BÖLÜM 19. ÖZÜT VE BİLGİ EDİNİMİ

Çoğu zaman sözlük işlemlerini (ekleme, arama, silme) içermesi istenen ve dinamik olan kümelere ihtiyaç duyulur. Örneğin, bir derleyici bir programı icra edeceği zaman bir sembol tablosu oluşturur ve bu sembol tablosunda alfanümerik karakterler ve bazı özel karakterlerden oluşan değişken isimleri vardır. Program icra edilirken, bir değişkenin ele alınması, onun değişken olup olmadığının önce anlaşılması gerekir. Bunun için sembol tablosunda verilen karakter dizisinin değişken olup olmadığı kontrol edilir. Sembol tablosu üzerinde yapılan aramaların çok etkili olması gerekir ki, programın icrası sırasında sembol tablosundan değişkene ulaşmak için fazla zaman harcanmamalıdır. Bir program icra edilirken, her blok başlangıcında sembol tablosuna yeni değişken isimleri eklenirken, her blok çıkışında sembol tablosundan değişken ismi silinir. Bu üç işlem dışında başka bir işleme ihtiyaç yoktur.

Değişken isimleri bir diziye veya listeye atılırsa, bir değişkenin isminin sembol tablosunda olup olmadığını kontrol etmek için arama algoritmasının mertebesi O(n) olur ve bu da pek tercih edilen bir sonuç değildir. Mertebesi O(1) olan algoritmaların kullanılması daha etkilidir ve bu imkanı sağlayan özel bir veri yapısı vardır ve bunlara **Çırpı (Hash) tablosu** veya **bakma tablosu** veya **direkt adres tablosu** denir.

Çırpı tablosu bir sanal veri yapısı olup, ikili arama ağaçlarının izin verdiği işlemlerin bir kısmı bu tablolar üzerinde yapılabilmektedir. Çırpı tablosunun uygulamasına çırpılama (hashing) denir. Çırpılama sabit zamanlı algoritmalarla silme, ekleme ve arama işlemlerinin yapılmasını sağlar. Sıralama işlemi bu tablolar tarafından desteklenmez. Bundan dolayı minimum bulma, maksimum bulma ve tabloyu sıralı yazma işlemleri etkili bir şekilde desteklenmez.

Genel Fikir: İdeal çırpı tablo veri yapısı, verileri içeren sabit uzunluklu bir dizidir. Arama işlemi verinin bir kısmı üzerinde yapılan bir işlemdir ve bu kısmına anahtar denir. Örneğin, bir öğrenci kaydında öğrencinin adı, soyadı gibi nüfuz bilgileri ile öğrencinin

üniversitesi ile bölümü bulunur. Fakat bunların hiçbiri anahtar değildir, fakat öğrencinin numarası anahtar olarak kullanılabilir (kendi üniversitesi içerisinde), çünkü bu numara bu üniversitede tektir ve bu öğrenciye aittir. Çırpı tablosunun kayıtların saklanması için bulundurmuş olduğu yer sayısına tablonun boyutu (m) denir ve her bir yere hücre denir.
Veriler, çırpı tablosuna yerleştirilirken her veriye 0 ile tablo boyutunun bir eksiği değerleri ve bu değerler arasında kalan değerlerden biri adres olarak verilir. Bu değeri verme işlemi gelişigüzel değildir ve bu bir algoritma ile yapılır. Bunun için kullanılan algoritmaya çırpı fonksiyonu denir ve bu fonksiyonun amacı veriler ile tablonun hücreleri arasında bire-bir eşleştirme yapmaktır. Bu fonksiyonun basit olması ve farklı olan iki anahtara farklı hücreler tayin etmesi önemlidir. Çırpı fonksiyonun verileri tabloya düzenli bir şekilde dağıtması önemlidir. Gelen verinin herhangi bir hücreye gitmesi olasılığı 1/m olmalıdır. Buna rağmen farklı olan iki anahtarın aynı hücreye eşleştirilmesi mümkündür ve buna çarpışma denir.

Çırpı Fonksiyonu: Eğer anahtar değerleri tamsayı ise

ANAHTAR mod m

Şeklinde tanımlanan fonksiyon kayda değer bir fonksiyondur. Bu durumda, çırpı fonksiyonun tanımlanması önemli bir konudur. Örneğin, anahtar değerlerinin hepsi tamsayı ve birler kısmı sıfır olsunlar ve m=10 olsun. Bu durumda yukarıda verilen çırpı fonksiyonu bütün anahtarları aynı hücreye yani 0 hücresine eşleştirecektir. Bundan dolayı tablo boyutu genellikle asal sayı olarak seçilir.

Genellikle, anahtar değerleri karakter katarı (string) şeklinde olurlar ve bu durumda çırpı fonksiyonunun tanımlanması daha fazla dikkat gerektirmektedir. Örneğin, çırpı fonksiyonu Algoritma 19.1' deki gibi verilebilir.

Algoritma 19.1. ÇırpıFonksiyonu(m,x)

▹m tablo boyutu ve x ise anahtar değeridir

1- ÇırpıDeğeri←0
2- i←0,1,...,|x|-1
3- ÇırpıDeğeri←ÇırpıDeğeri+ASCII(x(i))
4- Sonuç←ÇırpıDeğeri mod m

Karakterlerin ASCII değerlerini toplamak bazen iyi çözüm olmayabilir. Eğer tablo boyutu m=10007 olarak seçilirse ve her anahtar en fazla 8 karakter içeriyorsa, bir karakterin en büyük ASCII değeri 127 olduğundan bir anahtardan en fazla 1016 değeri elde edilir. Bunun anlamı bu tabloda 0 ile 1016 arasındaki hücreler düşünülmektedir ve bunların dışındakiler düşünülmemektedir.

Tablo boyutu m=10007 olmak üzere Türkçe alfabede 29 karakter var birde boşluk hesaba katılırsa, farklı iki harfli kelime sayısı 30^2=900 olur. Üç harfli kelime sayısı (boşluk hariç) 29^3=24389 olur. Üç karakterli anahtarlar için çırpı fonksiyonu Algoritma 19.2' de görülmektedir.

Algoritma 19.2. ÇırpıFonksiyonu(m,x)

▹m tablo boyutu ve x ise anahtar değeridir

1- ÇırpıDeğeri←ASCII(x(1))
2- ÇırpıDeğeri←ÇırpıDeğeri+ASCII(c(2))*30
3- ÇırpıDeğeri←ÇırpıDeğeri+ASCII(x(3))*900
4- Sonuç←ÇırpıDeğeri mod m

Algoritma 19.2' de görülen çırpı fonksiyonunu genelleştirecek olursak, Algoritma 19.3' te görülen çırpı fonksiyonu elde edilir.

Algoritma 19.3. ÇırpıFonksiyonu(m,x)

▷m tablo boyutu ve x ise anahtar değeridir

```
1- ÇırpıDeğeri←0
2- i←0,1,...,|x|-1
3-
   ÇırpıDeğeri←17*ÇırpıDeğeri+ASCII(x(i))
4- ÇırpıDeğeri←ÇırpıDeğeri mod m
5- Eğer ÇırpıDeğeri < 0 ise
6-     Sonuç←ÇırpıDeğeri+m
7- değilse
8-     Sonuç←ÇırpıDeğeri
```

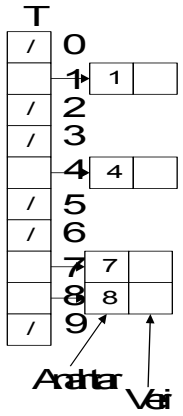
Algoritma 19.3' de verilen çırpı fonksiyonunda anahtar içindeki bütün karakterler kullanılmaktadır ve bundan dolayı verileri çırpı tablosuna iyi dağıtması beklenir. Fonksiyonun matematiksel tanımı

şeklinde olur. Dikkat edilirse, bu çırpı fonksiyonun bir polinom olduğu görülür. Örneğin, $h_k=h_0+17h_1+17^2h_2$ polinomu

$h_k=((h_2)17+h_1)17+h_0$

şeklinde olur. Böyle bir polinom için h_i sabitlerini anahtar içindeki karakterlerin ASCII değerlerinden almaktadır. Anahtarın en soldaki karakterin ASCII değeri polinomun derecesi en yüksek olan teriminin katsayısı olur ve en sağdaki karakterin ASCII değeri de polinomun sabitini belirlemektedir.

Anahtar değerleri tamsayı olan durumda diğer bir yaklaşım için elimizde anahtar değerlerini içeren bir U evrensel kümesi olsun. Bu U evrensel kümesinin eleman sayısı çok fazla olmadığı zaman direkt adres tablosu yöntemi çok etkili olur. Örneğin, her anahtar U={0,1,...,m-1} evrensel kümesinden bir değere sahip olsun (m çok büyük değil). Yapılan diğer bir kabul de iki tane farklı anahtar U evrensel kümesinden aynı değere sahip olamayacaktır. Bu durumda anahtar değeri doğrudan tabloda bir hücre adresidir. Bundan dolayı direkt adres tablosu adı verilmiştir. U={}0,1,2,3,4,5,6,7,8,9} evrensel kümesinin T={1,4,7,8} alt kümesinin çırpı tablosu Şekil 19.1' de görüldüğü gibi olur.



Şekil 19.1. *Dinamik kümenin T direkt adres tablosu ile uygulaması. U kümesindeki her değer T tablosunda bir indekse karşılık gelmektedir.*

Dinamik kümeyi ifade etmek için diziler veya direkt adres tablosu T[0,...,m-1] kullanılır. Direkt adres tablosunda her pozisyon U evrensel kümesindeki bir değere karşılıktır.

T tablosunda içeriği '/' karakteri ile gösterilen hücreler NULL (NIL) değeri içermektedir, yani T[k]=NIL, 0≤k≤9 demektir. T tablosunu kullanarak sözlük işlemleri sırasıyla Algoritma 19.4, Algoritma 19.5 ve Algoritma 19.6' da görülmektedir.

Algoritma 19.4. Arama(T,x)
▷x aranan değerin kendisi

T[x] değerini döndür.

Algoritma 19.5. Ekle(T,x)

▷x eklenecek değer

T[anahtar[x]]←x

Algoritma 19.6. Sil(T,x)

▷x silinecek değer

T[anahtar[x]]←NIL

T tablosu için verilen üç işlemin de mertebesi O(1) olduğu açıktır. Şekil 19.1' de görülen T tablosunda veri için tanımlanmış anahtar değeri saklanmaktadır ve T tablosunun bu hücresinden harici bir veri kümesine işaretçi içerilmektedir. Bazı uygulamalarda ise doğrudan verinin kendisi T tablosunun hücresine eklenir.

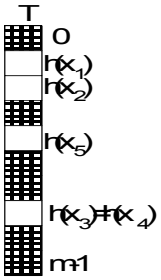
U evrensel kümesinin kullanılması ile beraber gelen zorluk aşikardır ve U kümesinin eleman sayısı çok arttığında direkt adres tablosu kullanılamaz hale gelir. Diğer bir problemde K gerçek anahtarlar kümesi olmak üzere $|K|<<|U|$ olduğu durumda T tablosunda boşta kalan çok yer olacaktır ve bundan dolayı gereksiz yere hafıza harcaması yapılmış demektir.

$|K|<<|U|$ durumunda direkt adres tablosu yerine kullanımı daha etkili olan çırpı tablosu kullanılır. Çırpı tablolarında arama işlemi hala O(1)' de yapılırken, uzay gereksinimi $\Theta(|K|)$ olur. Direkt adres tablosunda verilen bir x değeri x hücresine kayıt edilirken, çırpı tablosunda ise, x değeri h(x) hücresine kayıt edilir. h(x) fonksiyonuna **çırpı fonksiyonu**

denildiğini daha önce ifade edilmişti ve bu fonksiyon x değerini kullanarak hangi hücreye kayıt edilecekse, o hücrenin endeksini hesaplar. Burada çırpı tablosu T[0,1,...,m-1] olmak üzere h fonksiyonu U evrensel kümesini çırpı tablosuna eşleştirir ve

$h:U \leftarrow \{0,1,...,m-1\}$

olur. x değerinin çırpı değeri h(x) olup Şekil 19.2' de x değerleri ve çırpı tablosunda h(x) değerleri görülmektedir. $T \subseteq U$ ve $T=\{x_1,x_2,x_3,x_4,x_5\}$ için elde edilecek olan çırpı tablosu Şekil 19.2' de görüldüğü gibi olur.



Şekil 19.2. *Çırpı fonksiyonun U kümesinden değerleri çırpı tablosunda hücrelerle eşleştirmesi.*

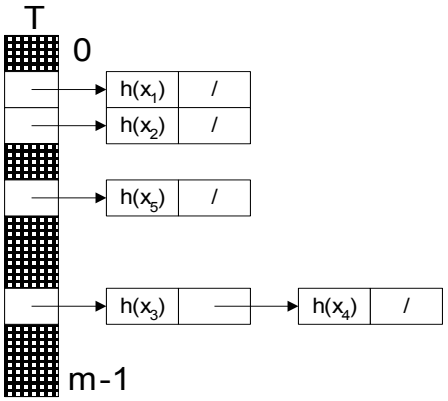
Çırpı tablosunda |U| değerinin ne kadar büyük olduğu önemli değildir; önemli olan gerçek anahtarlar kümesi olan K kümesinin eleman sayısıdır. Çünkü hafıza kullanımını bu değer belirlemektedir. Çırpı tablosunun tek sıkıntısı $h(x_i)=h(x_j)$ ve $x_i \neq x_j$ olduğu durumda ortaya çıkan çırpı fonksiyonu sonucunda aynı hücreye birden fazla kaydın denk gelme olayıdır. Buna çakışma denildiği bilinmektedir ve değişik yöntemlerle çakışma problemini çözme eğilimine gidilmiştir.

Bu yöntemlerden biri çırpı tablosunda her hücreye bir liste eklemektir. Gelen değerlerin bu listeye atılmasıdır. Şekil 19.3' te gelen değerlerin h(x) değerine göre çırpı tablosunda nasıl yerleştirildikleri görülmektedir. Çakışma olduğu zaman problem çıkmadan rahat bir şekilde tabloya değer yerleşimi yapılabilmektedir. h(x) değeri aynı olan çok sayıda anahtar olduğu durumlarda listenin boyu çok

artacağından arama işleminde fazla zaman kaybedilir. Çırpı tablosuna liste ekleme yöntemine **zincirleme** denir.

Çarpışma problemini çözmek için kullanılan zincir yönteminde her hücrenin sahip olduğu bir bağlı liste vardır. Çarpışma durumunda çarpışmanın olduğu hücreye ait bağlı listenin sonuna gelen kayıt eklenir. Her hücrenin bir bağlı listesi olduğundan bu yönteme ayrı zincirler yöntemi de denilmektedir.

Sözlük işlemleri, çırpı tablosu içinde uygulanabilmektedir ve yapı olarak basittirler. Sırasıyla Algoritma 19.7' de arama algoritması, Algoritma 19.8' de ekleme algoritması ve Algoritma 19.9' da ise silme algoritması görülmektedir.



Şekil 19.3. *Çırpı tablosuna liste ekleme ve çakışma probleminin bu yöntemle çözümü.*

Algoritma 19.7. Arama(T,x)

x değerini, T[h(x)] endeksine bağlı listede ara

Algoritma 19.8. Ekle(T,x)

x değerini, T[h(anahtar[x])] endeksine bağlı listenin başına ekle

Algoritma 19.9. Sil(T,x)

T[h(anahtar[x])] endeksine bağlı listeden x değerini sil

Ekleme algoritmasının en kötü durumu O(1) mertebesindedir. Arama ve sil algoritmaların mertebesi, o anda üzerinde bulunulan T tablosunun endeksine bağlı listenin boyutu ile orantılıdır. Silme algoritması için eğer liste çift yönlü bir liste ise, mertebe O(1) olur, tek yönlü bir liste ise, listenin boyutu ile orantılıdır.

Şekil 19.3' de ayrı zincirler yönteminin genel şekli verilmiştir. Ayrı zincirler yönteminin daha açık olarak anlaşılması için çırpı fonksiyonu

h(x)=x mod 10

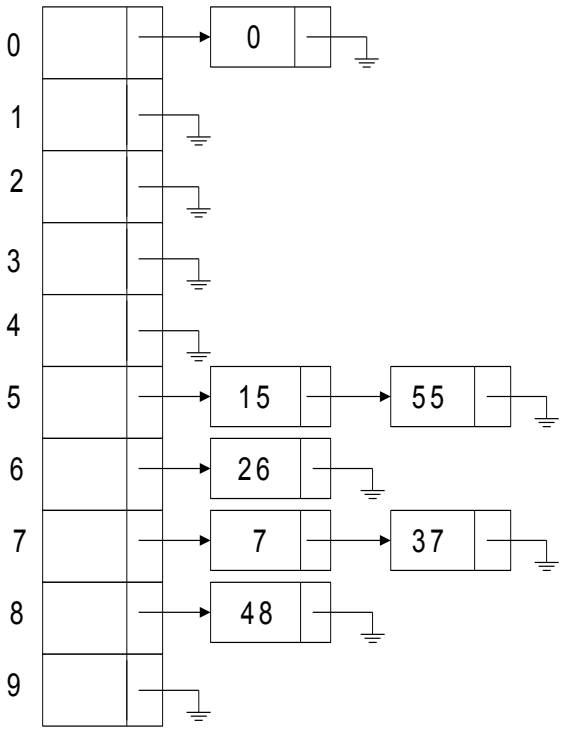
olsun. Bu fonksiyona göre boyutu m=10 olan bir çırpı tablosuna

K={0,7,15,26,37,48,55}

Anahtarları yerleştirilsin. Elde edilen şekil Şekil 19.4' de görüldüğü gibi olur.

En çok merak edilen noktalardan biri verilen bir çırpı tablosunda eğer zincirleme kullanılıyorsa, arama algoritmasının alacağı zaman ne kadar olur, konusudur.

m tane hücre içeren bir T tablosu verilsin ve bu tabloya n tane kayıt eklenmiş olsun. Bu tablo için **yükleme faktörü** n/m şeklinde tanımlanır ve α ile gösterilir. Bunun anlamı liste başına ortalama kaç tane kayıt düştüğünü göstermesidir. Bundan dolayı analizler α içeren ifadeler şeklinde olur. n ve m sonsuza giderken α değerinin sabit kaldığı kabul edilir.



Şekil 19.4. *h(x)=x mod 10 çırpı fonksiyonuna göre K={0,7,15,26,37,48,55} anahtarlarının ayrı zincirler yöntemi ile inşa edilen çırpı tablosuna yerleştirilmiş durumları.*

Zincirleme kullanılarak oluşturulan bir çırpı tablosunda en kötü durum vahim sonuç verir. Eğer tabloya n tane anahtar kayıt edildiyse; bu anahtarların hepsi aynı listeye denk gelmiş olma durumudur ve bu durumda boyutu n olan bir liste oluşmuş olur. Böyle bir tabloda eleman ekleme işleminin mertebesi O(1) olur, çünkü eklenecek eleman listenin başına eklenir ve listenin boyutundan bağımsız bir işlemdir. Silme ve arama işlemlerinin mertebeleri O(n) artı h(x) fonksiyonun hesaplama zamanı olur. Bu durum elemanların tek boyutlu bir diziye eklenmesi ve bu dizide eleman arama veya eleman silme ile aynı olur.

Bu durumda şu soru sorulabilir. Gelen kayıtları düzenli bir şekilde dağıtacak bir çırpı fonksiyonu tanımı yapılamaz mı? Günümüze kadar

kayıt dağıtımını iyi yapan çırpı fonksiyonları tanımlanmıştır. Hatta bu konu ile ilgili yeni yöntemler geliştirilmiştir. Gelen kayıtların m hücreden herhangi birine gitme olasılığı aynıdır ve daha önce kayıt edilen kayıtlardan bağımsızdır. Bu kabule **basit düzenli çırpılama** denir.

Çırpı fonksiyonu değer hesaplaması O(1) kabul edildiğinde T tablosunda bir x kaydını aramanın mertebesi T[h(x)] listesinin boyutu ile orantılı olur. Ortalama bir aramada yapılacak karşılaştırma sayısını bulmak için iki durum söz konusu olur. Birincisinde yapılacak aramanın başarısız olması; yani verilen kaydın T[h(x)] listesinde olmaması durumudur. İkinci durum ise, yapılan aramanın başarılı olmasıdır; yani verilen x kaydının T[h(x)] listesinde olması durumudur.

Bu noktaya kadar verilen verinin anahtar değerine çırpı fonksiyonu uygulandı ve elde edilen sonuç tablonun endeks değeri olarak alındı. Bu noktadan sonra iki aşamalı diyebileceğimiz iki tane farklı çırpı yöntemi üzerinde durulacaktır.

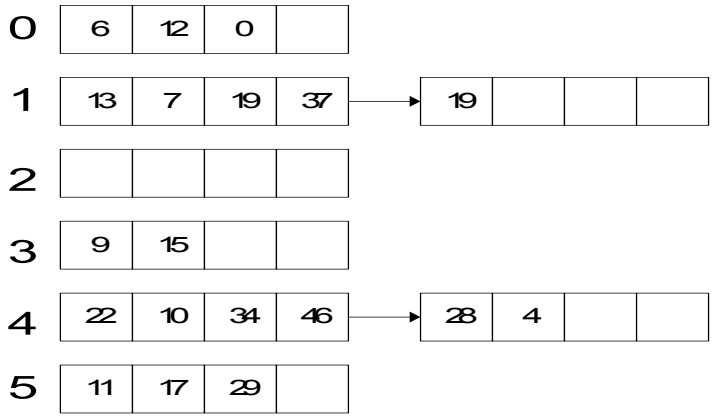
Çırpılama yöntemi, yukarıda anlatılan yöntemin genel ismidir. Verilen bir kaydın anahtarından, bu kaydın kayıt edilecek olan tablo hücresinin endeksini bulma işlemine **Çırpı algoritması** veya **Çırpı Fonksiyonu** denir.

Kayıtların saklanması tablodan yer ayrılmasını gerektirir ve bu yer ayırma işlemi yapılırken, mümkün olduğunca boş yer bırakılmamaya çalışılır. Kayıtlar için ayrılan bu alana **birincil alan** denir. Birincil alan, sepetlere bölünür ve her sepet bir veya daha fazla kayıt içerebilir.

Çırpı fonksiyonu kullanılarak bilgilerin tabloya kayıt edilmesi için endeks hesaplanır ve genelde çırpı fonksiyonu kayıtların sırasını korumaz. Bazı durumlarda farklı kayıtlar için hesaplama sonucunda aynı hücre endeksi bulunabilir. Bu duruma çakışma (collision) denir. Çakışma meydana geldiği zaman, bu problemi çözmek için birincil alandan hariç zincir alanı kullanılır. Buna zincirleme de (chaining) denir. Aynı çırpı değerli kayıtların hepsi birincil alandaki hücre doluncaya kadar bu hücreye yazılırlar. Birincil alandaki hücre dolduktan sonra zincir alanındaki hücreye yazılmaya başlanır. Taşma

alanındaki birinci hücre dolduktan sonra bir hücre daha zincir alanına eklenir ve bu sepete yazılmaya başlanır. Bu şekilde devam edilerek çakışma problemi çözülür. Şekil 19.5' te f(x)=x mod 6 fonksiyonuna göre gelen kayıtların disk adreslerinin nasıl hesaplandığı ve diske nasıl yazıldıkları görülmektedir.

Bir tablo için gerekli olan bütün uzay birincil alan için ayrılırsa, daha sonraki güncellemelerden dolayı zincirleme durumu ortaya çıkabilir ve bu zincirlerdeki hücre sayısı fazla olabilir. Çırpı fonksiyonu kayıtları düzenli dağıtmayacağından dolayı bazı çırpı değerine karşılık çok sayıda kayıt denk gelirken, bazı çırpı değeri boş kalabilir veya çok az sayıda kayıt denk gelebilir. Taşmayı sınırlamak için birincil alana gerekli olan alandan daha fazla alan ayrılır. Bundan dolayı birincil alanın yüzde kaçının en fazla dolu olacağının belirtilmesi gerekir. Birincil alan olarak ayrılmış bölgeye yazılması istenen kayıt sayısının bu alana yazılabilecek maksimum kayıt sayısına oranı **yükleme faktörü** (Lf) denilen bir değer verir.



Şekil 19.5. *Birincil alan ve taşma alanındaki sepetler. Burada çırpı fonksiyonu f(x)=x mod 6 şeklinde tanımlanmıştır.*

Bkfr, hücreye sığan kayıt sayısıdır ve M ise birincil alandaki hücre sayısı olmak üzere

$$Lf=\frac{n}{MxBkfr} \qquad (19.1)$$

olur. n dosya içindeki kayıt sayısı ve MxBkfr ise birincil alana yazılabilecek kayıt sayısıdır. Sepet boyutu ve yükleme faktörünün okuma zamanına olan etkisi incelenebilir.

19.1. Lineer Çırpı Yöntemi

Oluşturulan bir çırpı tablosunun tekrar organize edilmemesi için değişik yöntemler geliştirilmiştir ve bu yöntemlerden biri de lineer çırpı yöntemidir. Lineer çırpı yöntemi, ne kadar kayıt eklemesi yapılırsa yapılsın, yükleme faktörü sürekli sabit tutulur. Yükleme faktörünü sabit tutabilmek için, kayıt eklemeleri yapılırken belli bir kayıt eklemesi sonunda birincil alana hücre eklemesi yapılır.

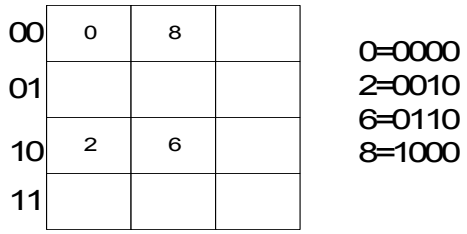
Lineer çırpı yönteminde her kaydın ikili tabanda yazılmış çırpı değerinin son bitleri kullanılarak tabloya kayıt eklemesi yapılır. Bundan dolayı ilk önce verilen kaydın çırpı değeri hesaplanır ve ondan sonra çırpı değerinin son k bitine göre tabloya yerleştirme işlemi gerçekleştirilir.

Tablo genişletildiği zaman, son k bite göre bütün kayıtları içeren sepetler son k+1 bitine göre iki parçaya bölünür. Örneğin son üç biti 011 olan bir sepet 1011 ve 0011 şeklinde iki tane sepete bölünür.

İki parçaya bölme işlemi belli bir algoritmaya göre yapılmaktadır ve bu algoritma kayıt ekleme kısmında anlatılacaktır. Bir kaydı okumak için bazı sepetlerin son k bitine göre ve bazı sepetlerde son k+1 bitine göre yerleştirme yapıldığının bilinmesi yeterlidir. Hücrelerin hangilerinin son k biti ve hangileri için son k+1 bit kullanılacağını gösteren değere **sınır değeri** denir. Sınır değeri ve k' nin değeri hafızada tutulur. İlk olarak tabloda kayıt arama için arama algoritması Algoritma 19.aşağıda verildiği gibidir.

Algoritma 19.10.LineerArama(T,x)

1. Çırpı fonksiyonun değerini hesapla
2. Çırpı değerinin son k bitine bak. Eğer son k bitin değeri sınır değerinden küçükse, aranan kaydı bulmak için son k+1 bitin kullanılması gerekir. Eğer son k bit değeri sınır değerine eşit veya büyükse, kaydın yeri son k bit kullanılarak bulunabilir.
3. Birincil alanda kayıt yoksa, taşma zincirini ardışıl olarak oku.

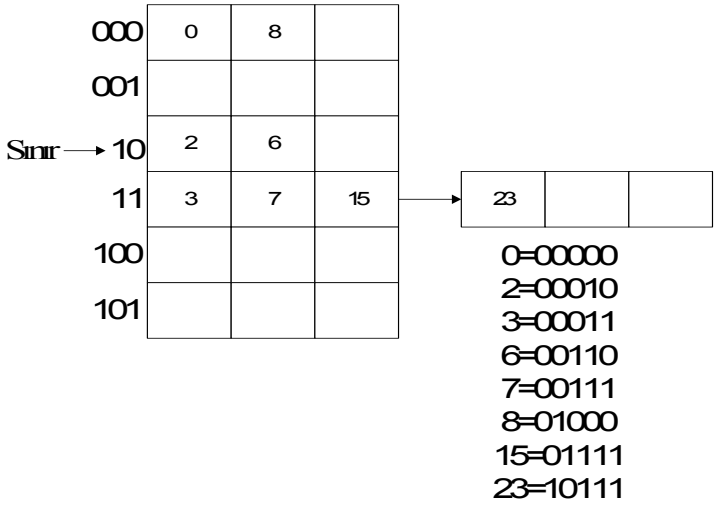


Şekil 19.6. *Lineer çırpı yöntemi ile oluşturulan tablo.*

Şekil 19.6' da görülen tabloda .ırpı değerlerinin son 2 bitine göre yerleştirme yapılmıştır ve sınır değeri 00' dır.

Şekil 19.7' de verilen çırpı değerlerinin son 2 veya 3 bitleri kullanılarak yerleştirme işlemi yapılmıştır ve sınır değeri de 10' dir. Çırpı değeri 10100 olan bir kaydın aranması için 00<10 olduğundan son 3 bit kullanılır. Eğer çırpı değeri 10110 olan bir kaydın aranması için 10=10 olduğundan son 2 bit kullanılır.

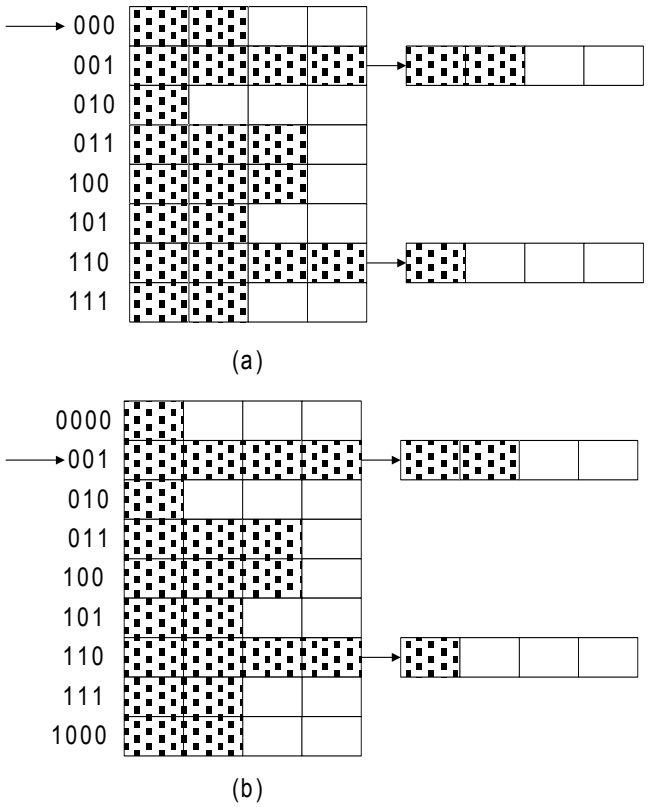
Aynı yükleme ve hücre faktörüne sahip olan lineer çırpı ve zincirli çırpı aramalarında, zincirli çırpı tablosunda arama yapma zamanı lineer çırpı tablosunda arama yapma zamanından daha küçüktür. Çünkü Lineer çırpı tablosunda son bitlere göre yerleştirme işlemi yapıldığından dolayı, bazı sepetler diğer sepetlerin yaklaşık iki katı doluluğunda olmaktadırlar.



Şekil 19.7. *Son 2 veya 3 biti kullanan lineer çırpı tablosu.*

Lineer çırpı tablosuna bir kayıt eklemek için yükleme faktörünün %75 ve Bkfr=4 olsun. Şekil 19.8 (a)' da eleman eklemeden önceki tablonun şekli görülmektedir. Dikkat edilirse, tablonun verilen yükleme faktörüne göre dolu; sınırda olduğu görülebilir ve bir tane yeni eleman ekleme ile tablonun genişleyeceği aşikardır. Şekilde de görüldüğü gibi sınır değeri 000 olup, eklenecek elemanın son üç biti 000 olsun. Bu kaydın eklenmesi sonucu tablonun yükleme faktörü değeri aşılacağından dolayı, tabloya bir sepet eklemesi yapılır ve sınır değeri 000 olduğuna göre genişleme 000 sepeti üzerinde yapılır. Eski sepetin adresi 0000 olur ve eklenen sepetin adresi 1000 olur. Yeni eklenmek istenen kayıt ile eski hücre içindeki kayıtlar bu iki hücre arasında dağıtımı yapılırken son 4 bitine göre dağıtım yapılır.

Bir lineer çırpı tablosu oluşturulduktan sonra eklenen kayıt sayısı Lf*Bkfr değerine ulaştığında veya bu değeri geçtiğinde tabloda genişleme yapılır. Şekil 19.8' de verilen tablo için 3 kayıt eklendikten sonra tabloda genişleme yapılır. Kayıt ekleme algoritması aşağıdaki gibi özetlenebilir.



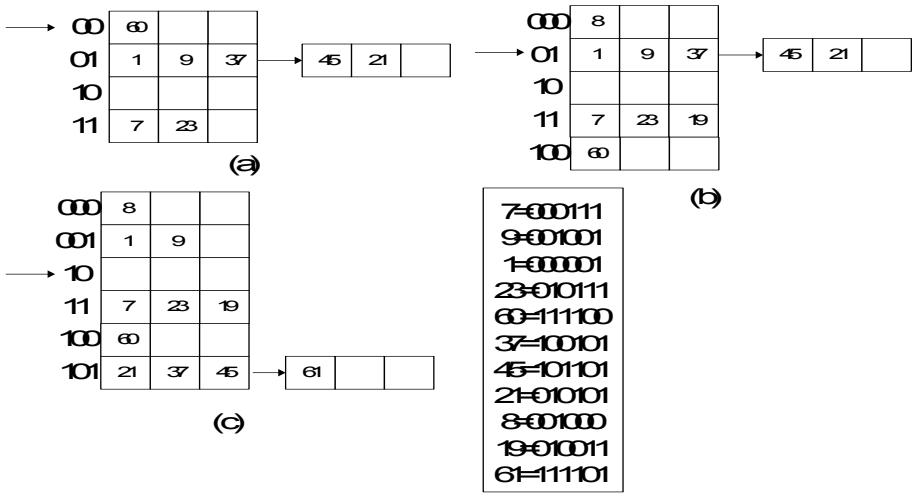
Şekil 19.8. *Lineer çırpı tablosuna eleman ekleme. (a) Sınır değeri 000 ve eklenecek olan kaydın çırpı değerinin son 3 biti 000 olsun. (b) Kayıt eklendikten sonra elde edilen lineer çırpı tablosu.*

Bu yöntemde, yüksek yükleme faktörü kullanıldığı durumlarda zincir bölgesinin oluşma ihtimali artar ve düşük yükleme faktörü de tabloda çok sık genişlemeye sebep olur. Bu tabloda 1/Bkfr olasılıkla yeni bir bağlantı (zincir bölgesine hücre ekleme) eklenir.

Bu tabloya bir kaydın eklenmesi sadece birincil alana eleman yazmadan ibaret olabilir veya zincir bölgesindeki bir hücreye eleman ekleme olabilir veya zincir bölgesine hücre ekleyerek bu kayıt bu hücreye eklenebileceği gibi birincil alana hücre ekleyerek yeni kayıt tabloya eklenir. Birincil alana hücre eklenmesi durumunda eski hücre ve bu eski hücreye bağlı olan zincir bölgesinde güncelleme gerçekleştirilir.

Algoritma 19.11.LineerEkleme(T,x)

1. Eklenecek kayıt için doğru sepeti ara.
2. Eğer sepet dolu ise, yeni bir sepet ekle ve kaydı bu sepete ekle ve bu yeni sepetin adresini eski sepetin son bloğuna ekle. Bu şekilde zincir elde edilmiş olur.
3. Eğer LfxBkfr değerinden fazla kayıt eklemesi yapılacaksa, birincil alana bir sepet ekle. O anda sınır değerinin göstermiş olduğu sepet üzerinde genişleme yapılır ve yeni sepetin adresi, sınır değerinin başına 1 değerinin eklenmesi ile elde edilir ve eski sepetin adresinin başına da 0 değeri eklenir. Eski sepet içindeki değerler , varsa eski sepete bağlı zincir içindeki kayıtlar ve eklenecek olan kayıt bu iki sepet ve zincir arasında dağıtılır.
4. Sınır değerine 1 ekle.



Şekil 19.9. *Yükleme faktörü 2/3 ve Bkfr=3 olan lineer çırpı tabloları. (a) 7, 9, 1, 23, 60, 37, 45 ve 21 kayıtlarının tabloya eklenmesi. (b) 8 ve 19 kayıtlarının eklenmesi. (c) 61 kaydının eklenmesi.*

Şekil 19.9' da yükleme faktörü 2/3 ve Bkfr=3 olan lineer çırpı yöntemi ile oluşturulan tablolar görülmektedir. Şekil 19.9 (a)' da 7, 9, 1, 23, 60, 37, 45 ve 21 kayıtlarının eklenmesi sonucunda oluşan tablo görülmektedir. Bu tabloya 8 ve 19 kayıtlarının eklenmesi için tabloda genişlemenin yapılması lazımdır. Sınır değeri 00 olduğundan genişleme bu sepet üzerinde yapılır ve bu sepetin adresi 000 ve yeni eklenen hücrenin adresi de 100 olur. Daha önce 60 kaydı 00 hücresindeyken yeni hücre eklenmesi sonucunda bu kayıt 100 hücresine eklenir ve 8 kaydı da 000 hücresine eklenir (Şekil 19.9 (b)). Şekil 19.9 (c)' da görüldüğü gibi 61 kaydının eklenmesi için tablonun genişlemesi gerekir ve genişleme 01 sepeti üzerinde yapılır. Eklenen sepetin adresi 101 olurken eski hücrenin adresi 001 olur. Eski hücreye bağlı bir zincir var ve bu zincir içinde 21 ve 45 kayıtları bulunurken eski hücre içinde 1, 9 ve 37 kayıtları bulunmaktadır. 61 kaydının eklenmesi sonucunda 21, 37, 45 kayıtları yeni eklenen hücreye ve 61 kaydı da eklenen hücreye bir zincir eklenerek bu kısma yazılır. Eski hücre içinde ise sadece 1 ve 9 kayıtları kalırken zincir kısmı ortadan kalkmıştır.

Lineer çırpı yönteminde bir kaydı silmek için, o kaydın bulunduğu hücreye bağlı zincir varsa, zincirin içindeki son kayıt alınır ve silinmesi istenen kaydın üzerine yazılır ve bu şekilde kayıt silinmiş olur. Bu kaydın taşınması sonucunda zincir hücreleri boşalırsa, zincir tamamen iptal edilir. Tabloda verilen yükleme faktörünün korunması gerekir. Eğer bu şart için gerekli olan kayıt sayısından Bkfr*Lf kadar az kayıt varsa, tabloda daraltma yapılır. Daraltma işlemi, genişletme işleminin tersidir.

Bu yöntemde kayıt eklemeleri durumda tabloda genişleme meydana gelebileceği gibi kayıt silme durumlarında da tabloda daraltma meydana gelebilir. Bundan dolayı lineer çırpı yöntemini kullanan organizasyonlarda genellikle tekrar organizasyona başvurmaya pek ihtiyaç duyulmaz.

Lineer çırpı yönteminin uygulaması biraz karmaşıktır çünkü disk üzerinde birincil alanın genişlemesi için uzay ayırımı yapmak için bitişik bloklardan yer ayrılması her zaman ve hatta genellikle mümkün değildir. Bu problem sırasız yığın dosyaları, sıralı ardışıl dosyalar içinde geçerlidir.

19.2. Teorik Bilgiler

Bu bölümde verilecek olan teorik bilgilerin detaylı şekli kaynaklardan [42] olarak verilmiştir. Burada ise, çırpı tabloları ile ilgili bazı önemli teoriler ele alınacaktır.

Teorem 19.1
Çakışmanın zincirleme ile çözüldüğü basit düzenli çırpılama yönteminde bir kaydı arama işleminin başarısız olmasının ortalama mertebesi $\Theta(1+\alpha)$ olur.

İspat
Basit düzenli çırpılama yönteminde verilen bir kaydın T tablosundaki m hücreden birine gitme olasılığı aynıdır. Bir x kaydını ortalama arama zamanı, m hücreden birinde olan listenin sonuna kadar aranması zamanının ortalama değeridir. Listelerin ortalama uzunluğu $\alpha=n/m$ kadardır. Bu durumda ortalama karşılaştırma sayısı α olur ve toplam zaman (h(x) hesaplama zamanı $O(1)$) $\Theta(1+\alpha)$ olur ♦

Teorem 19.2
Çakışmanın zincirleme ile çözüldüğü basit düzenli çırpılama yönteminde bir kaydı arama işleminin başarılı olmasının ortalama mertebesi $\Theta(1+\alpha)$ olur.

İspat
Aranan kaydın n kayıttan herhangi biri olmasının olasılığı aynı olduğu kabul edilmiştir. Kayıt eklemenin zincirin sonuna eklendiği; baş tarafa eklenmediği kabul edilmiştir. İkinci kabulün yapılması başarılı bir aramanın yapılmasında harcanacak zamanı etkilemez. Eleman ekleme listenin sonuna yapıldığından dolayı; kayıt arama için yapılacak karşılaştırma sayısı kayıt ekleme için yapılan karşılaştırma sayısından 1 fazladır. Bu durumda kayıt eklenen listenin ortalama uzunluğuna 1 eklendiği zaman arama için yapılacak karşılaştırma sayısı ortaya çıkar. Listenin ortalama uzunluğu $(i-1)/m$ olur ve başarılı bir arama için ortalama yapılacak karşılaştırma sayısı

$$\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{i-1}{m}\right)=1+\frac{1}{nm}\sum_{i=1}^{n}(i-1)$$

$$=1+\left(\frac{1}{nm}\right)\left(\frac{(n-1)n}{2}\right)$$

$$=1+\frac{\alpha}{2}-\frac{1}{2m} \qquad (19.2)$$

olur. Başarılı bir arama için çırpı fonksiyonun değer hesaplama zamanı ve karşılaştırma zamanı

$\Theta(2+\alpha/2-1/2m)$

olur ♦

Şu ana kadar anlaşılan bir durum var; o da çırpı tablosu oluşturulurken, kayıt dağılımını yapan çırpı fonksiyonun iyi olması gerektiğidir. Bunun için bölme ile kayıt dağıtma, çarpma ile kayıt dağıtma ve evrensel çırpılama yöntemleri kullanılır.

İyi bir çırpı fonksiyonu, kayıtları basit düzenli çırpılama özelliğini sağlayacak şekilde dağıtır (herhangi bir kaydın m hücreden herhangi birine denk gelme olasılığı aynıdır). P olasılık dağılımına göre U kümesinden değer seçildiği kabul edilsin; P(x) olasılığı x kaydının U kümesinden seçilme olasılığıdır. Basit düzenli çırpılama yönteminde olduğu gibi

j=0,1,..., m-1 için $\sum_{x:h(x)=j} P(x)=\frac{1}{m}$ (19.3)

olur. Çoğu zaman bu şartı kontrol etmek mümkün değildir, çünkü P olasılığı genellikle bilinmez.

Bazı durumlarda P dağılımı bilinir. Örneğin, x değerlerinin $0\le x<1$ aralığında düzenli ve bağımsız olarak dağıldığı bilinirse, çırpı fonksiyonu

$h(x)=\lfloor xm \rfloor$ (19.4)

olur.

Genelde çırpı fonksiyonları için anahtarlar evreni N={0,1,...} sayılar kümesinden oluştuğu kabul edilir. Böylece, eğer anahtarlar doğal sayı değilse, bu değerlerin doğal sayı olarak ifade edilmesinin bir yolunun bulunması gerekir. Örneğin, karakterler dizisinden oluşan bir anahtarın tamsayıya dönüştürülebilme imkanı vardır. Bu durumda genellikle karakterlerin ASCII değerleri kullanılır. Anahtarların doğal sayılardan oluştuğu kabul edilsin.

Çırpı fonksiyonu tanımlama yollarından biri, anahtar değerinin m değerine göre mod işleminin sonucu anahtarın hangi hücreye yerleştirileceği hakkında bilgi verir. Bu yönteme bölme ile çırpı fonksiyonu oluşturma denir ve çırpı fonksiyonu

$h(x)=x \bmod m$

olur. Bu fonksiyonun hesaplanması için sadece bir bölme işlemi kullanıldığından çırpı fonksiyonun mertebesi O(1) olur.

Bölme kullanılırken, tablodaki hücre sayısının bazı değerleri almaması için özen gösterilmelidir. Örneğin $m=2^p$ olmamalıdır. Eğer anahtarlar için kullanılan sayılar 10 tabanına göre iseler, $m=10^p$ olmamalıdır.

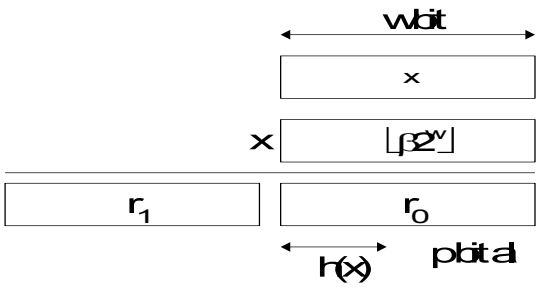
m için en güzel değer 2 sayısına fazla yakın olmayan asal sayılardır. Örneğin, çırpı tablosunda 2000 tane hücre olsun ve her hücre bir karakter dizisi barındıracaktır. Çakışma problemi de zincirleme ile çözülmektedir ve her karakter de 8 bitten oluşur. Başarısız bir aramada ortalama 3 karşılaştırma yapılması fazla önemsenmez, bundan dolayı m=2000/3=701 olarak alınabilir ve bu değer 2 sayısının kuvvetlerinden uzaktır. Her anahtar tamsayı olarak ele alınırsa, çırpı fonksiyonu

$h(x)=x \bmod 701$

olur.

Çarpma kullanılarak tanımlanan çırpı fonksiyonları da vardır. Çarpma kullanılarak çırpı fonksiyonu oluşturma iki adımdan oluşur. İlk olarak x anahtarı (0,1) aralığında olan β gibi bir sabit ile çarpılır ve virgülden sonraki kısım alınır. Elde edilen sonuç çırpı tablosundaki hücre sayısı ile çarpılır ve elde edilen sonucun taban fonksiyonu alınır. Elde edilen sonuç, çırpı fonksiyonun sonucu olur.

$h(x)=\lfloor m(\beta x \bmod 1)\rfloor$



Şekil 19.10. *Çarpma ile çırpı fonksiyonu tanımlama.*

Çarpma yönteminin avantajı, çırpı tablosundaki hücre sayısı doğrudan işin içerisine girmiyor ve sonuç üzerinde çok etkili değildir. Yani, m değeri kritik değildir. Genellikle m için $2\text{-}m=2^p$ şeklinde bir eşitlik kullanılır (p tamsayı). Bu şekilde bir seçimin yapılmasının sebebi, çoğu bilgisayar üzerinde bu değer ile işlem yapmak daha kolaydır.

Şekil 19.10' te görüldüğü gibi x anahtar değeri w-bit uzunluğunda olsun ve bu değer, (0,1) aralığında olan β gibi uygun bir sabit kullanılarak $\lfloor \beta.2^w \rfloor$ tamsayısı ile çarpılır. Sonuç 2w-bit olur ve $r_1 2^w + r_0$ olur. r_0 içerisinde en önemli p-bit seçilir ve bu seçilen değer h(x) değeri olur.

Bu yöntem ile çalışılırken, genellikle β sabitinin her değeri için iyi sonuçlar verir. Fakat verinin tabiatına bağlı olarak bazı β değerleri diğerlerine göre daha iyi sonuç verirler.

Bakma tablolarında oluşabilecek en kötü durum seçilen n tane anahtarın aynı hücreye denk gelmesi ve arama ve silme işlemlerinin mertebelerinin O(1) olmasıdır. Bundan dolayı bu gibi problemleri çözmek için değişik yöntemler uygulanmıştır. Bölme ve çarpma yöntemleri daha önce anlatıldı. Burada evrensel çırpılama denilen bir yöntem üzerinde durulacaktır. Bu yöntemde anahtar değerine bağlı olan sabit çırpı fonksiyonu kullanılmamaktadır. Çırpı fonksiyonları, anahtar değerinden bağımsız ve kayıt edilebilen gelişigüzel fonksiyon olarak seçilirler. Bu yöntemde anahtar değerleri ne olursa olsun, ortalama olarak iyi sonuç verir.

Bu yöntemin temel mantığı dikkatli bir şekilde seçilmiş olan fonksiyonlar arasında programın icrası sırasında gelişigüzel bir fonksiyon seçmektir. Gelişigüzel quicksort algoritmasının mantığı bu mantığa benzemektedir. Gelişigüzelliğin getirmiş olduğu özellik her programın icrasında farklı çırpı fonksiyonları seçilir. Bundan dolayı bu yöntem en kötü durumda iyi sonuç vereceğini garanti eder; çünkü verinin tabiatından bağımsız fonksiyonlar seçilmektedir.

Çırpı fonksiyonu olarak seçilmiş olan fonksiyonlar kümesi $\boldsymbol{H}$ olsun ve bu fonksiyonlar U kümesi içinde anahtar değerlerini {0,1,...,m-1} kümesine eşleştiren fonksiyonlardır. Eğer her $x,y \in U$ çifti için $h(x)=h(y)$ olan $h \in \boldsymbol{H}$ fonksiyon sayısı $|\boldsymbol{H}|/m$ ise, $\boldsymbol{H}$ kümesine evrensel küme denir. Bunun anlamı farklı x ve y çiftleri için $h(x)=h(y)$ olma olasılığı 1/m olur ve bu olasılık {0,1,...,m-1} kümesinde h(x) ve h(y) için gelişigüzel değer seçme olasılığı ile aynıdır.

Teorem 19.3

$n \leq m$ olmak üzere h fonksiyonu n tane anahtarı m tane hücre içeren bir tabloya dağıtacak olan ve $\boldsymbol{H}$ kümesinden seçilen bir küme ise, x gibi bir anahtar için beklenen çakışma sayısı 1' den küçük olur.

İspat

y ve z anahtarları $y \neq z$ olmak üzere $h(y)=h(z)$ ise, $c_{yz}=1$ olan bir gelişigüzel değişken olsun. Diğer durumlarda bu gelişigüzel değişkenin değeri 0 olsun. Bir anahtar çiftinin çakışma olasılığı 1/m olduğundan

$E[c_{yz}]=1/m$

olur. n tane anahtar içeren ve m tane hücresi olan T tablosunda x kaydını içeren toplam çakışma sayısı C_x olsun.

$$E[C_x]=\sum_{y\in T}E[c_{xy}]$$
$$=\frac{n-1}{m}$$

olur. $n\le m$ olduğundan $E[C_x]<1$ olur ♦

Fakat dikkati çeken bir konu var: Çırpı fonksiyonlarının evrensel sınıfını oluşturmak ne kadar kolaydır? Bazı bilgiler ışığında bunun kolay olduğu rahatlıkla görülebilir. T tablosunun boyutu m asal sayısı olsun. x sayısı r+1 byte' a ayrıştırılsın ve $x=<x_0,x_1,...,x_r>$ olur. Tek gerekli olan m sayısından küçük gerekli olan maksimum byte sayısıdır. {0,1,...,m-1} kümesinden gelişigüzel seçilen r+1 tane eleman $a=<a_0,a_1,...,a_r>$ olsun. Bununla ilişkili olan $h_a\in \boldsymbol{H}$ fonksiyonu

$$h_a(x)=\sum_{i=0}^{r}a_i x_i \bmod m \qquad (19.5)$$

şeklinde tanımlanır. Bu durumda $\boldsymbol{H}$ kümesinin

$$\boldsymbol{H}=\bigcup_a \{h_a\} \qquad (19.6)$$

m^{r+1} tane elemanı vardır.

Teorem 19.4
(19.5) ve (19.6) bağıntıları ile tanımlanan $\boldsymbol{H}$ kümesi çırpı fonksiyonlarının evrensel sınıfıdır.

İspat
$x\neq y$ olmak üzere x, y anahtar çifti düşünülsün. $x_0\neq y_0$ olduğu kabul edilsin (diğer byte pozisyonları için de aynı kabul yapılabilir).

$a_1,a_2,...,a_r$ sabit değeri için $h(x)=h(y)$ eşitliğini sağlayan a_0' ın kesinlikle bir tane değeri vardır. Bu a_0 değeri

denkleminin çözümüdür. Böylece x ve y anahtar çifti, a' nın m^r tane değer için çakışırlar ve her $<a_1,a_2,...,a_r>$ değeri için bir kez çakışırlar. a dizisinin m^{r+1} tane farklı değeri vardır ve x,y çiftinin çakışma olasılığı $m^r/m^{r+1}=1/m$ olur. Bundan dolayı ***H*** evrenseldir ♦

Diğer bir çırpı yöntemi de **açık adresleme** olarak bilinen T çırpı tablosuna kaydın kendisi kayıt edilir. Bundan dolayı T' nin hücrelerinde dinamik kümenin bir elemanı veya NIL değeri vardır. Arama yapılırken T' de istenen kayıt bulununcaya kadar bakılır veya NIL değeri görüldüğünde kaydın T' de olmadığı anlaşılır. Bu yöntemde zincirleme yoktur. Böylece tablo dolduğunda ekleme yapılamaz ve yükleme faktörü 1 değerini geçemez.

Bununla birlikte, bu yöntemde kullanılmayan hücreleri kullanan bağlı liste kullanılabilir. Fakat genelde bu yöntemde bağlı liste kullanılmamaya çalışılır. Liste kullanılmaktansa, hücreler için çakışma oranını azaltmak için boş yer tutulur ve bu şekilde arama işlemi de hızlanır.

Açık adresleme yönteminde kayıt eklemek için çırpı fonksiyon değeri hesaplanmış olan kaydın denk geldiği hücre dolu ise, bu hücreyi takip eden hücreler ardışıl olarak kontrol edilirler ve bulunan ilk boş hücreye kayıt yerleştirilir. Bir kayıt eklenirken boş hücreyi bulmak için yapılacak olan aramanın $\Theta(n)$ olmaması için çırpı fonksiyonuna ikinci bir parametre verilir ve çırpı değeri dolu hücreye denk geldiyse, bu hücreyi takip eden maksimum kaç tane hücre kontrol edileceğini gösteren bir parametredir. Bu durumda çırpı fonksiyonu

$h:UX\{0,1,...,m-1\}\rightarrow\{0,1,...,m-1\}$

olur. Verilen bir x anahtarı için incelenecek olan arama dizisi

<h(x,0), h(x,1), ..., h(x,m-1)>

şeklinde olur. Bütün bu bilgiler ışığında T çırpı tablosuna kayıt ekleme algoritması Algoritma 19.12' de görülmektedir.

Algoritma 19.12. Ekleme(T,x)

```
1-  i←0
2-  i=m oluncaya kadar tekrar et
3-      j←h(x,i)
          eğer T[j]=NIL ise
             T[j] ←x
             J' yi döndür
        değilse
          i←i+1
4-  hata: 'Tablo dolu'
```

Kayıt ekleme için incelenen hücreler kayıt arama için de incelenecektir. Böylece kayıt ekleme için boş hücre bulunduğunda kayıt eklenir ve işlem bitirilir; kayıt arama da ise ekleme için yapılan karşılaştırma sayısının bir fazlası yapılır ve kaydın olup olmadığı test edilmiş olur. Arama algoritmasında eğer kayıt T tablosunda ise, kaydın bulunduğu hücrenin endeksi değer olarak geriye döndürülür; diğer durumda NIL değeri geriye döndürülür. Algoritma 19.13' de T tablosunda kayıt arama algoritması görülmektedir.

Algoritma 19.13. Arama(T,x)

```
          1-  i←0
          2-  i=m veya T[j]=NIL oluncaya kadar tekrar et
          3-  j←h(x,i)
        eğer T[j]=x ise
          J' yi döndür
        değilse
          i←i+1
          4-  NIL döndür
```

Açık adresleme yöntemi ile oluşturulan bir bakma tablosunda kayıt silmek zordur. x gibi bir kaydı silmek için h(x) hücresine doğrudan NIL değerinin yazılması çözüm değildir; çünkü başka kayıtlarında çırpı fonksiyonu değerinin aynı hücreye denk geleceği unutulmamalıdır. Bu hücreye NIL değeri yazılırsa, bu hücreye denk gelen diğer kayıtları ulaşmak imkansız olur. NIL değeri yazma yerine bu kaydın silindiğini gösteren özel bir karakter yazılabilir. Silme işlemi bu şekilde gerçekleştirilirse, arama algoritmasının değiştirilmesi gerekir. Arama algoritmasında kayıt silinmiş karakterinin göz önüne alınması gerekir. Aynı zamanda ekleme algoritmasının da bu hücreleri boş hücre olarak kabul etmesi gerekir. Silme işlemi bu şekilde gerçekleştirildiği bir bakma tabloda arama algoritmasının mertebesi yükleme faktörü α bağlı olmaz.

Burada yapılacak analizde, her kayıt için arama dizisi {0,1,..., m-1} kümesinin m! permutasyondan biri olacağı kabul edilecektir ve buna **düzenli çırpılama** denir. Düzenli çırpılama, basit düzenli çırpılama yönteminin genelleştirilmiş şeklidir ve çırpı fonksiyonun tek değer yerine değerler dizisi ürettiği kabul edilecektir. Pratikte, düzenli çırpılama yöntemini uygulamak zordur ve bunun yerine yaklaşım kullanılır.

Açık adresleme yönteminde kullanılacak olan arama dizisini tespit etmek için üç yöntem kullanılır: **lineer araştırma** (linear probing), **kuadratik araştırma** (quadratic probing) ve **duble araştırma** (double probing). Bu üç yöntem de x gibi bir anahtar değeri için <h(x,1), h(x,2),..., h(x,m-1)> dizisinin {0,1, ..., m-1} kümesinin bir permutasyonu olduğunu garanti eder. Üç yöntem de düzenli çırpılama yönteminin bütün gereklerini tam olarak yerine getiremezler; çünkü m^2 tane farklı arama dizisi oluşturabilmektedirler. Duble araştırma yöntemi daha büyük araştırma dizileri üretebilmektedir ve bundan dolayı bu yöntem diğerlerine göre daha iyi görünmektedir.

h_1 fonksiyonu bir adi çırpı fonksiyonu olsun ve $h_1:U\rightarrow\{0,1,...,m-1\}$ şeklinde tanımlanmış olsun. **Lineer araştırma** yöntemi için çırpı fonksiyonu, i=0,1, ..., m-1 için

$$h(x,i)=(h_1(x)+i) \bmod m$$

şeklinde tanımlanır. Verilen bir x kaydı için ilk araştırılacak hücre T[h_1(x)] olur. Bir sonraki araştırılacak hücre T[h_1(x)+1] olur ve bu şekilde devam edilerek T[m-1] hücresi araştırıldıktan sonra başa dönülür ve T[0], T[1], ..., T[h_1(x)-1] hücreleri sırasıyla araştırılır. Lineer araştırma yönteminde en fazla m tane farklı araştırma dizisi oluşturulabilir.

Lineer araştırma yönteminin uygulanması çok basittir; fakat **birincil kümeleme** (primary clustering) problemi vardır. Çok uzun araştırma dizileri oluşturulur ve arama zamanı doğal olarak artar. Birincil kümeleme problemi, tablonun büyük bir kısmı boş iken, gelen kayıtların hepsinin tablonun belli bir bölgesinde yığılması durumudur.

Lineer araştırmanın yapıldığı bir çırpı tablosunda yükleme faktörü α olmak üzere kayıt ekleme ve başarısız aramada ortalama karşılaştırma sayısı

$$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$$

olur ve başarılı aramada ortalama karşılaştırma sayısı

$$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)}\right)$$

olur. Eleman ekleme ile başarısız arama aynı sayıda araştırma gerektirir. Çünkü her ikisinde de boş hücreye düşene kadar devam edilir veya tablo sonuna gelene kadar devam edilir.

Eğer kümeleme önemli değilse, ilgili formülleri elde etmek zor değildir. Çok geniş bir tablo düşünülsün ve her arama bir önceki aramadan bağımsız olduğu düşünülsün. Yükleme faktörü α değeri 1' e yakın olmadıkça çok geniş bir tabloda çarpışmalar gelişigüzel olarak çözülebilirler. İlk olarak başarısız aramada yapılan karşılaştırma sayısı bulunur. Bu boş hücre bulununcaya kadar beklenen yapılan

karşılaştırma sayısıdır. Boş hücrelerin oranı 1-α ve araştırılacak beklenen hücre sayısı da 1/(1-α) olur.

Başarılı bir arama için gerekli karşılaştırma sayısı, belli bir eleman eklendiğinde yapılan karşılaştırma sayısı ile aynı olur. Bir eleman eklenirken, bu başarısız arama sonucunda yapılan bir işlemdir. Böylece başarılı aramanın ortalama maliyetini hesaplamak için başarısız aramanın maliyeti kullanılır. Ekleme zamanının ortalamasını hesaplamak için kullanılan entegral ile ortalama tahmini yapılır ve

olur.

h_1 fonksiyonu bir adi çırpı fonksiyonu olsun ve $h_1:U\rightarrow\{0,1,...,m-1\}$ şeklinde tanımlanmış olsun. **Kuadratik araştırma** yöntemi için çırpı fonksiyonu, i=0,1, ..., m-1 için
$h(x,i)=(h_1(x)+c_1i+c_2i^2) \bmod m$

şeklinde tanımlanır. c_2 sabiti sıfırdan farklıdır. Bu yöntemde de ilk araştırılacak hücre $T[h_1(x)]$ olur. Bundan sonra araştırılacak hücre i değişkeninin ikinci kuvvetine ve c sabitlerine bağlıdır. Bu yöntem lineer araştırma yöntemine göre daha iyidir; fakat c_1, c_2 ve m değerleri bütün tablonun kullanılmasında sınırlayıcı özelliğe sahiptirler. Bu yöntemde eğer x_1 ve x_2 kayıtlarının araştırma dizilerinin ilk elemanı $h(x_1,0)=h(x_2,0)$ ise, $h(x_1,1)=h(x_2,1)$ olur. Bu şekilde devam edilerek $h(x_1,i)=h(x_2,i)$ olduğu rahatlıkla görülebilir. Bu durumun beraberinde getirdiği bir problem var ve o da **ikincil kümeleme** (secondary clustering) problemidir. Lineer araştırma yönteminde olduğu gibi ilk araştırma elemanı diğer araştırma elemanlarını etkilediğinden dolayı m tane farklı araştırma dizisi oluşturulabilir.

Duble araştırma yöntemi açık adreslemede kullanılan en iyi yöntemdir. Duble araştırma yönteminin kullandığı çırpı fonksiyonu

$h(x,i)=(h_1(x)+ih_2(x)) \bmod m$

şeklindedir ve $h_1(x)$, $h_2(x)$ yardımcı çırpı fonksiyonlarıdırlar. Araştırmaya başlama hücresi $T[h_1(x)]$ hücresi olur. Bundan sonraki hücre ise $T[(h_1(x)+h_2(x)) \bmod m]$ hücresi olur. Lineer ve kuadratik araştırma yöntemlerinden farklı olarak araştırma dizisi iki şekilde x kaydına bağlıdır; çünkü başlangıç hücresi, ofset veya her ikisi de değişebilir.

Mümkünse, $h_2(x)$ değeri ile m değeri aralarında asal olmalıdırlar. Aksi halde eğer $h_2(x)$ ve m değerlerinin d gibi bir ortak bölenlerinin en büyüğü varsa, bu araştırma yöntemi T tablosunun 1/d kısmını araştırabilir. Bu şekilde bir $h_2(x)$ tanımlayabilmek için m sürekli 2' nin kuvveti olsun ve $h_2(x)$ değeri tek olsun. Diğer bir yolda m asal bir sayı olur ve $h_2(x)$ fonksiyonunun döndürdüğü değer m' den küçük olmalıdır. Örneğin, m asal sayı olsun ve

$h_1(x)= x \bmod m$,
$h_2(x)=1+(x \bmod m_1)$

şeklinde tanımlanabilir ve m_1 değeri m' den küçük bir değerdir.

Duble araştırma yönteminde $h_1(x)$ ve $h_2(x)$ fonksiyonları birbirinden bağımsız olduklarından dolayı $\Theta(m^2)$ farklı araştırma dizisi oluşturulabilir. Bu diğer iki yönteme göre bir gelişmedir.

Zincirleme yönteminde kullanılan analizin yükleme faktörü türünden ifade edilmesine benzer olarak açık adresleme yönteminde de analiz yapılabilir. Hatırlanacak olursa, n tane kayıt m tane hücreli bir tabloda $\alpha=n/m$ olur. Açık adresleme yönteminde ise, her hücre için en fazla bir tane kayıt vardır ve bundan dolayı $n \leq m$ olur. Bunun sonucu olarak $\alpha \leq 1$ olur.

Düzenli çırpılama kullanıldığı kabul edilsin. $<h(x,0), h(x,1), ..., h(x,m-1)>$ araştırma dizisinin $\{0,1, ..., m-1\}$ kümesinin herhangi bir permutasyonu olma olasılığı aynıdır. Bunun anlamı arama veya ekleme için bu araştırma dizilerinden herhangi birinin kullanılması olasılığı aynıdır. Her kaydın sabit bir araştırma dizisi vardır. Olasılık dağılım durumu genel analiz için kullanılmaktadır.

Teorem 19.5
Yükleme faktörü $\alpha=n/m<1$ olan bir açık adresleme bakma tablosu T verilsin ve beklenen başarısız arama dizisi sayısı

$$\frac{1}{1-\alpha}$$

olur (düzenli çırpılama düşünülmüştür).

İspat
Başarısız bir aramada son kontrol edilen hücrenin içeriği boştur. p_i olasılığı, i tane araştırmanın dolu hücreye erişim yapma olasılığıdır. $i=0,1, 2, \ldots$ ve $i>n$ için $p_i=0$ olur. Bir tabloda dolu hücre sayısı en fazla n tane olur. Böylece, beklenen araştırma sayısı

$$1+\sum_{i=0}^{\infty} ip_i \qquad (19.6)$$

olur. (19.6) bağıntısının hesaplanması için q_i olasılığı en az i tane araştırma dolu hücreye erişim yapma olasılığıdır ve aynı şekilde $i=0,1,\ldots$ için olasılığın özelliğinden yola çıkarak

$$\sum_{i=0}^{\infty} ip_i = \sum_{i=1}^{\infty} q_i$$

olur. $i\geq 1$ için q_i ne olur? İlk araştırmanın dolu hücreye denk gelme olasılığı n/m olur; böylece

$$q_1 = \frac{n}{m}$$

olur. Düzenli çırpılama ile ikinci araştırma ise geriye kalan m-1 tane araştırmadan biridir ve bunlardan n-1 tanesi doludur. İlk araştırma dolu hücreye erişim yaparsa, ikinci araştırma yapılır; böylece

$$q_1 = \frac{n}{m}\cdot\frac{n-1}{m-1}$$

olur. İlk i-1 tane araştırma dolu hücrelere denk gelirlerse, i. araştırma da yapılır ve bu araştırma geriye kalan m-i+1 araştırmadan herhangi biri olma olasılığı aynıdır ve n-i+1 hücresi doludur. Böylece i=1,2, ..., n için

$$\begin{aligned} q_i &= \frac{n}{m}\cdot\frac{n-1}{m-1}\cdots\frac{n-i+1}{m-i+1} \\ &\le \left(\frac{n}{m}\right)^i \\ &= \alpha^i \end{aligned}$$

olur; çünkü j≥0 ve n≤m için

$$\frac{n-j}{m-j} \le \frac{n}{m}$$

olur. α<1 için başarısız araştırma sayısı

$$\begin{aligned} 1+\sum_{i=0}^{\infty} i p_i &= 1+\sum_{i=1}^{\infty} q_i \\ &\le 1+\alpha+\alpha^2+\alpha^3+\cdots \\ &= \frac{1}{1-\alpha} \end{aligned} \qquad (19.7)$$

olur. Bunun anlamı ilk araştırmanın gerekme olasılığı 1, ikinci araştırmanın gerekme olasılığı α, üçüncü araştırmanın gerekme olasılığı α_2 olur ve böyle devam eder ♦

Teorem 19.6

Düzenli çırpılama kullanıldığı kabul edilerek açık adresleme kullanan bir bakma tablosuna kayıt eklemek için ortalama $1/(1-\alpha)$ kadar araştırma yapılır.

İspat
Tabloda yer varsa, yeni kayıt eklemesi yapılabilir; ve böylece $\alpha<1$ olur. Bir kaydın eklenebilmesi için başarısız bir aramadan sonra bulunan boş hücreye kayıt eklenir. Böylece beklenen araştırma sayısı $1/(1-\alpha)$ olur ♦

Başarılı bir arama için gerekecek araştırma sayısını hesaplamak biraz daha komplikedir.

Teorem 19.7
Yükleme faktörü $\alpha<1$ olan bir açık-adresleme bakma tablosunda bir başarılı arama için ortalama yapılacak araştırma sayısı

$$\frac{1}{\alpha}\ln\frac{1}{1-\alpha}+\frac{1}{\alpha}$$

olur. Düzenli çırpılama kullanıldığı kabul edilmiştir ve herhangi bir kaydın aranma olasılığı aynıdır.

İspat
Bir x anahtarının aranması için gerekecek araştırma sayısı, o kaydı eklemek için gereken araştırma sayısının bir fazlasıdır. Teorem 19.6' dan faydalanarak aranacak olan x kaydı tabloya eklenen (i+1). kayıt ise, x için beklenen araştırma sayısı en fazla

$$\frac{1}{1-\frac{i}{m}}=\frac{m}{m-i}$$

olur. n tane kayıt için bu işlemin ortalama değeri bulunursa, bir başarılı aramanın ortalama araştırma sayısı ortaya çıkmış olur.

$$\frac{1}{n}\sum_{i=0}^{n-1} m \cdot r_{ai} = \frac{m}{n}\sum_{i=0}^{n-1} r_{ai}$$

$$= \frac{1}{\alpha}(H_m + H_{m-n})$$

ve H_i değeri i. harmonik değerdir. $\ln \le H_i \le \ln+1$ sınırı kullanıldığında

$$\frac{1}{\alpha}(H_m - H_n) \le \frac{1}{\alpha}(m \cdot \ln m + m - (n \ln n + n))$$

$$= \frac{1}{\alpha} m \ln \frac{m}{n} + \frac{m-n}{\alpha}$$

$$= \frac{1}{\alpha} m \ln \frac{1}{\alpha} + \frac{1-\alpha}{\alpha}$$

olur ve bu değer bir başarılı aramada beklenen ortalama araştırma sayısıdır ♦

Teorem 19.8
Tablo boyutu asal ve kuadratik araştırma kullanılırken, tablonun en az yarısı dolu ise, yeni eleman ekleme her zaman yapılabilir.

İspat
m tablo boyutu ve m>3 ve m asal olsun. İlk $\lceil m/2 \rceil$ yerlerinin hepsinin farklı olduğu gösterilecektir. Bu yerlerden iki tanesi

$(h(x)+i^2) \bmod m$ ve
$(h(x)+j^2) \bmod m$, $0 \le i,j \le \lfloor m/2 \rfloor$, $i \ne j$

şeklinde olur. Bu iki yerin farklı olmadığı kabul edilsin. Bu durumda

$h(x)+i^2 = h(x)+j^2 \pmod m$
$i^2 = j^2 \pmod m$
$i^2 - j^2 = 0 \pmod m$
$(i-j)(i+j) = 0 \pmod m$

olur. Tablo boyutu m asal olduğundan (i-j) ve (i+j) terimlerinden birinin 0 olması gerekiyor. $i \ne j$ olduğundan (i-j) teriminin 0 olması

mümkün değildir. $0 \le i,j \le \lfloor m/2 \rfloor$ olduğundan (i+j) teriminin de 0 olması mümkün değildir ♦

Bölüm Soruları

19.1. Açık adresleme ve lineer araştırma ile bakma tablosuna kayıt ekleyen bir algoritma yazınız.

19.2. Aşağıdaki bakma tabloların her biri için kayıt arama yapan algoritmaları yazınız.

a) Lineer araştırma
b) Kuadratik araştırma

19.3. Kullanılan bakma tablosunun boyutunun 13 olduğu kabul edilsin ve Çırpı fonksiyonu da mod işlemi kullanılarak tanımlanmıştır. Buna göre 10 100 32 45 58 126 3 29 200 400 0 değerleri için

a) Her değerin bakma tablosundaki adreslerini belirleyiniz ve toplam kaç tane çarpışma meydana geldi?
b) Bu sayıların rakamları toplandıktan sonra mod işlemine tabi tutulduğunda her anahtarın bakma tablosundaki adresi nedir ve toplam kaç tane çarpışma meydana geldi?
c) Bu değerler için hiç çarpışma olmayacak şekilde bir çırpı fonksiyonu tanımlayınız.

a), b) ve c) şıklarını bakma tablosunun boyutu 11 olduğunda tekrar yapınız