BEÜ Fen Bilimleri Dergisi
2(2), 148-153, 2013

BEU Journal of Science
2(2), 148-153, 2013

Araştırma Makalesi / Research Article

Skip List Veri Yapısında P Eşik Değerlerinin Rastgele Seviye Oluşturma ve Performansa Etkisi

Mustafa AKSU[*1], Ali KARCI[2], Şaban YILMAZ[1]

[1]*Kahramanmaraş Meslek Yüksek Okulu, Sütçü İmam Üniversitesi, Kahramanmaraş*
[2] *İnönü Üniversitesi,Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Malatya*

Özet
Skip list veri yapısında bağlı listeler kullanılır. Düğümler, birbirine bağlı listeler halinde farklı seviyelere yerleştirilir. Bu seviyeler sayesinde eleman arama, ekleme, silme gibi işlemler kolayca (O(lg N)) yapılabilir. Bununla birlikte, bu veri yapısında seviyelere düğüm ekleme işleminde olması gerekenden yüksek seviyeler üretilebilmektedir. Yüksek seviyeler üretilmesi bu veri yapısının performansını olumsuz etkilemektedir. Bu makalede rastgele seviye üretme problemi ele alınarak farklı "p" eşik değerlerinin (0.1, 0.25, 0.5, 0.75, 0.9 gibi) performansı nasıl etkilediği ele alınıp çözüm önerilmiştir. Skip list veri yapısındaki yüksek seviye üretme problemi optimum p eşik değeri bulunarak çözülmüştür. Böylece, Skip list veri yapısı için p eşik değerlerine bağlı olarak ideal seviyeler oluşturulmuştur.

Anahtar Kelimeler: Skip list, rastgele seviye, p eşik değerleri, ideal seviye, optimizasyon.

Effects of P Threshold Values in Creation of Random Level and to the Performance of Skip List Data Structure

Abstract
In Skip list data structure, linked lists are used. Nodes are allocated to different levels as linked list. By the help of these levels operations like item search, insertion, or removal can be performed easily (O(lg N)). However, in this data structure in the operation of adding nodes to levels, higher levels can be produced than needed. Creating higher levels affects the performance of this data structure negatively. In this article how randomly creation of levels and different "p" thresholds (0.1, 0.25, 0.5, 0.75, 0.9) affect the performance is studied and solutions are proposed. The problem of creating higher levels in Skip list data structure is solved by introducing optimum p thresholds. Hence, ideal p threshold levels are created for Skip list data structure.

Keywords: Skip list, random level, p thresholds, ideal level, optimization.

1. Giriş

Veri yapıları ve algoritmalar bilgisayar bilimlerinin temellerini oluşturur. Birçok problemin çözümünde değişik veri yapıları ve algoritmalar kullanılır. Bu veri yapıları ihtiyaca göre statik veya dinamik olabilir. Bazen varolan bir veri yapısı ihtiyaca cevap vermez veya yetersiz kalır. İşlem, zaman, donanım gibi kısıtlamalar olabilir. Bütün bunlar göz önüne alındığında yeni veri yapıları veya algoritmalar ortaya çıkmaya devam edecektir. Bunlara örnek olarak bağlı listeler, B-ağaçları, dinamik tablolar, ikili yığınlar, Fibonacci yığınları, binom yığınları, Splay veri yapıları, ayrık küme veri yapıları vb. verilebilir. Algoritmalar ise, QuickSort, MergeSort, HeapSort, B-ağaçları algoritmaları, dinamik tablo algoritmaları, vb. verilebilir. Skip List veri yapısı da bağlı listelerde arama işleminin doğrusal aramaya dönmesi sonucunda ortaya konulmuş bir veri yapısıdır.

Bağlı liste veri yapısında; arama, ekleme, silme gibi işlemler doğrusal olup eğer liste N tane eleman içeriyorsa, bu işlemler için zaman karmaşıklığı O(N) olmaktadır. Bu problemi çözmek için

[*] Sorumlu yazar: *m.aksu@ksu.edu.tr*

PUGH tarafından Skip list veri yapısı ortaya konulmuştur[1,2,3] ve yapı olarak ray yapısına benzemektedir. Bu yapıyı oluşturmak için üst-üste bağlı listeler oluşturulur; en alt listede N tane düğüm varsa, bir üst seviyede N/2 tane düğüm olur, ondan sonra gelen bağlı listede N/4 tane düğüm olur ve böyle devam eder[2]. En alt listede bütün düğümler bulunur ve üst listelere doğru düğüm sayısı azaltılmış kopyalar bulunur. Bundan dolayı Skip list veri yapısındaki her bir liste kendi üzerindeki listeleri kapsar. N elemanlı bir Skip list veri yapısında arama, ekleme, silme işlemleri için zaman karmaşıklığı O(lgN)'dir. Dizilerde ve bağlı listelerde bu işlemler O(N) zaman karmaşıklığına sahipken Skip list veri yapısında O(lgN) olması büyük bir avantajdır [4].

Skip list veri yapısı algoritmalarının, analizi ve iyileştirilmesi adına bir takım çalışmalar [5,6,7,8,9,10,11,12,13] yapılmıştır. Bu yapılan çalışmalar da göstermektedir ki bu veri yapısının bazı algoritmalarında sorunlar vardır. Bu yüzden Skip list veri yapısı stabil değildir.

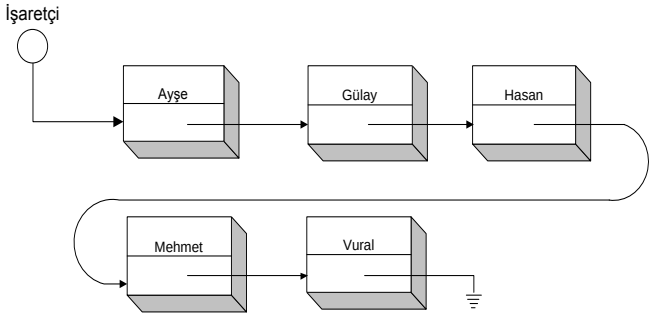
Bizim yaptığımız çalışmada, Skip list veri yapısına düğüm eklerken olasılıksal olarak oluşturulan seviye (level) değerlerinin, p eşik değerlerinden nasıl etkilendiği üzerinde durulmuştur. Skip list veri yapısı üzerinde çalışma yapan bir çok araştırmacı p eşik değerini ~0.5 almıştır. Bizim çalışmamızda ise p eşik değerinin ~0.25 alındığında bu veri yapısının performasının en iyi düzeye çıktığı uygulamalı olarak gösterilmiştir. Böylece, optimum p eşik değeri bulunarak Skip list veri yapısının (arama, ekleme, silme) performansı artırılmıştır.

2. Materyal ve Metot

2.1. Bağlı Listeler

Statik veri yapılarına örnek olarak diziler, statik yığıtlar, statik kuyruklar verildi. Bağlı listelerde, veriler belli bir sırada yerleştirilirler ve o ana kadar kaç tane veri geldiyse, bağlı listenin boyutu odur. Programın icrası sırasında bağlı listenin boyutu değişebilmektedir. Bundan dolayı bağlı listelere dinamik veri yapıları denir. Dinamik veri yapıları hafızanın etkin bir şekilde kullanılması istendiği durumlarda mutlaka kullanılmalıdırlar. Bağlı listelerin tanımlanması için veri tiplerinden işaretçi (referans) kullanılır. İşaretçi bir veri tipi olup hafıza hücrelerinin adresini tutan değişkenlerdir. Bu değişkenler başka bir değişkenin bulunduğu hafıza adresini tuttuğundan dolayı bu yolla değişkenler birbirine bağlanır. Bir zincirin halkaları gibi başlangıç değişkenden yola çıkarak diğer değişkenlerin değerleri elde edilebilir.

Bağlı listeler, düğüm adı verilen veri parçacıklarının bir araya getirilip birbirlerine bağlanmasıyla oluşturulan bir veri yapısıdır. Bağlı listelere erişim için bir başlangıç düğümüne (işaretçi) ihtiyaç vardır. Her düğüm veri(ler) ve bir sonraki düğüme bağlantı (sonraki) bileşenlerinden oluşur. Ayrıca bağlı listenin bittiğini gösteren bir sonlandırıcıya ihtiyaç vardır (Şekil 1).



Şekil 1. Tek yönlü sıralı bağlı liste ve düğüm yapısı

Bu sıralı yapı herhangi bir pozisyona düğüm ekleme, silme ve düğüm arama gibi işlemlerde etkin bir şekilde kullanılır. Bağlı listeler, çift yönlü bağlı listeler, çevrimsel bağlı listeler, Skip list gibi farklı şekillerde oluşturulup, farklı amaçlar için kullanılabilir. Bilgisayar bilimlerinde yaygın olarak kullanılan yığıt, kuyruk, Skip list, skip graph gibi veri yapıları da bağlı listelerden oluşur.

2.2. Skip list veri yapısı

Skip list veri yapısında bağlı listeler kullanılır ve bağlı liste elemanları sıralı olarak değişik seviyelere yerleştirilerek (Şekil 2) arama, ekleme, silme işlemlerinde kolaylık sağlanması amaçlanır.

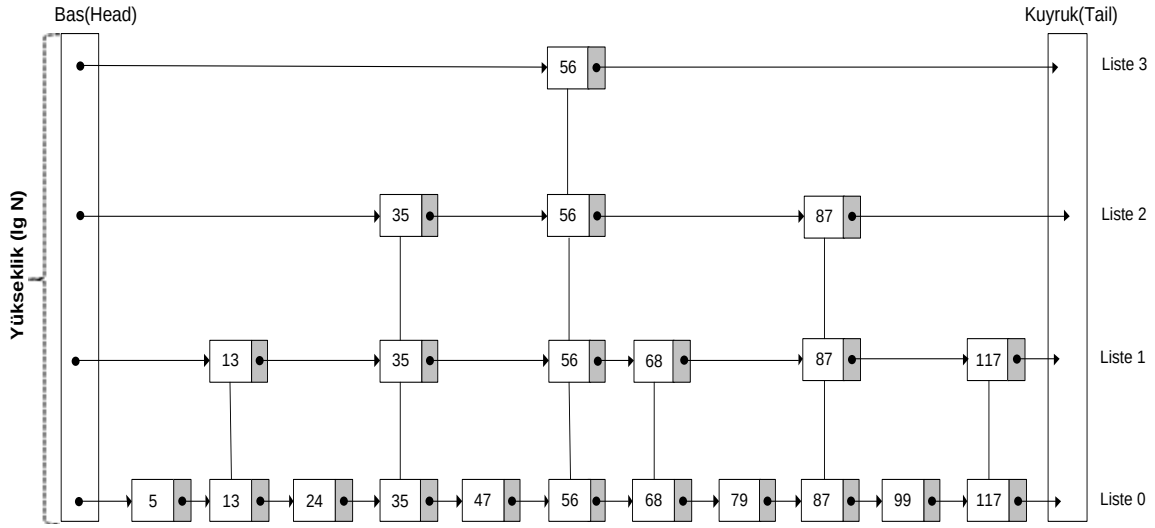
Skip list veri yapısı bağlı listelerden oluştuğu için, bağlı liste veri yapısında olduğu gibi listenin her bir elemanı düğüm olarak kabul edilir.

2.2.1. Skip listin Oluşturulması

{5, 13, 24, 35, 47, 56, 68, 79, 87, 99, 117} düğümlerinden oluşan bir Skip list Şekil 2' de görülmektedir.

Skip list oluşturulurken ilk olarak tüm düğümler Liste 0'da yer alır ve sol taraftan başlanarak her 2^i. düğüm (i=0,..,MaxLevel(31)) atlanarak üste doğru her seviyeyi temsil eden işaretçiler oluşturulur. $Liste_0 \supseteq Liste_1 \supseteq .....\supseteq Liste_l$. $Liste_0$, Skip list veri yapısında en alt seviyedeki bağlı liste olup tüm düğümleri kapsar. Skip list veri yapısı N tane sıralı düğümden oluşuyorsa $Liste_0$, bu N tane sıralı düğümlerin tamamından oluşur (Şekil 2-Liste 0) [2].

Bağlı listelerde yapılan arama, ekleme, silme gibi işlemler O(N) karmaşıklığında yapılmakta iken Skip list veri yapısında bu karmaşıklık O(lg N)'e azalmıştır.



Şekil 2. Skip list veri yapısı

Skip list veri yapısı ekleme, silme, arama işlemleri için kullanılabilmektedir. Arama algoritmasında aranacak düğüm üst seviyeden başlanarak aşağı seviyelere doğru aranır. Ekleme işleminde önce eklenecek düğüm aranır bulunamamışsa rastgele belirlenen seviyeden itibaren eşleşen konuma yeni değer eklenir, işaretçiler ve listeler güncellenir. İşlem düğümün ekleneceği diğer seviyeler için tekrarlanır. Silme işleminde ise en üst seviyeden başlanarak alt seviyelere doğru arama gerçekleştirilir. Silinecek düğüm bulununca silinir, işaretçiler ve listeler güncellenir. Aynı işlemler düğümün olduğu diğer seviyeler için tekrarlanır.

Skip list veri yapısına düğüm ekleme, silme ve arama ile ilgili daha detaylı bilgi ve algoritmalar PUGH [1,2,3]'un makalelerinde mevcuttur.

Bağlı ve sıralı listeler kullanıldığında Skip list algoritmalarının (arama, ekleme, silme) zaman karmaşıklığı O(lg N) olmakta, bu da diğer algoritmalar ile karşılaştırıldığında önemli bir zaman farkı avantajı oluşturmaktadır.

PUGH'un Skip list veri yapısı için geliştirmiş olduğu algoritmalarda bazı problemler vardır. Bu problemler Skip list işlemlerinde (arama, ekleme, silme) performansı düşürmektedir.

Bu veri yapısında, bir düğüm eklerken eklenecek bu düğüm için seviye oluşturmada kullanılan PUGH'un olasılıksal randomLevel() algoritması yüksek seviye üretmektedir. Farklı p eşik değerleri Skip listin seviyelerini ve performansını etkilemektedir. Bizim çalışmamızda bu sorun ele alınmış ve çözüm üretilmiştir.

2.2.2. Seviye (level) oluşturma

Skip list veri yapısı oluşturulurken, düğümler olasılıksal olarak rastgele belirlenen seviyelere yerleştirilir. PUGH'un **randomLevel()** algoritması (**Algoritma 1**) düğüm eklerken seviye (level) oluşturmak için 1-MaxLevel(31) arası rastgele bir değer üretir [4]. Bu değer (level) üretme şöyle gerçekleşir: lvl başlangıç değeri 1 alınır. Daha sonra yazı tura atma işlemi gibi rastgele 0-1 arası ondalıklı değer üretilir. Bu üretilen değer, **p** eşik değerinden küçük ve lvl<MaxLevel olduğu sürece lvl değeri 1 artırılır. İşlem bu şekilde devam eder. Ne zaman üretilen sayı **p** eşik değerinden büyük olursa yada lvl değeri MaxLevel değerini ulaşmışsa lvl üretilmiş olur. Eklenecek düğüm, üretilen bu lvl değeri esas alınarak yapıya eklenir.

PUGH'un geliştirmiş olduğu ***randomLevel()*** algoritması şöyledir;

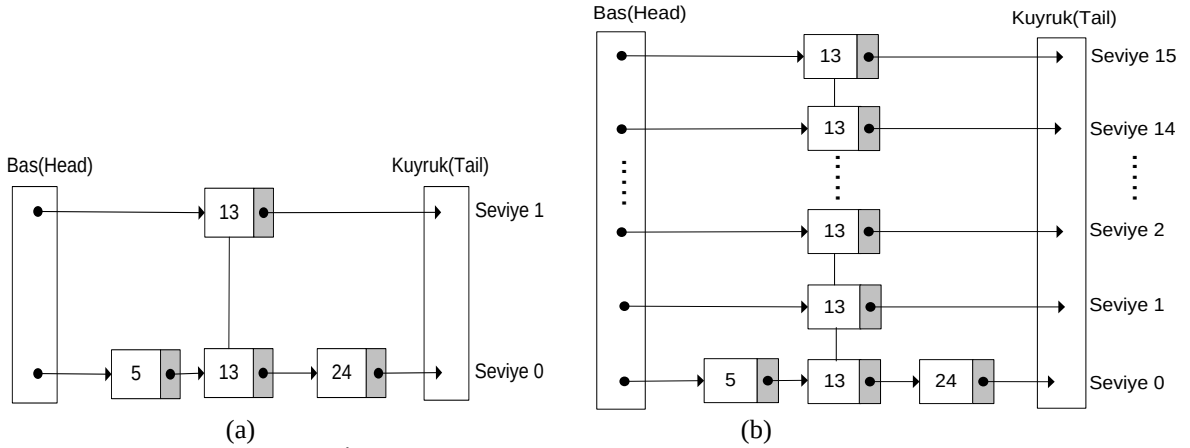
Algoritma 1

```
randomLevel()
  lvl := 1                          {lvl başlangıç değerini 1 al}
   -- random() that returns
   --a random value in [0...1)          { [0..1) aralığında ondalıklı rastgele bir değer üret }
while random()< p and lvl < MaxLevel do { Üretilen değer p eşik değerinden küçük ve … }
    lvl := lvl + 1                      { ... MaxLevel değerine ulaşılmamışsa lvl değerini 1 artır. }
return lvl
```

Olasılıksal olarak seviye oluşturulurken eğer p eşik değeri 1'e yakın olursa Skip list veri yapısının seviyesi (yüksekliği) fazla olacak. Bu da Skip list veri yapısında 2-3 düğüm varken bile seviye (level) olarak yüksek değerler oluşturabilmektedir (Şekil 3.b). Bu durum Skip listin performansını olumsuz etkiler. Halbuki 2 veya 3 düğüm varken ideal seviye level=$\log_2$ (N=3)'den 2 olmalıdır. İdeal seviyeye yakın seviye elde etmek için p eşik değeri 0-0.5 arası olmalıdır.

İdeal Skip listte N sayıda düğüm varsa bu yapı için olması gereken en yüksek seviye düğüm sayısının 2 tabanında logaritması alınarak bulunur. Skip listin o anki olması gereken ideal seviyesi $\boldsymbol{level = \log_2 N}$ olmalıdır. Fakat bu ideal durumu yakalamak düğüm sayısı bilinmeden çok zordur [4].

Şekil 3'te {5,13,24} düğümlerinden oluşan iki ayrı Skip list yapısı görülmektedir. Şekil 3.a'da olması gereken ideal seviye level=$\log_2$ (N=3)'den 2 olmaktadır. Şekil 3.b'de ise istenmeyen (0..31) seviyede gerçekleşen Skip list görülmektedir [4].



Şekil 3. (a) İdeal seviyede Skip list, (b) Gerçekleşebilecek Skip list [4]

3. Bulgular ve Tartışma

Bizim bu uygulamalı çalışmamızda, **p** eşik değeri olarak [0...1) arasında farklı değerler alınıp sonuçlar incelenmiştir. Uygulama standart C programlama dili komutları kullanılarak Dev-C++ derleyici ortamında geliştirilmiştir. Uygulama İntel i3-330M işlemci ve 8 gigabyte belleğe sahip bir bilgisayar üzerinde koşturularak sonuçlar elde edilmiştir.

Geliştirdiğimiz uygulama ile farklı sayıda düğümlerden oluşan Skip list yapıları oluşturarak, farklı p eşik değerleri (0.1, 0.25, 0.5, 0.75, 0.9) için sonuçların nasıl değiştiğini inceledik. Elde ettiğimiz veriler Tablo 1,2,3,4,5'te görülmektedir. Tablolardaki Skip list oluşturulma süreleri milisaniye (ms) cinsindendir.

Tablo 1. P eşik değeri 0.1 için

Düğüm sayısı	1000	5000	10000	20000	30000	60000	100000
Oluşan Seviye ~	3	4	4	5	5	5	5
Skip List Oluşma Süresi ~	0-1 ms	4 ms	10 ms	20 ms	28 ms	69 ms	142 ms

Tablo 2. P eşik değeri 0.25 için

Düğüm sayısı	1000	5000	10000	20000	30000	60000	100000
Oluşan Seviye ~	4	5	6	6	7	8	9
Skip List Oluşma Süresi ~	0 ms	3 ms	7 ms	15 ms	23 ms	59 ms	109 ms

Tablo 3. P eşik değeri 0.5 için

Düğüm sayısı	1000	5000	10000	20000	30000	60000	100000
Oluşan Seviye ~	10	11	12	13	15	16	18
Skip List Oluşma Süresi ~	1 ms	4 ms	10 ms	19 ms	32 ms	69 ms	135 ms

Tablo 4. P eşik değeri 0.75 için

Düğüm sayısı	1000	5000	10000	20000	30000	60000	100000
Oluşan Seviye ~	24	27	32	34	34	35	37
Skip List Oluşma Süresi ~	1-2 ms	5 ms	15 ms	23 ms	37 ms	87 ms	152 ms

Tablo 5. P eşik değeri 0.9 için

Düğüm sayısı	1000	5000	10000	20000	30000	60000	100000
Oluşan Seviye ~	61	77	89	96	107	114	132
Skip List Oluşma Süresi ~	2-3 ms	10 ms	23 ms	39 ms	72 ms	141 ms	274 ms

Tablo 1,2,3,4 ve 5'teki değerler incelendiğinde Skip listin seviyesi (yüksekliği) fazlaysa (Tablo 3 ,Tablo 4 ve Tablo 5) Skip listin oluşturulma süresi de artmaktadır. Ayrıca yüksekliği fazla olan Skip listte düğüm arama, ekleme, silme işlemlerinin performansı da kötü olacaktır. Tablo 1 incelendiğinde çok küçük **p** (0.1) eşik değerleri de Skip list veri yapısının oluşturulma süresini arttırdığı görülmektedir. **P**=0.1 iken seviye olması gerekenden az olduğu için performans olumsuz etkilenmektedir.

Yukarıdaki Tablolara (Tablo 1, 2, 3, 4, 5) dikkat edilirse en ideal p eşik değeri ~0.25'tir. 100000 (yüzbin) düğümlü bir Skip listte Tablo 1 **(p=0.1)** ve Tablo 5 **(p=0.9)** kıyaslandığında, Tablo 1'de oluşan seviye=5 (yükseklik) ve Skip listin oluşma süresi=142 ms iken Tablo 5'de oluşan seviye=132 (yükseklik) ve Skip listin oluşma süresi=274 ms olmaktadır. Tablo 1 ve Tablo 5'teki sonuçlara dikkat edilirse **p=0.1** için Skip list yaklaşık olarak bir bağlı listeye (satıra) **p=0.9** değeri için dikey bir sütuna dönüşmektedir. Yani arama, ekleme, silme gibi işlemler logaritmik yerine doğrusala dönüşmektedir. Tablo 2 incelendiğinde p eşik değeri ~0.25 alınırsa oluşan seviye=9 (yükseklik) ve Skip listin oluşma süresi=109 ms olmaktadır. Yani p eşik değeri ~0.25 alındığında çok yüksek seviyeler üretilmez. Bundan dolayı Skip list oluşturma süresi, arama, ekleme, silme gibi işlemlerin süresi diğerlerine göre daha iyidir. Bu durumda Skip list yapısı düzenli oluşacağından arama, ekleme, silme işlemlerinde zaman kaybı olmaz. *Yani **level*** değeri Skip listin o anki olması gereken ideal seviyesine yakın bir seviyede olur.

4. Sonuç ve Öneriler

Skip list veri yapısı, PUGH tarafından ortaya atılan ve bağlı listelerin kullanıldığı, düğüm arama, ekleme, silme işlemlerinde çok hızlı dinamik bir veri yapısıdır. Skip list veri yapısı, bağlı liste veri yapısındaki arama, ekleme, silme işlemlerinin O(N) olan zaman karmaşıklığını O(lg N) zaman karmaşıklığına indirmektedir. Bundan dolayı önemli bir veri yapısıdır.

Buna karşılık seviye oluşturma (**randomlevel**) ve buna bağlı olarak düğüm ekleme, silme işlemlerinde iyileştirmeye ihtiyaç vardır. Skip list veri yapısından iyi sonuçlar elde etmek için optimum seviye üretilmesi çok önemlidir. Bu da p eşik değerine bağlıdır.

Bizim çalışmamızda PUGH'un **randomlevel()** algoritması (**Algoritma 1**) ele alınmıştır. Bu algoritmada olasılıksal olarak **p** eşik değerlerine bağlı seviyeler üretilmektedir. Bu üretilen seviyelere düğümler eklenmektedir. Bu eklenen düğümlerin seviyeleri çok yüksek (p eşik değeri 0.5, 0.75, 0.9 gibi) ya da çok düşük (p eşik değeri 0.1 gibi) ise Skip list veri yapısının performansı olumsuz etkilenmektedir. Bizim bu çalışmamızda net bir ifade içermeyen $p<0.5$ alındığında iyi sonuçlar elde edilir yaklaşımının $p<0.2$ ve $p>0.3$ için doğru olmadığı; en ideal **p** eşik değerinin ~0.25 olacağı uygulamalı olarak gerçekleştirilmiş ve sonuçları Tablo 1, 2, 3, 4 ve 5'te sunulmuştur.

Yine bu çalışmamızda, Skip list veri yapısında **p** eşik değeri ~0.25 olarak alınarak tutarsız seviye üretme sorunu da çözülmüştür. Böylece Skip list oluşturulurken veya düğüm eklenirken çok yüksek seviyelerin oluşturulması sorunu ortadan kalkmıştır. Bu çalışmayla, **p** eşik değerinin ~0.25 alınması durumunda Skip list veri yapısının seviyesinin (level) yani yüksekliğinin ideal hale geleceği gösterilmiştir. Bunun için farklı sayılarda düğümlerden oluşturulan Skip list veri yapıları üzerinde uygulama yapılıp sonuçlar Tablo 1,2,3,4,5'te gösterilmiştir.

Kaynaklar

1. Pugh W. 1990. Skip Lists: A Probabilistic Alternative to Balanced Trees, Communications of the ACM. 33(6): 668–676.
2. Pugh W. 1989. A Skip List Cookbook, Dept. of Computer Science, University of Maryland, College Park, UMIACS–TR–89–72.1.
3. Pugh W. 1989. Concurrent Maintenance of Skip Lists, Dept. of Computer Science, University of Maryland, College Park, TR–2222
4. Aksu M., Karcı A., Yılmaz Ş. 2013. Skip List veri yapısında Seviye Optimizasyonu, ISITIES2013 (1st Internatıonal Symposıum on Innovatıve Technologıes ın Engıneerıng and Scıence), pp389-396.
5. Herlihy M., Lev Y., Luchangco V., Shavit N. 2007. A Simple Optimistic Skiplist Algorithm, SIROCCO, pp124-138.
6. Colvin R., Groves L., Luchangco V., Moir M. 2006 Formal verification of a lazy concurrent list-based set. In Proceedings of Computer-Aided Verification.
7. Vyukov D., 2010. Concurrent Skip List. http://software.intel.com/sites/default/files/d6/31/33084 (Erişim Tarihi: 16.01.2014)
8. Kirschenhofer P., Martinez C., Prodinger H. 1995. Analysis of an optimized search algorithm for skip lists, Theoretical Computer Science 144:199-220.
9. Devroye L. 1992. A limit theory for random skip lists, Annals of Applied Probability, 2(3):597–609.
10. Kirschenhofer P., Prodinger H. 1994. The path length of random skip lists, Acta Informatica, 31(8):775–792.
11. Papadakis T. 1993. Skip lists and probabilistic analysis of algorithms, Ph.D. Thesis, University of Waterloo, Tech. Report CS-93-28.
12. Papadakis T., Munro J.I., Poblete P.V. 1992. Average search and update costs in skip lists, BIT 32:316–332.
13. Poblete P.V., Munro J.I., Papadakis T. 2006. The binomial transform and the analysis of skip lists, Theoretical Computer Science 352:136 –158.