log k = O(d^2 \log n(\log d + \log\log n))$. It follows that the total cost is $O(d^4 \log^2 n \cdot (\log^2 d + (\log\log n)^2))$. Since the number of systems of congruences is d, computing the A_j's in the Chinese Remainder Filter needs $O(d^5 \log^2 n(\log^2 d + (\log\log n)^2))$ bit operations.

In the last step we have to determine the roots of the polynomial $f(z)$. For a polynomial $f(z) = a_d z^d + \ldots + a_1 z + a_0$ let

$$K = \sum_{i=0}^{d} |a_i|.$$

It is clear that all coefficients of our polynomial are at most n^d, which implies that $K < dn^d$. It follows from [6] that the running time of Heindel's algorithm is $O(d^{10} + d^7 \log^3 K)$. We have to use the bisection method at most $d-1$ times, which requires $O(d \log n)$ operations, because the length of each interval is at most n. Thus the total cost to determine all roots requires at most $O(d^{10} + d^{10} \log^3 n + d \log n) = O(d^{10} \log^3 n)$ bitoperations. This implies that the total cost of the Chinese Remainder Filter Algorithm is $O(d^2 \log^2 n + d^5 \log^2 n(\log^2 d + (\log\log n)^2) + d^{10} \log^3 n) = O(d^{10} \log^3 n)$ bit operations. □

Please note that there is a more sophisticated algorithm than Heindel's method, it can be found in [15]. The running time of this algorithm is better than Heindel's algorithm.

4 Conclusions

Motivated by the problem of error location in a linear encoder in this paper we introduced a novel variant of a classic combinatorial search task called *parity group testing*. After presenting the basic framework we showed how to adapt the Chinese Remainder Theorem based search algorithm to our scenario such that d defectives can be found in a set of n elements using $O\Big(d^4 \log^2 n \cdot \log d + d^4 \log^2 n \cdot \log\log n\Big)$ parity group tests, using $O(d^{10} \log^3 n)$ bit operations.

References

[1] P. C. van Oorschot A. J. Menezes and S. A. Vanstone. *Handbook of Applied Cryptography*, volume 4. CRC Press, 1996.

[2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[3] Aris Christou. *Electromigration and Electronic Device Degradation.* Wiley-Interscience, 1994.

[4] Ding Zhu Du and Frank Hwang. *Combinatorial group testing and its applications*. World Scientific, 1993.

[5] David Eppstein, Michael T Goodrich, and Daniel S Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM Journal on Computing*, 36(5):1360–1375, 2007.

[6] R. Loos G. E. Collins. Polynomial real root isolation by differentation. In *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation*, pages 15–20. ACM, 1976.

[7] E. Kowalski H. Iwaniec. *Analytic Number Theory*, volume 53. American Mathematical Society, 2004.

[8] Jad Hachem, I-Hsiang Wang, Christina Fragouli, and Suhas Diggavi. Coding with encoding uncertainty. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 276–280. IEEE, 2013.

[9] Lee E. Heindel. Integer arithmetic algorithms for polynomial real zero determination. *J. ACM*, 18(4):533–548, October 1971.

[10] FK Hwang. A method for detecting all defective members in a population by group testing. *Journal of the American Statistical Association*, 67(339):605–608, 1972.

[11] FK Hwang and VT Sós. Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hungar*, 22:257–263, 1987.

[12] Hao Jiang, Stef Graillat, and Roberto Barrio. Accurate and fast evaluation of elementary symmetric functions. In *IEEE Symposium on Computer Arithmetic*, pages 183–190, 2013.

[13] Michael Nicolaidis. *Circuit-Level Soft-Error Mitigation*. Springer, 2011.

[14] Viktor V. Prasolov. *Polynomials*. Springer, 2004.

[15] Michael Sagraloff and Kurt Mehlhorn. Computing real roots of real polynomials. *Journal of Symbolic Computation*, 2015.

[16] G. Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 46. Cambridge University Press, 1995.

Appendix

We need the following facts about polynomials [14]. For $m \geq 0$, let

$$\sigma_m = \sigma_m(t_1, \ldots, t_d) = \sum_{1 \leq j_1 < j_2 < \ldots < j_m \leq d} t_{j_1} \cdot \ldots \cdot t_{j_m}$$

be the m^{th} elementary symmetric polynomial of $t_1, \ldots, t_d$.

We can compute the elementary symmetric polynomials by using the following algorithm [12].

Algorithm 3 Elementary Symmetric Polynomial Calculator

Input: $X = (x_1, \ldots, x_d)$ and m
Output: all the elementary symmetric polynomials $\sigma_1, \ldots, \sigma_d$

1: function $\sigma_m^{(d)} = \texttt{SumESF}(X, m)$
2: $\sigma_0^{(i)} = 1, 1 \leq i \leq d-1; \sigma_j^{(i)} = 0, j > i; \sigma_1^{(1)} = x_1$
3: **for** $i = 2$ to d **do**
4: **for** $j = 1$ to i **do**
5: $\sigma_j^{(i)} = \sigma_j^{(i-1)} + x_i \sigma_{j-1}^{(i-1)}$
6: **end for**
7: **end for**

It is also well known [14] that if we have a polynomial $p(x)$, where α_i denotes its coefficients and β_is are the roots of $p(x)$,

$$p(x) = x^d + \ldots + \alpha_{d-1}x + \alpha_d = (x - \beta_1) \ldots (x - \beta_d),$$

then we have $\alpha_i = (-1)^{d-i}\sigma_i(\beta_1, \ldots, \beta_d)$.

Received 15th June 2015