www.dergipark.gov.tr
ISSN:2148-3736
El-Cezerî Fen ve Mühendislik Dergisi
Cilt: 4, No: 3, 2017 (429-435)

El-Cezerî Journal of Science and Engineering
Vol: 4, No: 3, 2017 (429-435)

ECJSE

Makale / Research Paper

Pascal Üçgeni, Kombinasyon ve Tümevarım Kullanarak Fibonacci Dizisinin N. Elemanını Bulma

Faruk BULUT

İstanbul Rumeli Üniversitesi, Müh. ve Mimarlık Fakültesi, Bilgisayar Müh. Bölümü, İstanbul /TÜRKİYE, faruk.bulut@rumeli.edu.tr

Received/Geliş: 31.05.2017 **Revised/Düzeltme:** 16.08.2017 **Accepted/Kabul:** 20.08.2017

Özet: Bilindiği üzere Fibonacci dizisi bilişim teknolojileri dâhil birçok mühendislik alanında kullanılmaktadır. Fibonacci dizisinin *n*. elemanını bulabilmek için (*n*-1). ve (*n*-2). elemanlarının da hesaplanması gerekir. Bu işlem bilinmeyen her bir elemanın hesaplanması işlemi özyinelemeli olarak 1. ve 2. elemana kadar gider. Bu çalışmada Paskal üçgeninden faydalanılarak Fibonacci dizisinin *n*. elemanını doğrudan bulabilen bir formül önerilmiştir. Bilindiği üzere Paskal üçgenine sol alttan sağ yukarı doğru diagonal düzlemdeki tüm elemanlar toplandığında Fibonacci dizisinin elemanları sırayla hesaplanabilmektedir. Bu düzlemde gizli olarak bulunan örüntü, matematikteki Kombinasyon, Tümevarım ve Fonksiyon konuları ile modellenerek yeni bir formül haline dönüştürülmüştür. Fibonacci serisindeki elemanları bulmak için özyinelemeli ve dinamik programlama yöntemleri ile yapılan hesaplamalara göre daha az zaman ve alan karmaşıklığı ile benzer sonuçlar bulunmuştur.

Anahtar kelimeler: Fibonacci dizisi, özyineleme, formül.

Calculating Nth Element of Fibonacci Sequence using Pascal Triangle, Combination and Sigma Symbol

Abstract: As is known, Fibonacci sequence is used in many engineering fields including information technology. It is an obligation to calculate the (*n*-1)th and (*n*-2)th elements in the Fibonacci Sequence in order to find the (*n*)th element. These calculations recursively go to the 1st and 2nd elements. In this study, the Pascal triangle is used to determine the Fibonacci sequence. A formula has been proposed that can find the required Fibonacci element directly. It is known that the elements of the Fibonacci sequence can be calculated sequentially when all the elements in the diagonal plane are collected from left to right in the Pascal triangles. The hidden pattern in this triangle is transformed into a new formula by modeling with Combination, sigma symbol and Functions in mathematics. To find the Fibonacci series, time and space complexity is reduced to the minimum according to calculations made by recursive and dynamic programming.

Keywords: Fibonacci sequence, recursion, formula.

1. Giriş

13. yüzyılda yaşayan İtalyan bilim adamı Leonardo Fibonacci tarafından bulunan ve Liber Abaci isimli meşhur kitabında yer verdiği Fibonacci dizisi, altın oran dahil olmak üzere bir çok alanda kullanılmaktadır [1]. Bu sayı dizisi, tavşanların üremesiyle ilgili problemin hesaplanması esnasında ortaya konulmuştur. Esasen Fibonacci dizisinin 6. yüzyılda Hintli matematikçiler tarafından bulunduğu da söylenmektedir [2]. Fibonacci sayı dizisi matematiksel olarak şu şekilde tanımlanır:

Bu makaleye atıf yapmak için
Bulut, F., *"Pascal Üçgeni, Kombinasyon ve Tümevarım Kullanarak Fibonacci Dizisinin N. Elemanını Bulma"* El-Cezerî Fen ve Mühendislik Dergisi 2017, 4(3); 429-435.

How to cite this article
Bulut, F., "*Calculating Nth Element of Fibonacci Sequence using Pascal Triangle, Combination and Sigma Symbol*" El-Cezerî Journal of Science and Engineering, 2017, 4(3); 429-435.

$$fib(n) = \begin{cases} 0 & , \quad n = 0 \\ 1 & , \quad n = 1 \\ fib(n-1) + fib(n-2) & , \quad n > 1 \end{cases} \tag{1}$$

Fibonacci dizisinde $fib(n)$ fonksiyonu ile *n*. elemanın değeri bulunmak istendiğinde dizideki kendinden bir önceki ve iki önceki elemanlar toplanır. Dizideki ilk ve ikinci eleman 1 değerlikli olmak koşuluyla tüm elemanlar bu kurala göre hesaplanabilir. Ayrıca n. eleman F_n şeklinde ifade edildiğinde, F_0=0 ve F_1=1 olmak kaydıyla $F_n = F_{n-1} + F_{n-2}$ şeklinde de matematiksel olarak özyinelemeli bir şekilde gösterilebilir. Bu dizinin ilk birkaç elemanı şu şekildedir: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, … Görüldüğü üzere bu dizide *n*. elemana ulaşabilmek için geriye doğru yaklaşık olarak *n* işlem yapılması gerekmektedir. Bu çalışmada ortaya konan yaklaşım ile n işleme gerek kalmadan önerilen formül ile *n*. eleman rahatlıkla bulunabilmektedir.

Fibonacci sayı dizisi biyoloji [3] ve ekonomi dâhil [4] birçok bilim alanında kullanıldığı gibi bilgisayar bilimlerinde de yaygın olarak kullanılmaktadır. Örneğin Fibonacci Arama tekniği (*Fibonacci Search Method*), tek boyutlu ve optimize edilmiş bir arama yöntemidir [5]. Ayrıca Fibonacci Heap, bir veri yapısı modelidir [6]. Diğer bir taraftan Graph teorilerinde network topolojisi olarak paralel ve dağıtık sistemleri birbirine bağlamak için kullanılan Fibonacci kübü vardır [7].

Matematik ve bilgisayar alanında, Fibonacci kodlama tekniği ile (*Fibonacci Coding*) ile pozitif tamsayılar ikili sayı sistemine çevrilebilmektedir [8]. Bilindiği üzere her bir pozitif tam sayı bir veya daha fazla birbirinden farklı Fibonacci sayısının toplamı olarak yazılabilir. Ayrıca bu teknikte kullanılan hiçbir Fibonacci sayısı peş peşe değildir. Kısaca anlatılan ve Fibonacci Kodlama tekniğine dayana bu yönteme Zeckendorf Teoremi denir [9].

Ayrıca Öklid Algoritmasının çalışma zamanı hesaplamasında iki tamsayının en büyük ortak bölenini (EBOB) bulurken Fibonacci sayı dizisinin son iki elemanı en kötü durumu ifade etmektedir [10].

Ayrıca Fibonacci sayı dizisi, Sözderastsal (rastgele) Sayı Üreteci (*Pseudorandom Number Generator*, PRNG) tekniğinde kullanılmaktadır. Bu teknikte elemanlar arasında kolaylıkla örüntü veya ilişki kurulamayacak bir sayı dizisi türetme yaklaşımı vardır [11].

Fibonacci sayı dizisinin kullanıldığı başka bir alan ise Planlama Pokeri (Planning Poker, Scrum Poker) oyunudur. Planlama Pokeri, yazılım geliştirme sürecinde faydalanılan bir hedef belirleme oyunudur [12].

Çok fazlı birleştirmeli arama yönteminde (*Polyphase Merge Sort*) de Fibonacci sayı dizisinden yararlanılmıştır. Bu yöntemde, sıralı olmayan bir sayı dizisinde sayılar her iterasyonda özyinelemeli olarak ikili gruplara ayrılırken grup uzunlukları ardışık Fibonacci elemanları kadar seçilir. Bu sayede gruplar arası oran altın orana yakın oluşturulmuş olur [13].

Programlama dilleri yardımıyla Fibonacci dizisinin elemanlarını bulmak oldukça kolaydır. Özyinelemeli (*recursion*) ve dinamik programlama (*dynamic programming*) yöntemleriyle bu dizinin elemanları bulunabilir. Fibonacci dizisinin özyinelemeli olarak bulan C/C++ parçacığı şu şekildedir:

```
int fibonacci(int n)
{
```

```
    if(n == 0)
        return 0;
    else if(n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
```

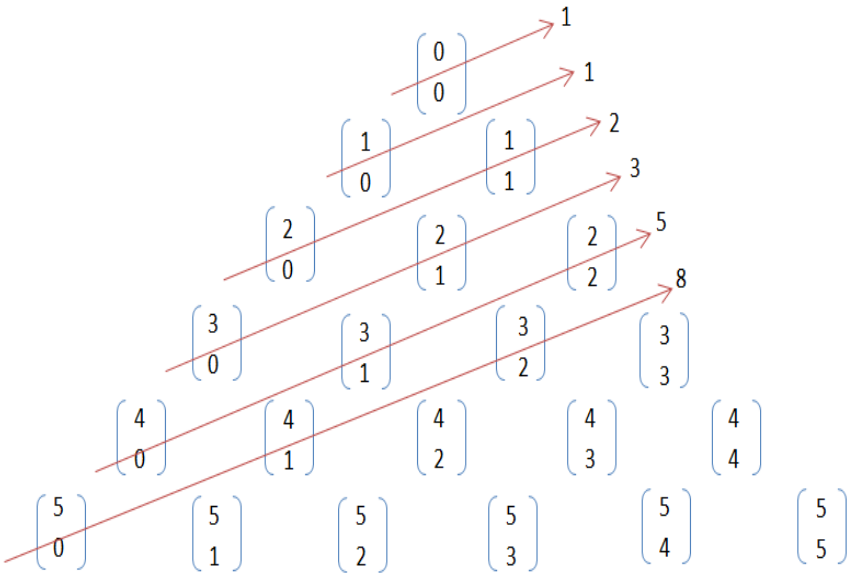
Dinamik programlamada kullanılan aşağıdan yukarı (*bottom-up*) yöntemli ile de bu sayı dizisi bulunabilmektedir. Esasen bu teknik burada lineer bir yönteme dönüşmektedir ve şu şekilde bulunabilir:

```
int fibonacci(int n)
{
    int f[n+1];
    f[1] = f[2] = 1;
    for (int i = 3; i <= n; i++)
        f[i] = f[i-1] + f[i-2];
    return f[n];
}
```

Makalenin geriye kalan kısmında üç bölüm daha vardır. Giriş kısmından sonraki ikinci bölümde önerilen yöntemin matematiksel açıklamasına, üçüncü bölümde karmaşıklık analizinin yapılmasına ve kaynakçadan önceki son bölümde de değerlendirmelere yer verilmiştir.

2. Yöntem

Şekil 1'de de gösterildiği gibi Pascal üçgeninde sol yandan sağ yukarı doğru diyagonal (çapraz) düzendeki her bir eleman toplandığında Fibonacci serisinin elemanları sırayla bulunabilmektedir. Bilindiği üzere Paskal üçgeni, kombinasyon dizilimleri ile oluşturulmaktadır. Yaygın olarak bilinen bu iki bilgiden yola çıkarak bu çapraz dizilimde gizli bulunan bir örüntü tespit edilmiştir.



Şekil 1. Paskal üçgeni

Bu çalışmada, var olan bu örüntüyü bir formül ile ifade etmeye ve daha az zaman karmaşıklığı (*time complexity*) ve hesaplama zamanı (*computational time*) ile Fibonacci dizisinin *n*. elemanını bulmaya çalıştık. Bu sayede Fibonacci serisinin *n*. elemanını bulabilmek için (*n*-1). ve (*n*-2). elemanından başlayarak ilk elemana kadar hesaplama yapmaya gerek kalmamıştır.

Binomsal bir terim olan $(x \pm y)^n$'in açılımındaki kat sayıları bulabilmek için Pascal Üçgeni ile oluşturulan katsayılara ihtiyaç vardır. Pascal üçgeninin oluşturulmasında ve çalışmamızda kullanılan bazı kavramsal ifadeler ve yöntemler kısaca aşağıda anlatılmıştır.

2.1. Kombinasyon

r, *n* ∈ N ve $r \leq n$ olmak üzere, *n* elemanlı bir A kümesinin *r* elemanlı alt kümelerinin her birine A kümesinin *r*'li kombinasyonu denir. .A kümesinin *r*'li kombinasyonları şu şekilde formülize edilir:

$$C(n,r) = \binom{n}{r} = \binom{n}{n-r} = \frac{P(n,r)}{r!} = \frac{n!}{r!\,(n-r)!} \tag{2}$$

2.2. Sigma Sembolü

$\sum$, Yunan alfabesinin 18. Harfidir ve sigma olarak okunmaktadır. *f*: Z→R, *f*(*k*) = a_k ve $r \leq n$ olacak şekilde bir dizi şu şekilde tanımlanmaktadır:

$$\sum_{k=r}^{n} a_k = a_r + a_{r+1} + a_{r+2} + \cdots + a_n$$

Yukarıdaki ifadesinde, $k \in Z$'ye indis ya da değişken, $r \in Z$ ye alt sınır, $n \in Z$ ye de üst sınır denir ve *f* fonksiyonu da *r*'den *n*'ye kadar olan (*n*-*r*+1) tane terimin toplamıdır.

2.3. Örüntüler

Şekil 1'de de görüldüğü üzere diyagonal olarak sol alttan sağ üste doğru çizilen okun üzerine gelen kombinasyonlar toplandığında Fibonacci sayı dizisinin elemanları bulunmuş olur. Bu durumu Fibonacci dizisinin 4., 5., ve 6. elemanları örnek olarak aşağıda gösterilmektedir:

$$fib(4) = \binom{3}{0} + \binom{2}{1} = 1 + 2 = 3$$
$$fib(5) = \binom{4}{0} + \binom{3}{1} + \binom{2}{2} = 1 + 3 + 1 = 5$$
$$fib(6) = \binom{5}{0} + \binom{4}{1} + \binom{3}{2} = 1 + 4 + 3 = 8$$

Yukarıda verilen örneklerden iki farklı örüntü olduğu örüntünün olduğu söylenebilir. Görüldüğü gibi birinci örüntü, Fibonacci dizisinin çift indis numaralı elemanları parantezli kombinasyon ile ifade edilince parantezin üst kısımları $(n-1)'$den başlayıp birer birer azalarak $\frac{n}{2}$'ye kadar gitmektedir.

Alt kısımları da 0'dan başlayıp birer birer artarak $(\frac{n}{2}-1)$'e kadar sıralı bir şekilde artmaktadır. Örnek olarak 8. indisteki elemanı hesaplayabilmek için ilk 7 elemanın hesaplanması gerekmektedir. Fakat burada kombinasyon ve sigma sembolü ile 8. indisteki eleman şu şekilde ifade edilebilir:

$$fib(8) = \sum_{k=1}^{4} \binom{8-k}{k-1} = \binom{7}{0} + \binom{6}{1} + \binom{5}{2} + \binom{4}{3} = 1 + 6 + 10 + 4 = 21$$

İkinci örüntüde ise tek indis numaralı elemanlarda kombinasyon ifadesinin üst kısımlar $(n-1)$'den başlayıp birer birer azalarak $(\frac{n-1}{2})$'ye kadar gitmektedir. Kombinasyon ifadesinin alt kısımları ise 0'dan başlayıp birer birer artarak $(\frac{n-1}{2})$'ye kadar ardışık olarak artmaktadır. Örnek olarak 9. indisteki elemana ulaşabilmek için aşağıdaki sigma ve kombinasyonlu ifade yazılabilir.

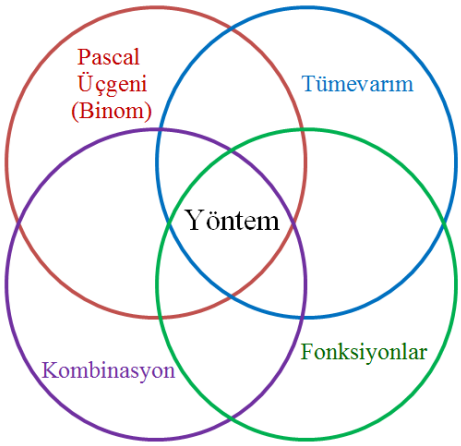
$$fib(9) = \sum_{k=1}^{5} \binom{9-k}{k-1} = \binom{8}{0} + \binom{7}{1} + \binom{6}{2} + \binom{5}{3} + \binom{4}{4} = 34$$

2.4. Fibonacci formülü

Tek ve çift indis numaralı Fibonacci sayıları için farklı formüller tek bir parçalı fonksiyon ile şu şekilde ifade edilebilir:

$$fib(n) = \begin{cases} \sum_{k=1}^{\frac{n}{2}} \binom{n-k}{k-1} & , \frac{n}{2} \in Z \\ \sum_{k=1}^{\frac{n+1}{2}} \binom{n-k}{k-1} & , \frac{n}{2} \notin Z \end{cases} \quad (3)$$

Yukarıdaki (3) numaralı formülü kullanılarak Fibonacci dizisinin *n*. elemanı, kendinden önce gelen elemanların bilinmesine gerek kalmadan kolayca hesaplanabilir. Bu sonuç ile matematik alanındaki Pascal Üçgeni, Binom açılımı, Kombinasyon, Tümevarım ve Fonksiyon konuları ile bir yöntem önerilmektedir. Bir araya getirilen yöntemler bir şema ile şu şekilde gösterilebilir:



Şekil 2. Projede kullanılan yöntemler

3. Karmaşıklık Analizli

Önerilen yöntem zaman karmaşıklığı (*time complexity*) ve alan karmaşıklığı (*space complexity*) açısından avantajlıdır denilebilir. Kullanılan özyinelemeli, Yukarıdan aşağı dinamik programlama, aşağıdan yukarı dinamik programlama ve önerilen yöntem arasındaki karmaşıklık karşılaştırması aşağıdaki tabloda verilmektedir.

Tablo 1. Zaman ve alan Karmaşıklıklarının karşılaştırılması

	Yöntem	**Alan Karmaşıklığı**	**Zaman Karmaşıklığı**
1	Özyinelemeli	$O(2^N + N)$	$O(2^N)$
2	Dinamik Programlama (Yukarıdan aşağı)	$O(N)$	$O(N^2)$
3	Dinamik Programlama (Aşağıdan yukarı)	$O(N)$	$O(N^2)$
4	Önerilen yöntem	$O(N)$	$O(N)$

Fakat burada şunu belirtmek isteriz ki ilk etapta 3 numaralı eşitlikte verilen fonksiyon yalın olarak incelendiğinde karmaşıklık $O(N/2)$ olduğu ve katsayılar ihmal edildiğinde O(N)'e dönüştüğü şu şekilde görülebilir:

$$fib(n) = \sum_{k=1}^{n/2} \binom{n-k}{k-1} = \binom{n}{0} + \binom{n-1}{1} + \binom{n-2}{2} + \cdots + \binom{\frac{n}{2}}{\frac{n}{2}-1}$$

$$= \frac{n!}{0!\,n!} + \frac{(n-1)!}{1!\,(n-2)!} + \cdots + \frac{(\frac{n}{2})!}{(\frac{n}{2}-1)!\,1!}$$

Ayrıca denklem içerisinde Kombinasyon değerlerinin hesaplama maliyeti eklenmemiştir. Örneğin C(N,k) kombinasyonu için faktöriyel hesaplanması yaklaşık $O(N)$ maliyetli olacaktır ve çalışma zamanını arttıracaktır. Ayrıca Fib(N) = $O(N/2*N)$ = $O(N^2)$ şekline dönüşür. N sayısı çok büyük bir tam sayı değeri alması durumunda ise faktöriyel değerinin hesaplanması oldukça zaman alacaktır. Örneğin 100! Sayısının değeri NaN çıkacaktır. Bu durumda Kombinasyon değeri dinamik programlama tekniği ile hesaplanıp bir veri yapısında tutulur ise tekrarlı işlemler yapılmadığından dolayı hesaplamalar çok daha kısa sürecektir. Dinamik dizi aynı zamanda Paskal üçgeninin kendisi de olabilir. Kısaca toplam karmaşıklık Fib(N) = C(N) + O(N/2) = O(N) olur. Bu durumda alan karmaşıklığı ise $O(N)$ şeklinde ifade edilebilir.

Bilinen en iyi Fibonacci çalışmasının zaman karmaşıklığı $O(\log N)$ şeklindedir [14]. 2×2'lik birim matris ve Matris Zincir Çarpımı (*Matrix Chain Product*) yöntemi kullanılarak bir matrisin N. kuvveti alınarak hesaplanmıştır ve tablo içerisinde verilerek çalışma zamanı $O(\log N)$ olacak şekilde indirgenmiştir.

Görüldüğü üzere ilk 3 alan karmaşıklığı eleman sayısı ile aynıdır çünkü her hangi bir sayının bulunabilmesi için sayı dizisindeki ilk elemana kadar hesaplama yapılması gerekmektedir. Bu hesaplamaların tek tek hafızada kaydedilmesi gerektiği için alan karmaşıklığı $O(N)$'dir. Önerilen yöntemde herhangi bir elemanın bulunması için daha önceki elemanların bulunması ve bir hafıza biriminde kaydedilmiş olması gerekmemektedir. Zaman karmaşıklığının O(N) olarak ifade edilmiş olması önerilen yöntemin lineer bir yapıda olduğunu göstermektedir.

Bilindiği üzere karmaşıklık analizinde bölüm ve çarpımda bulunan sabit sayılar silinmektedir ve parametreler sonsuza giderken karekteristik açıdan incelendiğinde sabit sayıların bir önemi kalmamaktadır. Bu yüzden önerilen yöntem ile var olan yöntemlerin karmaşıklık ifadeleri sadeleştirilerek yazılmıştır. Fakat deneysel uygulamalarda hesaplama zamanı açısından diğer bilinen yöntemlere göre daha hızlı sonuca ulaşıldığı görülmüştür.

5. Sonuç ve Değerledirmeler

Birçok alana ilham kaynağı olan Fibonacci serisinin *n*. elemanını hesaplayabilmek için serideki (*n*-1) elemanın o ana dek tek tek hesaplanmış olması gerekmektedir. Normalde ilk (*n*-1) eleman bilişim dünyasında özyinelemeli yöntemle veya dinamik programlama ile bulunmaktadır. Bu çalışmada Paskal üçgeninde gizli bulunan örüntüden yararlanarak binom, tümevarım ve kombinasyon kullanarak matematiksel bir fonksiyon önerilmiştir. Fibonacci serisinin herhangi bir elemanını bulmak için kullanılan ve klasik bir yaklaşım olan geriye dönük tüm elemanların hesaplanması yöntemine karşı zaman ve alan karmaşıklığı daha düşük bir yöntem önerilmiştir. Bu yöntem sayesinde daha az zaman ve alan karmaşıklığı ile aynı sonuçlar daha kısa hesaplama zamanı ile bulunabilmektedir.

References

[1] Hsu C.H., Hung-Son D., "The application of Fibonacci sequence and Taguchi method for investigating the design parameters on spiral micro-channel." Applied System Innovation (ICASI), 2016 International Conference on IEEE, 2016.
[2] Plofker K., Hannah J., "Mathematics in India." Aestimatio: Critical Reviews in the History of Science 7, 45-53, 2015.
[3] Goel, N.S., Richter N., Stochastic models in biology, Elsevier, USA, 2016.
[4] Brasch TV. Byström J., Lystad L.P., "Optimal Control and the Fibonacci Sequence", Journal of Optimization Theory and Applications, 154 (3): 857–78, doi:10.1007/s10957-012-0061-2, 2012.
[5] Orozco-Henao, C., "Active distribution network fault location methodology: A minimum fault reactance and Fibonacci search approach.", International Journal of Electrical Power & Energy Systems 84, 232-241, 2017.
[6] Kaplan H., Tarjan R.E., Zwick U., "Fibonacci heaps revisited." arXiv preprint arXiv:1407.5750, 2014).
[7] Klavžar S., "Structure of Fibonacci cubes: a survey." Journal of Combinatorial Optimization 25(4):505-522, 2013.
[8] Stakhov A.P., Massingue V., Sluchenkova A., "Introduction into Fibonacci coding and cryptography." Osnova, Kharkov, 1999.
[9] Lengyel T., "A counting based proof of the generalized Zeckendorf's theorem." Fibonacci Quarterly 44.4, 324, 2006.
[10] Knuth, D.E., The Art of Computer Programming, 1: Fundamental Algorithms (3rd ed.), Addison–Wesley, p. 343, ISBN 0-201-89683-4, 1997.
[11] DeBellis, R.S., Ronald M.S., Phil C.Y., "Pseudorandom number generator." U.S. Patent No. 6,044,388. 28 Mar. 2000.
[12] Cohn, M., Agile estimating and planning. Pearson Education, ISBN-13: 978-0131479418, 1st Edition, 2005.
[13] Niemann Thomas, Sorting and Searching Algorithms kitabı, Oregon, USA, 2010.
[14] Edson M, Yayenie O., "A New Generalization of Fibonacci Sequence & Extended Binet's Formula." Integers, 9(6), 639-654, 2009.