ÜÇ TERİMLİ POLİNOMLAR İÇİN KARATSUBA BENZERİ ÇARPMA YÖNTEMLERİNİN ARAŞTIRILMASI

Sedat Akleylek, Nurşah Kaya

Bilgisayar Mühendisliği Bölümü, Ondokuz Mayıs Üniversitesi, Samsun, Türkiye

sedat.akleylek@bil.omu.edu.tr, nursahkaya93@gmail.com

ÖZET

Bu çalışmada, katsayıları tamsayı olan iki polinomu aritmetik karmaşıklık açısından daha verimli çarpan yöntemlerin araştırılması hedeflenmektedir. Bu yüzden, Böl-ve-Fethet mantığını kullanan, Karatsuba-Ofman Algoritmasından yola çıkarak çarpma işlemlerini daha az maliyetli toplama/çıkarma işlemleriyle değiştiren denklemler bulan bir yazılım geliştirilmiştir. Geliştirilen uygulamada, üç terimli iki polinomun katsayılarının olası kombinasyonları kullanılarak çarpma işleminden sonra bütün çarpım katsayılarının bulunup bulmadığını test edilmektedir. Üç terimli polinomları çarpmak için 3 farklı yöntem olduğu ve bu yöntemlerin hepsinde 6 çarpma, 13 toplama/çıkarma işlemine ihtiyaç duyulduğu hesaplanmıştır. Bunlara ek olarak, daha fazla terimli polinomların çarpımı için ne tür uygulamalara ihtiyaç duyulduğu konusunda detaylara da yer verilmiştir.

Anahtar Kelimeler: Polinom çarpımı, aritmetik karmaşıklık, sembolik hesaplama, Karatsuba-Ofman, Böl-ve-Fethet.

SEARCHING NEW KARATSUBA-LIKE POLYNOMIAL MULTIPLICATION ALGORITHMS FOR 3-TERM POLYNOMIALS

ABSTRACT

In this paper, new efficient methods are investigated to multiply two polynomials whose coefficients are integer. To achieve this, a software, based on divide-and-conquer idea, is developed with the help of Karatsuba-Ofman algorithm by replacing multiplication operations with addition/subtraction. This software checks all possible combinations of polynomial multiplications of three terms. With experimental results, there are three different methods to multiply 3-term polynomials with integer coefficients that need 6 multiplications and 13 additions. Moreover, the details are provided to extend this application for large dimensions.

Keywords: Polynomial multiplications, symbolic computation, complexity, Karatsuba-Ofman, divide-and-conquer

I. GİRİŞ (INTRODUCTION)

Bilgisayarlarda gerçekleştirilen bütün işlemler, mantıksal ve aritmetik işlemlere indirgenmektedir. Bilgisayar mimarisinde kullanılan temel aritmetik işlemler toplama ve çarpmadır. Sık kullanılmalarının nedeni; bilgisayar mimarisinde yapılan bir çok işlemin temelinde bu işlemlerin yer almasıdır [1]. Toplama işleminin maliyeti, doğrusal karmaşıklığa sahip olduğu için bazı durumlarda göz ardı edilmektedir. Çarpma işleminin maliyeti, toplama işleminin maliyetinden daha fazla olduğu için çarpmayı daha az maliyet ile gerçeklemek önem kazanmaktadır. Bu nedenle, bu çalışma kapsamında çarpma işlemine odaklanılacaktır.

Küçük sayılarda yapılan işlemlerde çalışma zamanı kabul edilebilir büyüklükte olsa da sayılar büyüdükçe işlemler yavaşlamakta hatta sonuca ulaşmak, işlemci gücüne göre farklılık gösterse de, yıllar almaktadır. İşlem gücü artan bilgisayarlara, yüklenen iş gücü de her geçen gün artmaktadır. Doğru orantılı olarak ilerleyen bu artış nedeni ile her zaman aritmetik işlemleri daha az maliyetle yapmanın yolu denenecektir. Bu yüzden çarpma işlemi üzerine olan

ilgi sürekli olarak devam etmektedir ve muhtemel olarak devam edecektir [2].

Klasik çarpma işleminin zaman karmaşıklığı $O(n^2)$'dir. Bugüne kadar bu karmaşıklığı azaltmak ve daha az maliyetli bir çarpma algoritması bulmak için çeşitli çalışmalar yapılmış ve yapılmaya da devam edilmektedir. Karatsuba algoritması ile bu alandaki çalışmalar tetiklenmiştir [3]. Böl-ve-Fethet mantığıyla çalışan bu algoritma, klasik çarpma algoritmasından asimptotik olarak daha iyi bir başarım gerçekleştirmektedir. $O(n^2)$ olan karmaşıklığı $O(n^{\log_2 3})$'e indirgemektedir. Karatsuba algoritmasının temelindeki mantık; denklemlerde bulunan bazı özelliklere sahip birden çok çarpma işlemini, daha az çarpma ile yapılmasıdır. Bunu gerçekleştirirken varolan çarpmalar, toplama/çıkarma işlemleri ile değiştirilmektedir. Karatsuba'dan sonra bu alana yönelim; kazancının fazla olması ve her alanda kullanılan uygulamaların performansını etkilemesi sebebiyle, gittikçe artmaktadır. Karatsuba algoritmasının ortaya çıkmasından sonra farklı bakış açıları da ortaya çıkmıştır. Toom-Cook algoritması, 1963'te Andrei Toom tarafından bulunmuştur. Bu algoritma, Karatsuba algoritması gibi Böl-ve-Fethet mantığıyla çalışmaktadır. Fakat; büyük boyutlardaki iki sayıyı, iki eşit parçaya bölmek yerine, l uzunluğunda k eşit parçaya bölerek; başka bir ifade ile interpolasyon mantığını kullanarak çarpma işlemlerini gerçekleştirmekte ve karmaşıklığı azaltmaktadır. Toom-Cook-3 algoritması, k=3 olduğu durumdur ve karmaşıklığı $O(n^{\log_3 5})$'tir [4]. 1971'de A. Schönhage ve V. Strassen tarafından, sadece herhangi iki büyük sayının veya fazla terimli polinomların çarpımını hesaplamaya yönelik hızlı Fourier dönüşümüne dayalı $O(n \log n \log\log n)$ karmaşıklığında olan Schönhage-Strassen algoritması geliştirilmiştir. Hızlı Fourier dönüşümü, ayrık Fourier dönüşümünü $O(n \log n)$ karmaşıklık ile gerçekleştiren bir tekniktir [5]. Daha sonra 2007 yılında Schönhage-Strassen'den daha hızlı, $log^* x := \min\{k \in N : log^{ok} x \leq 1\}$, $log^{ok} := \log o \ldots_{kx} o\ log$ olmak üzere işlem karmaşıklığını $O(n \log n\, 2^{O(log^* n)})$'e düşüren Fürer algoritması, Martin Fürer tarafından bulunmuştur [6].

Bu çalışmaların temel amacı çarpma işleminin aritmetik karmaşıklığının azaltılmasını sağlamak ve ihtiyaç duyulan küçük boyutlu çarpma sayısını azaltmaktır. Bu çalışma kapsamında; katsayıları tamsayı olan iki polinomu çarpmak için gereken çarpma sayısını azaltarak, polinom çarpımındaki verimliliğin arttırılması amaçlanmaktadır. Katsayıları ifade eden bazı çarpmaları, daha az maliyetli olan toplama ve çıkarma işlemleriyle değiştirerek çarpma sayısını azaltan denklemlerin elde edilmesi hedeflenmektedir. Bu denklemleri elde edilmesi sırasında dikkat edilmesi gereken bir diğer husus ise toplama/çıkarma karmaşıklığıdır. Çarpma sayısı azaltılırken, arttırılan toplama/çıkarma işlemi sayısı, çarpma karmaşıklığını geçmemelidir. Bu yüzden yeni denklemler elde edildikten sonraki hedef aynı çarpma sayısına sahip, toplama/çıkarma sayısı daha az olan denklem gruplarının bulunmasıdır.

A. Motivasyon ve Katkı

Karatsuba'dan sonra bu alana ilgi artmış fakat Böl-ve-Fethet mantığıyla çalışan Karatsuba'dan farklı denklemler ile çarpma sayısını azaltmaya yönelik çok fazla çalışma yapılmamıştır. 2005 yılında, Montgomery'nin [7] numaralı çalışmasında; 5, 6 ve 7 terimli polinomların çarpımında Karatsuba denklemleri gibi denklemler arama yöntemi ile varolan çarpma sayısı bazı tek değerler için azaltılmıştır. Fakat bu çalışmada denklem arama algoritması açıklanmamıştır. [8] numaralı çalışmada yazarlar, n^2 çarpma ve $(n-1)^2$ toplama/çıkarma işlemi kullanarak polinom çarpımını gerçekleştiren Klasik çarpma yöntemi ile Karatsuba yöntemini karşılaştırmıştır. Bu karşılaştırmanın sonucunda çarpma ve toplama işlemlerinin birbiri türünden maliyetini hesaplanarak, kazanç belirlenmiş ve beş terimli iki polinomun çarpımı için Karatsuba'dan daha verimli bir çarpma yöntemi önerilmiştir. Aynı zamanda farklı bölme boyutlarına göre (örneğin, ikiye parçalama yerine herhangi bir asal sayıya göre parçalama) Karatsuba algoritmasının genelleştirilmesi yapılmıştır.

İşlemci firmaları özellikle kriptografik işlemleri verimli yapabilmek amacıyla polinom çarpımları için özel teknikler kullanmakta ve bunlar için Böl-ve-Fethet yaklaşımı baz alınarak alt işlemciler tasarlamaktadırlar [9]. Bunlar göz önüne alındığında küçük boyutlu polinomların çarpımı, detaylı olarak araştırılması gereken önemli bir konu olduğu ortaya çıkmaktadır.

Bu çalışma, Karatsuba algoritması ile aynı çarpma sayısına ve daha az toplama/çıkarma sayısına sahip çarpma yöntemlerini bulmak üzere yapılmıştır. Sembolik hesaplama tabanlı çalışan arama algoritması için bir veri yapısı tasarlanmış ve uygulanmıştır.

B. Organizasyon

Bölüm 2'de üç terimli iki polinomun çarpımı için çarpma yöntemlerinin bulunmasına ve bu yöntemlerde kullanılan veri yapısına detaylı olarak değinilmektedir. Veri yapısı içerisindeki sembolik hesaplama işlemlerinden ayrıntılı olarak bahsedilmektedir. Bölüm 3'de daha fazla elemana sahip polinomların çarpımı için bazı öneriler ve sonuçlar verilmektedir. Ayrıca, gelecek çalışmalar hakkında açıklamalar detaylandırılmıştır.

II. ÜÇ TERİMLİ İKİ POLİNOMUN VERİMLİ ÇARPMA YÖNTEMLERİNİN BULUNMASI (FINDING EFFICIENT MULTIPLICATION METHODS FOT TWO POLYNOMS HAVING THREE TERMS)

Bu bölümde, Karatsuba çarpma yöntemi hatırlatılmakta, çalışmanın temelini oluşturan arama

algoritması verilmekte ve daha sonra Karatsuba benzeri çarpma yöntemlerini bulabilmek amacıyla oluşturulan yazılımın detayları anlatılmaktadır.

Karatsuba çarpım gruplarında, eşitlik (1)'deki n terimli iki polinomun çarpımındaki katsayıların kombinasyonlarının oluşturduğu denklemlerin toplanması veya çıkarılması ile eşitlik (2)'deki sonuç denklemlerinin katsayıları elde edilmektedir.

$$\begin{aligned} a(x) &= a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \\ b(x) &= b_{n-1}x^{n-1} + \ldots + b_1x + b_0 \end{aligned} \quad (1)$$

$$\begin{aligned} a(x)\,b(x) = {} & a_{n-1}b_{n-1}x^{2(n-1)} + \ldots \\ & + (a_0b_1 + a_1\,b_0)\,x + a_0b_0 \end{aligned} \quad (2)$$

Karatsuba denklemlerinin bulunma mantığı, çalışmanın gidişini anlamak açısından önemli bulunmaktadır. Bu yüzden, eşitlik (3)'deki üç terimli iki polinomun çarpımında kullanılan Karatsuba denklemleri eşitlik (5)'te detaylandırılmıştır.

$$\begin{aligned} a(x) &= a_2x^2 + a_1x + a_0 \\ b(x) &= b_2x^2 + b_1x + b_0 \end{aligned} \quad (3)$$

$$\begin{aligned} a(x)\,b(x) = {} & a_2b_2x^4 + (a_1b_2 + a_2b_1)\,x^3 \\ & + (a_2b_0 + a_1b_1 + a_0b_2)\,x^2 + (a_0b_1 + a_1b_0)x + a_0b_0 \end{aligned}$$

Üç terimli iki polinomun çarpımında oluşan 5 katsayı eşitlik (4)'de bulunmaktadır:

$$\begin{aligned} Ks[1] &= a_0b_0 \\ Ks[2] &= a_2b_2 \\ Ks[3] &= a_0b_1 + a_1b_0 \\ Ks[4] &= a_1b_2 + a_2b_1 \\ Ks[5] &= a_2b_0 + a_1b_1 + a_0b_2 \end{aligned} \quad (4)$$

Bu katsayıları elde etmek için 9 çarpmayı, 6 çarpmaya düşüren Karatsuba çarpımları ve denklemleri eşitlik (5)'te bulunmaktadır:

$$\begin{aligned} M_0&: a_0b_0 \quad M_3: (a_0+a_1)(b_0+b_1) \\ M_1&: a_1b_1 \quad M_4: (a_1+a_2)(b_1+b_2) \\ M_2&: a_2b_2 \quad M_5: (a_2+a_0)(b_2+b_0) \end{aligned} \quad (5)$$

$$\begin{aligned} Ks[1] &= M_0, \quad Ks[3] = M_3 - M_0 - M_1 \\ Ks[2] &= M_2, \quad Ks[4] = M_4 - M_1 - M_2 \\ Ks[5] &= M_5 - M_0 - M_2 \end{aligned}$$

A. Karatsuba Benzeri Çarpma Yöntemlerini Arama Algoritması

Bu çalışmada, üç terimli polinomların çarpımlarını üretirken Karatsuba Algoritması'nın temelde kullandığı mantıkla ilerleyerek, tüm olasılıkları kullanan bir arama algoritması tasarlanmıştır. Bu algoritmanın adımları şu şekildedir:

1. $(a_2x^2 + a_1x + a_0)$ ve $(b_2x^2 + b_1x + b_0)$ polinomlarının çarpımındaki katsayıların bütün olası kombinasyonları eşitlik (6)'daki şekilde hesaplanmış ve bütün M çarpımları kaydedilmiştir. Ek_1'de bütün M çarpımları gösterilmiştir.

$$M = (d_0a_0 + d_1a_1 + d_2a_2)(e_0b_0 + e_1b_1 + e_2b_2) \quad \forall\, d_i, e_i \in \{0,1\} \quad (6)$$

2. 49 çarpmadan, istenilen çarpma sayısınca yani 6'sı seçilerek bir çarpma grubu oluşturulmaktadır. Üç terimli iki polinomun çarpımında, $\binom{49}{6}$ olası çarpma grubu bulunmaktadır.

3. Bu çarpmalar, olabilecek bütün toplama/çıkarma işlemlerine tabi tutulmaktadır. Öncelikle seçilen 6 M değerinden biri rastgele alınmakta ve başa getirilmektedir. Daha sonra kalan M değerlerinin toplama ve çıkarmadan oluşan bütün kombinasyonları eşitlik (7)'deki şekilde hesaplanmaktadır.

$$\binom{6}{1}[\,2\binom{5}{1} + 2^2\binom{5}{2} + 2^3\binom{5}{3} + 2^4\binom{5}{4} + 2^5\binom{5}{5}] \quad (7)$$

Bu işlem sonucunda 1432 tane denklem ortaya çıkmaktadır. Bunlar, M çarpımlarından oluşan k denklemlerini ifade etmektedir. Bu k denklemleri eşitlik (8)'de bulunmaktadır.

$$k = M_{40} - M_{21} + M_7 + M_{34} - M_{10} + M_{18} \quad (8)$$

4. Oluşan her k denklemi, sonuç katsayıları ile karşılaştırılmaktadır. 1432 denklem içinde 5 sonuç katsayısını da sağlayan denklemler bulunduğunda, bu denklem grubu sonuç denklem grubuna eklenmektedir. Olası M çarpımlarının hepsi bitene kadar 2. adıma dönülmekte ve yeni M çarpımları seçilerek, sonraki adımlar tekrarlanmaktadır. Olası M çarpımlarının hepsi tarandıktan sonra bir sonraki adıma geçilmektedir.

5. En son bulunan bütün denklem grupları içinden en az çarpma ve toplama/çıkarma işlemiyle bütün katsayıları üreten denklem grubu aranmaktadır. Amaç en az çarpma ile çarpma ve toplama/çıkarma oranını minimize edilmesidir. Sonuç denklemlerinin çarpma ve toplama/çıkarma oranı, eşitlik (9)'da belirtilen yapıya göre kontrol edilmektedir.

Yazarlar, [8] numaralı çalışmada, bu sayıyı kontrol etmek için bir r oranı belirlemiştir. Bunu basit olarak göstermek gerekirse; n terimli iki polinomun çarpımını n^2 çarpma işlemi ve $(n-1)^2$ toplama işlemi ile gerçekleştiren klasik çarpma yöntemi ile, üç terimli iki polinomun çarpımı için daha az maliyetli çalıştığı bilinen Karatsuba algoritmasının karşılaştırılması eşitlik (9) ve eşitlik (10)'da bulunmaktadır.

$$\begin{gathered} r = tm / ta \\ tm: 1 \text{ çarpma işleminin maliyeti} \\ ta: 1 \text{ toplama işleminin maliyeti} \\ cs \text{ (klasik çarpma yöntemi)} = 4ta + 9tm \\ ck \text{ (Karatsuba Algoritması)} = 13ta + 6tm \\ ck < cs \end{gathered} \quad (9)$$

$$13\ ta + 6\ tm < 4ta + 9\ tm \quad (10)$$

$$9\ ta < 3tm$$

$$3 < tm / ta$$

$$3 < r$$

Eşitlik (10)'daki karşılaştırma işleminin sonucunda 3<r oranı elde edilmektedir. Başka bir deyişle bir çarpma, üç toplamadan daha maliyetlidir [8].

B. Veri Yapısı

Bu kısımda, çalışma için yapılan uygulama içerisinde sembolik hesaplama işlemi için kullanılan veri yapısından ve bu veriler üzerindeki sembolik hesaplama işlemlerinden bahsedilmektedir.

Global sınıfı, gerekli değişkenleri ve fonksiyonları diğer sınıflardan soyutlanmış olarak tanımlayarak gerektiğinde bütün sınıflarda kullanılabilmesini sağlamakta ve sınıf kütüphanesi olarak adlandırılmaktadır. Diğer sınıflar içerisinde, her fonksiyondan erişime açık global bir değişken tanımlamak yerine; Global sınıfı içerisinde değişkenler kullanılmıştır.

1) Genel Yapıyı Oluşturan Fonksiyonlar:

Şekil 1. Fonksiyon şeması

Şekil 1'de sembolik hesaplama için oluşturulan yazılımın fonksiyonel şeması kabaca verilmiştir. Bu kısımdaki fonksiyonların kapsamlı şeması EK-2'de verilmiştir. Kombinasyon işlemlerini hesaplamak için kullanılan bazı fonksiyonları anlatmaya gerek duyulmamıştır.

Main() fonksiyonunda, girilen polinomların katsayıları belirlenmekte ve iki polinomun çarpımındaki katsayıların bütün olası kombinasyonları hesaplanarak eşitlik (11)'deki M ve Eşitlik (12)'deki $Toplam$ dizilerine iki farklı şekilde yazdırılmaktadır. Bu iki dizi aslında aynı kavramı ifade etmektedir. Fakat, ayrı yerlerde kullanılmak üzere tasarlanmıştır. M dizisinde, çarpımların dağılmış hali tutulmaktadır.

$$M_i = \text{“}a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1\text{”} \quad (11)$$

M dizisi, M çarpımları üzerinde sembolik işlemleri uygularken kolaylık sağlanması, her seferinde yapılması gereken çarpma işlemini kaldırmak ve yalnızca sembolik toplama/çıkarma işlemiyle uğraşmak amacıyla tanımlanmıştır. Bundan sonra bütün işlemler M dizisi ile gerçekleştirilecektir. Oluşturulduktan sonra üzerine herhangi bir işlem uygulanmayacaktır. Bu yüzden global sınıfında tanımlanmıştır. $Toplam$ dizisinde ise, çarpımların dağılmamış hali tutulmaktadır.

$$Toplam_i = \text{“}(a_0 + a_1)(b_0 + b_1)\text{”} \quad (12)$$

Daha sonra yapılması planlanan, sonuç denklemleri üzerinde toplama sayısını hesaplamak için benzer yapıların olup olmadığını kontrol etmek amacıyla oluşturulmuştur. M dizisinde olduğu gibi bütün değişikliklere kapalı bulunmaktadır.

Bundan sonra M_asil_atama() fonksiyonu çağrılmaktadır. Bu fonksiyonda, M dizisinin elemanlarının indislerinin 6'lı kombinasyonları alınarak, eşitlik (13)'deki M_Asil dizisi üretilmektedir. Burada M_Asil dizisinin elemanları;

$$M\_Asil_i = \{\text{‘17’, ‘5’, ‘42’, ‘29’, ‘34’, ‘16’}\} \quad (13)$$

şeklinde, sayılardan oluşan string değerleridir. Her eleman için iç içe bir for döngüsü açılmaktadır. İlk for döngüsü 0'dan başlamakta ve 49'a kadar gitmektedir. Daha sonraki for döngüleri, bir üstteki for döngüsünün başlangıç değerinin bir fazlasından başlamakta ve 49'a kadar gitmektedir. Sona doğru yaklaştıkça hesaplama sayısı azalmaktadır. Bu kısımda iç içe açılan for'ları parçalara bölerek farklı bilgisayarlarda ya da aynı bilgisayarda paralel olarak çalıştırmak zamandan kazanç sağlamaktadır. Bir işi birden çok iş bölümüne ayırarak eş zamanlı olarak çalıştırmak, işlemlerin daha kısa zamanda yapılmasına olanak sağlamaktadır. Bu çalışmada, for döngülerini eş zamanlı olarak çalıştırarak sonuçlar elde edilmektedir. Her döngüde üretilen M_Asil dizisi, Carpma_Hesaplama() fonksiyonuna parametre olarak gönderilmektedir. Bütün fonksiyonlar çalıştırılmakta ve sonuç değerleri üretilmektedir. En son M_asil_atama() fonksiyonuna geri dönülmekte ve bütün for'lar sonlanana kadar yeni bir M_Asil dizisi üretilmektedir. Fakat yeni bir dizi oluşturulmadan önce gerekli global değişkenler sıfırlanarak yeni dizi için hazır hale getirilmektedir.

Buraya kadar anlatılan kısımda temel amaç, işlemler başlamadan önce kullanılacak ve değişmeyecek bazı global değişkenlerin tanımlanması ve ana yapıdan bahsedilmesidir. Bundan sonraki kısımda M çarpımları üzerindeki sembolik hesaplamanın nasıl yapıldığına değinilecektir.

2) Sembolik İşlemleri Gerçekleştiren Fonksiyonlar:

Carpma_Hesaplama() fonksiyonu, M_Asil dizisinin elemanlarından birini başa koyulmakta ve kalan 5 değerin olası bütün kombinasyonları bulunmaktadır. İfade (14) bu kombinasyonların sayısı verilmektedir.

$$\binom{6}{1}[\binom{5}{0} + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} + \binom{5}{4} + \binom{5}{5}] = 196 \quad (14)$$

Bütün toplama/çıkarma içeren olasılıkların taranması gerektiği için bu dizinin her elemanı tek tek arti_eksi() fonksiyonuna yollanmakta ve toplama/çıkarma işlemi içeren k denklemleri elde edilmektedir. Burada bütün elemanları aynı anda yollamak ve hesaplanan değerleri tek bir dizi içerisinde tutmak, bilgisayarların işlem gücünü ve hafızasını zorladığı için her eleman ayrı olarak yollanmaktadır. Bu sayede hepsinin sonucunu tutan, çok büyük boyutlu bir dizi tanımlanması gerekmemektedir. Elde edilen her bir denklem; sonucu sağlayıp sağlamadığını kontrol edildikten sonra silinmektedir. Bu da bize hafızadan kazanç sağlamaktadır.

Toplama/çıkarma işlemi içeren eşitlik (8)'deki k denklemlerini bulmak için doğruluk tablosu mantığı kullanılmaktadır. Şöyle ki; diziler uzunluğuna bağlı olarak sayısı değişen for döngülerine girmektedirler. İç içe olan bütün for döngüleri iki defa dönmekte ve her dönüşte işaret değiştirilmektedir. Bu işlemler sonucunda bütün olası kombinasyonlar elde edilmektedir. Temel mantık, uzunluğu 2 olan denklemler için Algoritma 1'de örneklenmektedir.

Bu kısımda elde edilen işaretli k denklemleri kaydedilmemektedir. Eşitlik (8)'de tanımlı k denklemi sistemde eşitlik (15)'deki şekilde tutulmaktadır.

$$k_i = 40 - 21 + 7 + 34 - 10 + 18 \quad (15)$$

Algoritma 1: Uzunluğu 2 Olan k Denklemleri

```
for i to 2 do
   for j to 2 do
      if s(0) = "+" Then
         m_string = s(1) + arr(1) + s(0) + arr(0)
         s(0) = "-"
      else if s(1) = "-" Then
         m_string = s(1) + arr(1) + s(0) + arr(0)
         s(0) = "+"
      end if
   end for
   if s(1) = "+" Then
      s(1) = "-"
   else
      s(1) = "+"
   end if
end for
```

Burada bulunan sayılar ile M çarpımlarının index değerleri temsil edilmektedir. Unutulmamalıdır ki; burada yer alan sayılar ile standart aritmetik işlemler değil, sembolik hesaplama işlemleri yapılmaktadır. Elde edilen işaretli k denklemi, m_cevir() fonksiyonuna yollanmaktadır. Bu fonksiyon kendisine gelen işaretli k denkleminde eşitlik edilen M çarpımlarının indislerini almakta ve M dizisinde yerlerine koyarak Eşitlik (16)'da gösterilen değişimi gerçekleştirmektedir.

Girdi = "0+16-32+20" dizisinin ifade ettiği denklem,

$$\text{"0+16-32+20"} = M_0 + M_{16} - M_{32} + M_{20} \quad (16)$$

$$\text{Çıktı} = a_0b_0 + a_2b_2 - a_0b_0 - a_0b_2 - a_2b_0 - a_2b_2 + a_2b_0 + a_2b_1 + a_2b_2$$

$$= a_2b_1 + a_2b_2 + a_0b_2$$

Buradaki çevrim işleminde algoritma şu şekilde çalışmaktadır: String olarak tutulan k denkleminin birinci elemanını almakta ve bir sonraki eleman '+' veya '-' mi diye kontrol etmektedir. Eğer '+' veya '-' ise tek basamaklı bir sayı olduğuna karar vermektedir. İki veya daha fazla basamaklı olma durumunu da aynı şekilde kontrol etmektedir. İfade (7)'de gösterildiği gibi, bu yapıdaki M çarpımlarının sayısı 49 olduğu için üç basamaklı indis değeri bulunmamaktadır. Bu yüzden, basamak değeri ikiye kadar kontrol edilmektedir. Üçten fazla terimli iki polinomun çarpımı için burada bulunan basamak sayısı kontrol edilmeli ve bunun için gerekli olan arttırma yapılmalıdır. Basamak sayısına karar verdikten sonra, string içinden çekilen karakterler integer türüne çevrilmektedir. Daha sonra M dizisinin indisi olarak kullanılmakta ve M dizisinde ifade ettiği çarpım değeri çıktı değişkenine atanmaktadır. Her indis için bu işlemler tekrar edilmektedir. Burada en önemli nokta: çevrim işleminden sonra M çarpımlarını birleştirirken işarete dikkat edilmesidir. İşaret '-' olduğu zaman çarpımın içindeki bütün '+' işaretleri, '-' işaretlerine çevrilmekte ve denklemin başına '-' işareti koyulmaktadır. Örnek olarak Eşitlik (16)'da girdi olarak gelen k denkleminin çevrim işlemini adım adım anlatılmıştır:

1. Bu denklemde ilk eleman 0 olduğu için M dizisinin M_0 elemanına gidilmektedir. M_0 elemanı 'a_0b_0' denk gelmektedir. 'a_0b_0' çıktı değişkenine atanmaktadır. 0'dan sonra gelen eleman '+' veya '-' ise işaret belirlenmekte ve bir sonraki işlemi etkileyeceği için işaret değişkenine atanmaktadır. Bu değişken sayesinde M dizisinin elemanları önlerinde bulunan işarete göre yeniden düzenlenmektedir. Eğer sonradan gelen eleman '+' veya '-' değil ise, '+' veya '-' bulunana kadar her eleman birleştirilmekte daha sonra indis olarak kullanılmaktadır.

   çıktı = M_0 şeklinde değiştirilmektedir.

2. "16-" değeri için 1'i ve 6'yı almakta ve '-' karakterini görünce durmaktadır. Daha sonra "16" stringini integer değere dönüştürmektedir. Bir

önceki işlemden gelen işaret değişkeni '+' olduğu için; M_{16} denkleminin içi olduğu gibi bırakılmakta ve başına '+' işareti atanmaktadır. Çıkarken bir sonraki eleman '-' karakteri olduğu için işaret değişkeni '-' yapılmaktadır.

çıktı = çıktı + "+" + M_{16} şeklinde değiştirilmektedir.

3. Bir sonraki indis 32'dir. işaret değeri '-' olduğu için M_{32} denkleminin içinde bulunan bütün '+' işaretleri, '-' işaretine dönüştürülmekte ve başına '-' atanmaktadır. Çıkarken bir sonraki eleman '+' karakteri olduğu için işaret değişkeni '+' yapılmaktadır.

çıktı = çıktı + "-" + M_{32}.replace('+','-') şeklinde değiştirilmektedir.

4. Bir sonraki indis 20'dir. İşaret değeri artı olduğu için M_{20} denklemi olduğu gibi çıktı değerine eklenmekte ve başına '+' işareti atanmaktadır.

çıktı = çıktı + "+" + M_{20} şeklinde değiştirilmektedir.

Yukarıda anlatılan 4 adımda yapılan işlemler her *k* denklemi için yeniden tekrar edilmektedir. Adım sayısı denklemin uzunluğuna bağlı olarak değişmektedir. En son oluşan çıktı değeri Denklem_Sonuc() fonksiyonuna gönderilmektedir.

Bu fonksiyon sembolik toplama ve çıkarma işlemlerini gerçekleştirmekte ve string üzerindeki gerekli sadeleştirmeleri yaparak sonucu döndürmektedir.

Sadeleştirme işlemleri aşağıdaki şekilde gerçekleşmektedir:

1. String değeri, ilk bulunan '+' karakterine göre 2 parçaya bölünmekte ve bir diziye atanmaktadır. Daha sonra aynı string, ilk gelen '-' karakterine göre 2 parçaya bölünmekte ve başka bir diziye atanmaktadır.
2. Bu dizilerin uzunluğuna bakılarak gelen işaret belirlenmektedir. Uzunluğu daha küçük olan dizi, ilk gelen işaretin ne olduğunu belirlemekte ve bu değer işaret değişkenine atanmaktadır. İşaret değişkeni, önünde bulunan denklemi etkilediği için bir sonraki döngüde kullanılmaktadır.
3. İşaret değişkeni belirlendikten sonra işaret, '+' ise bu eleman '+' işaretli çarpımları tutan diziye, '-' ise '-' işaretli çarpımları tutan diziye eklenmektedir.
4. İki dizi karşılaştırılmakta ve aynı eleman bulunduğu durumda iki diziye de 'null' değeri atanmaktadır. Başka bir deyişle; farklı işaretli, aynı elemanlar birbirini sadeleştirmektedir.
5. Son olarak, ayrılan diziler birleştirilerek bir denklem oluşturulmaktadır. İki dizinin elemanları bir dizide birleştirilirken 'null' değeri ile karşılaşıldığında dizi bir kaydırılmaktadır.

Daha sonra bu sonuç, karsilastir_ve_bitir() fonksiyonuna gönderilmektedir. Herhangi bir katsayı ile eşleşip eşleşmediği kontrol edilmektedir. Bu kontrol işlemi, sonuç ifadesinin katsayıdan çıkarılması ile gerçekleştirilmektedir. Eğer bulunan değer 'null' ise katsayı ile eşleşmektedir. Bu yüzden katsayıların kontrol edildiği dizide eşleşen katsayı bir arttırılmaktadır. Daha sonra Carpma_Hesaplama() fonksiyonuna dönülmektedir. Burada M_Asil dizisinden oluşturulan kombinasyonlardan bir diğeri alınmakta ve aynı işlemler tekrar edilmektedir. Bütün kombinasyonlar kontrol edildikten sonra katsayıların hepsi sağlanıyorsa M_Asil dizisi diğer bir deyişle *M* çarpım grubu istenilen şartları sağlıyor denilmektedir. Daha sonra M_Asil_Atama() fonksiyonuna dönülmekte ve yeni bir M_Asil dizisi oluşturulmaktadır. Bütün M_Asil kombinasyonları tarandığında program sonlanmaktadır.

C. Bulunan Sonuçlar ve Karşılaştırma

Çalışma kapsamında, üç terimli iki polinomun çarpımında ortaya çıkan katsayıları, Karatsuba'dan farklı çarpım grupları ile elde edilmesi sağlanmıştır. Bu çarpım grupları aşağıdaki gösterilmiştir:

1.Grup

M çarpımları: 0, 8, 16, 24, 32, **48**

1. katsayı $(a_0b_0) = M_0$
2. katsayı $(a_2b_2) = M_{16}$
3. katsayı $(a_0b_1+a_1b_0) = M_{24} - M_8 - M_0$
4. katsayı $(a_1b_2+a_2b_1) = \boldsymbol{M_{48} - M_{32} - M_{24} + M_0}$
5. katsayı $(a_2b_0+ a_1b_1+a_0b_2) = M_8 + M_{32} - M_{16} - M_0$

2.Grup

M çarpımları: 0, 8, 16, 32, 40, **48**

1. katsayı $(a_0b_0) = M_0$
2. katsayı $(a_2b_2) = M_{16}$
3. katsayı $(a_0b_1+a_1b_0) = \boldsymbol{M_{48} - M_{32} + M_{16} - M_{40}}$
4. katsayı $(a_2b_0+ a_1b_1+a_0b_2) = M_{40} - M_8 - M_{16}$
5. katsayı $(a_2b_0+ a_1b_1+a_0b_2) = M_8 + M_{32} - M_{16} - M_0$

3.Grup (Karatsuba Ofman)

M çarpımları: 0, 8, 16, 24, 32, 40

1. katsayı $(a_0b_0) = M_0$
2. katsayı $(a_2b_2) = M_{16}$
3. katsayı $(a_0b_1+a_1b_0) = M_{24} - M_8 - M_0$
4. katsayı $(a_1b_2+a_2b_1) = M_{40} - M_8 - M_{16}$
5. katsayı $(a_2b_0+ a_1b_1+a_0b_2) = M_8 + M_{32} - M_{16} - M_0$

Koyu olarak yazılan çarpım denklemleri, Karatsuba çarpımlarına farklılık katan çarpım denklemleridir. Bu iki çarpım grubu da üç terimli iki polinomun çarpımını 6 çarpma, 13 toplama/çıkarma işlemiyle gerçekleştirmektedir. Üç terimli polinomların

denklem uzayı çok büyük olmadığı için Karatsuba'dan farklı çarpım grubu sayısı az bulunmaktadır. Çarpım uzayı büyüdükçe bunu sağlayan daha fazla çarpım grubu elde edileceği öngörülmektedir.

Çarpma sayısını hesaplanırken, seçilen M_Asil dizisinin eleman sayısına bakılmakta ve işlem yapılan *M* çarpımı kadar çarpmaya ihtiyaç duyulmaktadır.

Toplama sayısını hesaplanırken dikkat edilmesi gereken en önemli husus tekrar eden yapıların belirlenmesi ve daha önce hesaplanan bir yapının yeniden hesaplanmamasıdır. Toplama işlemi hesabını daha iyi anlamak için, 1.Grup için toplama/çıkarma sayısının hesaplanışı adım adım gösterilmiştir:

1.Grup *M* çarpımları:

$M_0 = a_0 b_0$

$M_8 = a_1 b_1$

$M_{16} = a_2 b_2$

$M_{24} = (a_0 + a_1)(b_0 + b_1)$

$M_{32} = (a_0 + a_2)(b_0 + b_2)$

$M_{48} = ((a_0 + a_1) + a_2)((b_0 + b_1) + b_2)$

İlk 3 çarpımda toplama/çıkarma bulunmamaktadır. M_{24} ve M_{32} çarpımlarında 2 tane toplama bulunmaktadır. M_{48} çarpımında toplama sayısı 4 gibi görünse de daha önce hesaplanan parantez içindeki yapılar bir daha hesaplanmayacağı için 2 toplama bulunmaktadır.

1.Grup katsayı denklemleri:

1. katsayı $(a_0 b_0) = M_0$
2. katsayı $(a_2 b_2) = M_{16}$
3. katsayı $(a_0 b_1 + a_1 b_0) = (M_{24} - M_8) - M_0$
4. katsayı $(a_1 b_2 + a_2 b_1) = \boldsymbol{M_{48} - M_{32} - (M_{24} + M_0)}$
5. katsayı $(a_2 b_0 + a_1 b_1 + a_0 b_2) = M_8 + M_{32} - M_{16} - M_0$

İlk iki katsayıda toplama/çıkarma bulunmamaktadır. 3. katsayıda 2 çıkarma işlemi bulunmaktadır. 4. katsayıda çıkarma işlemi sayısı 3 gibi görünse de, 3. katsayıda hesaplanan parantez içindeki yapı bir daha hesaplanmamaktadır. Bu yüzden 4. katsayıda 2 çıkarma işlemi bulunmaktadır. 5. katsayıda 4 toplama/çıkarma işlemi bulunmaktadır. Bu işlemde dikkat edilmesi gereken nokta: toplama/çıkarma sayısını azaltmak için paranteze alınan yapıların, başka bir eleman ile paranteze alınanamasıdır.

Sonuç olarak, M çarpımından ve katsayı denklemlerinden gelen toplama/çıkarma işlemi sayıları eşitlik (17)'deki şekilde toplanmaktadır. Elde edilen sonuçlar Tablo 1'de özetlenmiştir.

(2 + 2 + 2) + (2 + 2 + 3) = 13 toplama/çıkarma (17)

Tablo 1. İşlem Sayısı Karşılaştırması

	İşlemler	
Denklem Grupları	**Çarpma**	**Toplama**
1. Grup	6	13
2. Grup	6	13
Karatsuba Algoritması	6	13

III. SONUÇLAR VE GELECEK ÇALIŞMALAR (CONCLUSIONS AND FUTURE STUDIES)

Bu çalışmada, üç terimli iki polinomun çarpımı için gerçeklenen uygulamayı, beş terimli iki polinomun çarpımı için de geliştirilmiş bulunmaktadır. Ancak arama uzayı çok büyük olduğu için detaylı sonuçları burada verilememektedir. Beş terimli iki polinomun çarpımında olası *M* çarpımlarının sayısı Eşitlik (18)'deki şekilde hesaplanmaktadır:

$$[\binom{5}{0} + \binom{5}{1} + \binom{5}{2} + \binom{5}{3} + \binom{5}{4} + \binom{5}{5}]^2 = 31.31 = 961 \quad (18)$$

Beklenilen 15 veya daha az çarpma ile polinom çarpımını gerçekleştirmektir. Bu yüzden çarpma sayısını 15 seçilmekte ve beş terimli iki polinomun çarpımındaki olası çarpma sayısı, $\binom{961}{15}$'li kombinasyonu şeklinde hesaplanmaktadır. Bu kombinasyonlar, eşitlik (13)'de gösterilen üç terimli polinomlar için oluşturulan M_Asil dizisini, beş terimli polinomlar için oluşturmaktadır. M_Asil dizisinin elemanlarından oluşturulan olası bütün pozitif denklemlerin sayısı Eşitlik (19)'daki şekilde hesaplanmaktadır.

$$\binom{15}{1}[\binom{14}{0} + \binom{14}{1} + \cdots + \binom{14}{13} + \binom{14}{14} \quad (19)$$

Sadece toplama içeren Eşitlik (19)'daki denklemlerden, toplama ve çıkarma içeren denklemler elde edilmektedir. Bütün durumları içeren denklemlerin sayısı eşitlik (20)'deki şekilde hesaplanmaktadır.

$$\binom{15}{1}[2\binom{14}{1} + 2^2\binom{14}{2} + \cdots + 2^{14}\binom{14}{14}] \quad (20)$$

Bu hesaplama işlemleri, n = terim sayısı olmak üzere aşağıdaki eşitlik için genelleştirilirse Eşitlik (21) ve (22)'deki işlemler elde edilmektedir.

$$a(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

$$b(x) = b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$$

$$a(x)\,b(x) = a_{n-1}b_{n-1}x^{2(n-1)} + \ldots + (a_0 b_1 + a_1 b_0)x + a_0 b_0$$

n terimli iki polinomun çarpımdaki *M* çarpımlarının sayısı:

$$\left(\sum_{m=0}^{n}\binom{n}{m}\right)^2 \quad (21)$$

Bütün durumları içeren *k* denklemlerinin sayısı:

$$\binom{k}{1}\sum_{i=1}^{k} 2^i\binom{k}{i} \quad (22)$$

Beş terimli iki polinomun çarpımında, yeterli hesaplama gücüne sahip olunmadığı için çarpım sonuçları elde edilememiştir. Uygulama, hesaplama gücü arttırıldığında daha fazla terimli polinom çarpımlarını gerçekleştirebilecek şekilde tasarlanmıştır. İçerisindeki sembolik hesaplama işlemleri çözüme ulaşma süresini arttırmaktadır. Uygun denklemleri bulmak için yapılan hesaplamalar tamamen paralel hale getirilirse, programın çok daha verimli çalışacağı ve daha çok terimli polinom çarpımlarındaki katsayı denklemlerini de bulabileceği öngörülmektedir.

Bu çalışmada, sembolik hesaplama ile üç terimli iki polinomun çarpımı için olası tüm çarpma yöntemlerini elde eden bir arama algoritması tasarlanmış ve bunun uygulaması gerçeklenmiştir. Oluşturulan yazılımın daha fazla elemana sahip polinomlar için de kullanılabileceği belirtilmiş ve bunlar için gerekli olan tüm hesaplamaların nasıl yapılacağı genelleştirilerek açıklanmıştır.

BİLGİ VE TEŞEKKÜR

Bu çalışma, 116E279 proje numarası ile TÜBİTAK tarafından desteklenmiş ve ISCTurkey 2017 etkinliğinde sunulmuş ve en iyi bildiri ödülü almıştır.

KAYNAKLAR

[1] Von zur Gathen, J. ve J. Gerhard, Modern Computer Algebra, 3rd ed., Cambridge University Press, 2013.

[2] Knuth, D. The Art of Computer Programming, 3rd ed., vol. 2. Seminumerical Algorithms, Addison-Wesley Longman, Amerika, 1997.

[3] Karatsuba, A. ve Y. Ofman, "Multiplication of Many-Digital Numbers by Automatic Computers", Physics-Doklady, 1963.

[4] Cook, T. ve A. Stephen, On the Minimum Computation Time of Functions, Doktora Tezi, Harvard University, Department of Mathematics, 1966.

[5] Heideman, M., D. Johnson ve C. Burrus, "Gauss and the history of the fast fourier transform", IEEE ASSP Dergisi, Cilt 1, 14-21, 1984.

[6] Fürer, M. "Faster Integer Multiplication", Pennsylvania State University, Amerika, 2007.

[7] Montgomery, P. L. "Five, Six, and Seven-Term Karatsuba-Like Formulae", IEEE Transactions On Computers Dergisi, Cilt 54, Numara:3, 2005, syf. 362-369.

[8] Paar, C. ve A. Weimerskirch, "Generalizations of the Karatsuba Algorithm for Efficient Implementations", Ruhr-Universty Bochum, Almanya, 2006, http://eprint.iacr.org/2006/224.pdf (Son erişim tarihi: 15 Mart 2017).

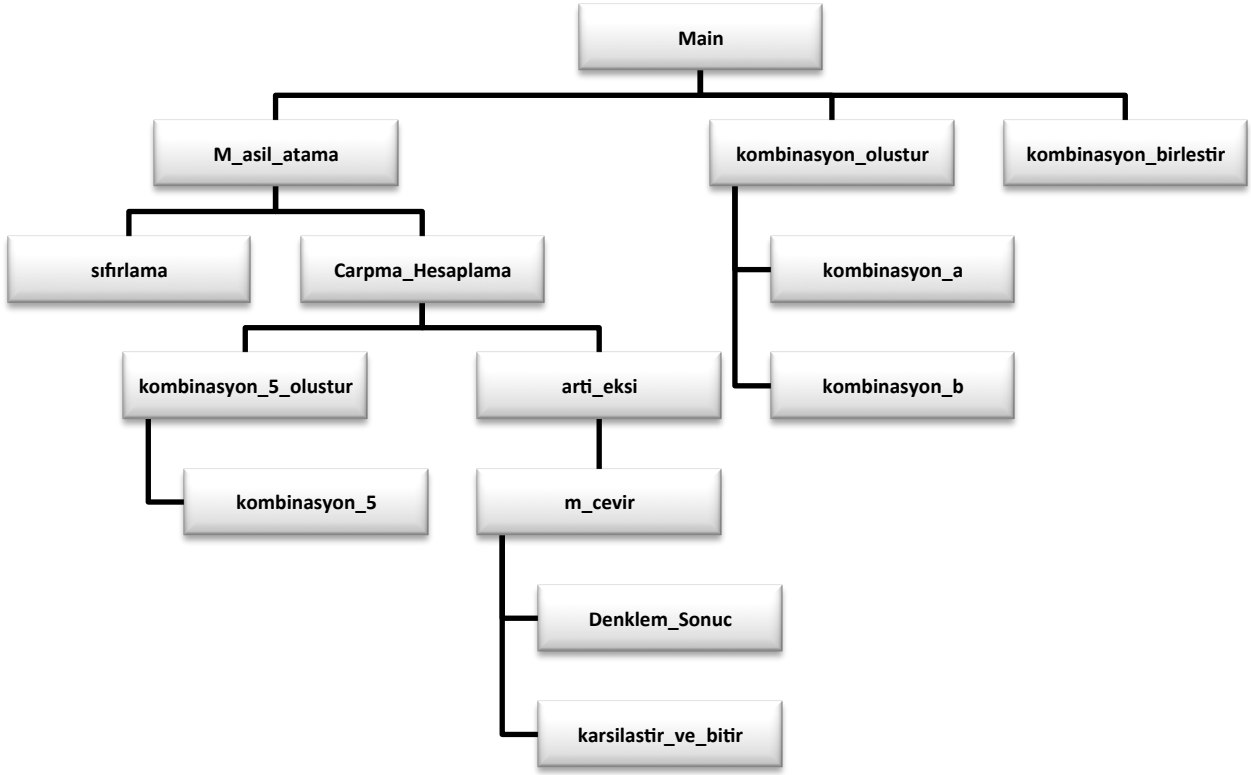
[9] Jankoqski, K., P. Laurent ve A. O'Mahony, Intel Polynomial Multiplication Instruction and its Usage for Elliptic Curve Cryptography, Intel White Paper, 2012, http://www.intel.co.kr/content/dam/www/public/us/en/documents/white-papers/polynomial-multiplication-instructions-paper.pdf (Son erişim tarihi: 15 Mart 2017)

EK-1 *M* Çarpımları:

M_i	M_Asil[] Dizisi	Toplam[] Dizisi
M_0:	a_0*b_0	$(a_0)*(b_0)$
M_1:	a_0*b_1	$(a_0)*(b_1)$
M_2:	a_0*b_2	$(a_0)*(b_2)$
M_3:	$a_0*b_0 + a_0*b_1$	$(a_0)*(b_0 + b_1)$
M_4:	$a_0*b_0 + a_0*b_2$	$(a_0)*(b_0 + b_2)$
M_5:	$a_0*b_1 + a_0*b_2$	$(a_0)*(b_1 + b_2)$
M_6:	$a_0*b_0 + a_0*b_1 + a_0*b_2$	$(a_0)*(b_0 + b_1 + b_2)$
M_7:	a_1*b_0	$(a_1)*(b_0)$
M_8:	a_1*b_1	$(a_1)*(b_1)$
M_9:	a_1*b_2	$(a_1)*(b_2)$
M_{10}:	$a_1*b_0 + a_1*b_1$	$(a_1)*(b_0 + b_1)$
M_{11}:	$a_1*b_0 + a_1*b_2$	$(a_1)*(b_0 + b_2)$
M_{12}:	$a_1*b_1 + a_1*b_2$	$(a_1)*(b_1 + b_2)$
M_{13}:	$a_1*b_0 + a_1*b_1 + a_1*b_2$	$(a_1)*(b_0 + b_1 + b_2)$
M_{14}:	a_2*b_0	$(a_2)*(b_0)$
M_{15}:	a_2*b_1	$(a_2)*(b_1)$
M_{16}:	a_2*b_2	$(a_2)*(b_2)$
M_{17}:	$a_2*b_0 + a_2*b_1$	$(a_2)*(b_0 + b_1)$
M_{18}:	$a_2*b_0 + a_2*b_2$	$(a_2)*(b_0 + b_2)$
M_{19}:	$a_2*b_1 + a_2*b_2$	$(a_2)*(b_1 + b_2)$
M_{20}:	$a_2*b_0 + a_2*b_1 + a_2*b_2$	$(a_2)*(b_0 + b_1 + b_2)$
M_{21}:	$a_0*b_0 + a_1*b_0$	$(a_0 + a_1)*(b_0)$
M_{22}:	$a_0*b_1 + a_1*b_1$	$(a_0 + a_1)*(b_1)$
M_{23}:	$a_0*b_2 + a_1*b_2$	$(a_0 + a_1)*(b_2)$
M_{24}:	$a_0*b_0 + a_0*b_1 + a_1*b_0 + a_1*b_1$	$(a_0 + a_1)*(b_0 + b_1)$
M_{25}:	$a_0*b_0 + a_0*b_2 + a_1*b_0 + a_1*b_2$	$(a_0 + a_1)*(b_0 + b_2)$
M_{26}:	$a_0*b_1 + a_0*b_2 + a_1*b_1 + a_1*b_2$	$(a_0 + a_1)*(b_1 + b_2)$
M_{27}:	$a_0*b_0 + a_0*b_1 + a_0*b_2 + a_1*b_0 + a_1*b_1 + a_1*b_2$	$(a_0 + a_1)*(b_0 + b_1 + b_2)$
M_{28}:	$a_0*b_0 + a_2*b_0$	$(a_0 + a_2)*(b_0)$
M_{29}:	$a_0*b_1 + a_2*b_1$	$(a_0 + a_2)*(b_1)$
M_{30}:	$a_0*b_2 + a_2*b_2$	$(a_0 + a_2)*(b_2)$
M_{31}:	$a_0*b_0 + a_0*b_1 + a_2*b_0 + a_2*b_1$	$(a_0 + a_2)*(b_0 + b_1)$
M_{32}:	$a_0*b_0 + a_0*b_2 + a_2*b_0 + a_2*b_2$	$(a_0 + a_2)*(b_0 + b_2)$
M_{33}:	$a_0*b_1 + a_0*b_2 + a_2*b_1 + a_2*b_2$	$(a_0 + a_2)*(b_1 + b_2)$
M_{34}:	$a_0*b_0 + a_0*b_1 + a_0*b_2 + a_2*b_0 + a_2*b_1 + a_2*b_2$	$(a_0 + a_2)*(b_0 + b_1 + b_2)$
M_{35}:	$a_1*b_0 + a_2*b_0$	$(a_1 + a_2)*(b_0)$
M_{36}:	$a_1*b_1 + a_2*b_1$	$(a_1 + a_2)*(b_1)$
M_{37}:	$a_1*b_2 + a_2*b_2$	$(a_1 + a_2)*(b_2)$
M_{38}:	$a_1*b_0 + a_1*b_1 + a_2*b_0 + a_2*b_1$	$(a_1 + a_2)*(b_0 + b_1)$
M_{39}:	$a_1*b_0 + a_1*b_2 + a_2*b_0 + a_2*b_2$	$(a_1 + a_2)*(b_0 + b_2)$
M_{40}:	$a_1*b_1 + a_1*b_2 + a_2*b_1 + a_2*b_2$	$(a_1 + a_2)*(b_1 + b_2)$
M_{41}:	$a_1*b_0 + a_1*b_1 + a_1*b_2 + a_2*b_0 + a_2*b_1 + a_2*b_2$	$(a_1 + a_2)*(b_0 + b_1 + b_2)$
M_{42}:	$a_0*b_0 + a_1*b_0 + a_2*b_0$	$(a_0 + a_1 + a_2)*(b_0)$
M_{43}:	$a_2*b_1 + a_1*b_1 + a_2*b_1$	$(a_0 + a_1 + a_2)*(b_1)$
M_{44}:	$a_0*b_2 + a_1*b_2 + a_2*b_2$	$(a_0 + a_1 + a_2)*(b_2)$
M_{45}:	$a_0*b_0 + a_1*b_0 + a_2*b_0 + a_0*b_1 + a_1*b_1 + a_2*b_1$	$(a_0 + a_1 + a_2)*(b_0 + b_1)$

M_{46}:	$a_0*b_0 + a_1*b_0 + a_2*b_0 + a_0*b_2 + a_1*b_2 + a_2*b_2$	$(a_0 + a_1 + a_2) * (b_0 + b_2)$
M_{47}:	$a_0*b_1 + a_1*b_1 + a_2*b_1 + a_0*b_2 + a_1*b_2 + a_2*b_2$	$(a_0 + a_1 + a_2) * (b_0 + b_2)$
M_{48}:	$a_0*b_0 + a_0*b_1 + a_0*b_2 + a_1*b_0 + a_1*b_1 + a_1*b_2 + a_2*b_0 + a_2*b_1 + a_2*b_2$	$(a_0 + a_1 + a_2) * (b_0 + b_1 + b_2)$

EK-2 Fonksiyon Şeması



Main(): Kombinasyon_olustur() ve kombinasyon_birlestir() fonksiyonlarını çağırarak işlemlerden önce M[] ve Toplam[] dizilerini hazır hale getirir. Daha sonra M_asil_atama() fonksiyonunu çağırır ve işlemleri başlatır.

Kombinasyon_olustur(): Kombinasyon_a() ve kombinasyon(b) fonksiyonları ile iki sayının katsayı kombinasyonlarını oluşturur.

Kombinasyon_a(): a(x) polinomunun katsayılarının bütün kombinasyonlarını bulur.

Kombinasyon_b(): b(x) polinomunun katsayılarının bütün kombinasyonlarını bulur.

Kombinasyon_birlestir(): Kombinasyon_olustur() fonksiyonundan gelen bütün olası kombinasyonları birlestirir.

M_asil_atama(): M_Asil[] dizilerini belirleyerek Carpma_Hesaplama() fonksiyonuna gönderir. Her yeni M_Asil[] dizisi oluşturulmadan önce sifirlama() fonksiyonuyla bütün ortak değişkenler sıfırlanır.

Sifirlama(): Bütün uygulamada kullanılmak üzere oluşturulmuş Global sınıfının üyelerini sıfırlamak için kullanılır.

Carpma_Hesaplama(): Kendisine gelen M_Asil[] dizisinin elemanlarını kombinasyon_5_olustur() fonksiyonuna gönderir ve oradan gelen sadece toplama içeren katsayı denklemlerinin çıkarma içeren olasılıklarını da bulmak için arti_eksi() fonksiyonuna gönderir.

Kombinasyon_5_olustur(): Kombinasyon_5() alt fonksiyonun yardımıyla, M_Asil[] dizisinin elemanlarından oluşan yalnızca çıkarma içeren olası katsayı denklemlerini oluşturur.

Arti_eksi(): Yalnızca toplama işlemi içeren katsayı denklemlerinin çıkarma işlemi de içeren bütün olasılıklarını bulur ve m_cevir() fonksiyonuna gönderir.

M_cevir(): Katsayı denklemlerindeki M çarpımlarının yerine gereken sembolik ifadeleri koyar.

Denklem_Sonuc(): Sembolik ifadeler üstünde sadeleştirme işlemlerini gerçekleştirir.

Karsilastir_ve_bitir(): Denklem_sonuc() fonksiyonundan gelen denklem sonuçları ile çarpım katsayılarını karşılaştırır ve aynı olanları kayıt eder.