Karma Tablosu

Karma tablosu veri yapısı,basit bir veri yapısıdır ve sadece bir dizi elemandan oluşmaktadır.Dizideki her bir eleman , sayı veya katar olabileceği gibi herhangi bir veri yapısından da gelebilir.Tablo N elemandan oluşuyorsa ,karma tablosunun büyüklüğü N dir denir ve elemanlar 0 ile N-1 arasında normal bir dizideki gibi adreslenır.Karma veri yapısının özelliği elemanlar ile adresleri birbirine bağlayan birebir bir fonksiyon kullanılmasıdır.Bu fonksiyona karma fonksiyonu denir.

Karma tablosu sabit dizi ile tanımladığından ,büyüklüğü sabittir ve N alanı ile gösterilmektedir.

```
Public class Karma{
Ornek []tablo;
Boolean []silindi;
Int N;
Public karma(int N){
Tablo=new Ornek[N];
Silindi=new boolean[N];
This.N=N;
} }
```

5 eleman içeren örnek bir karma tablosu

0	
1	71
2	9
3	423
4	11
5	
6	76

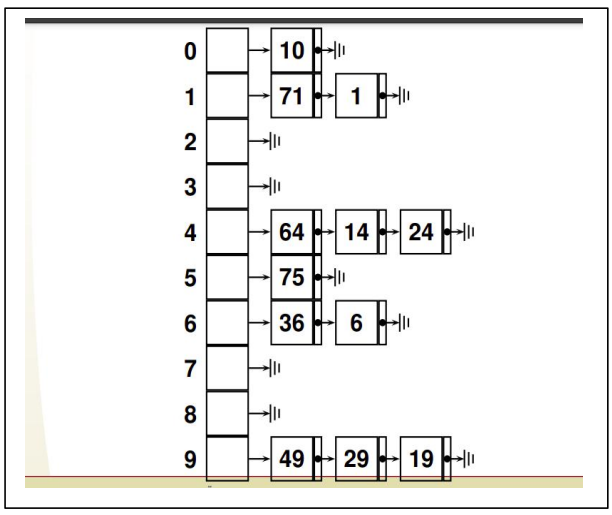
Katarlar (karakter dizileri) için kullanılabilecek örnek bir karma fonksiyonu

```
Int karma(String katar,int N){
Int i,pozisyon=0;
For(i=0;i<katar.lenght;i++){
Pozisyon=39*pozisyon+katar.charAt(i);
}
Pozisyon=pozisyon%N;,
Return pozisyon;
}
```

Türkçede 29 değişik harf bulundugu ve bir karakter dizisinde de rakamların da bulunabileceği göze alınırsa bir karakter katarında gözlemlenebilcek değşik katar sayısı 39 olacaktır.Eğer büyük küçük harf ayrımı varsa bu durumda gözlemlenebilecek karakter sayısı 2*29+10=68 olacaktır.

Baglı Liste ile Karma Tablosu Tanımı

Çakı ma problemini çözmek için kullanılabilcek ilk strateji ,aynı pozisyona adreslenen elemanları bir ba lı liste ile tutmaktır.



```
1.  Public class Karma{
2.  Liste [9 tablo;
3.  Int N;
4.  Public Karma(int N){
5.  Int i;
6.  Tablo=new Liste[N];
7.  For(i=0;i<N;i++)
8.  Tablo[i]=new Liste(9;
9.  This.N=N;
10. }
11. }
```

Önce karma tablosu için hafızada yer açılmakta,daha sonra karma tablosunun her elemanı için yenı bir bağlı liste yaratılmaktadır.Bu bağlı listelerın tamamının içi başlangıcta boştur .Yeni elemanları karma tablosuna yerleştırırken bu bağlı listelere ekleme yapılacaktır.

Her elemanı bağlı liste olan bir karma tablosunda belirli bir sayı arama

```
1.  Eleman karmaAra(int icerik){
2 .  int adres;
3 .  adres = karma(icerik);
4 .  return tablo[adres].listeAra( icerik );
5 .  }
```

Fonksiyon önce aranan sayıyı karma fonksiyonuna gönderek aranan sayının karma tablosundaki adresini belirklemektedir(3).Belirlenen adresteki bağlı liste içinde sayı aranarak işlem tamamlanmaktadır(4).

Her elemanı bağlı liste olan bir karma tablosuna yeni bir eleman ekleme

```
1 void karmaEkle(Eleman eleman){
2 int adres;
3 adres = karma(eleman.icerik);
4 tablo [adres]. listeyeEkle (eleman);
5 }
```

Fonksiyon önce aranan sayıyı karma fonksiyonuna gönderek aranan sayının karma tablosundaki adresini belirklemektedir.Belirlenen adresteki yeni listeye yeni elemanı eklenır.

Her elemanı bağlı liste olan bir karma tablosundan bir elemanı silme

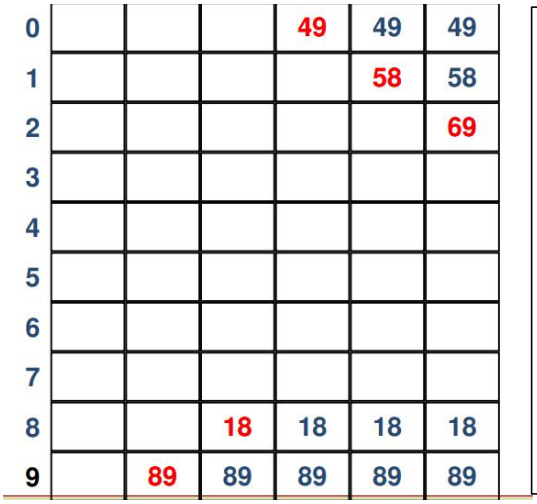
```
1 void karmaSil(int icerik){
2 Eleman eleman;
3 int adres;
4 adres = karma(icerik);
5 eleman = tablo[adres]. listeAra ( icerik );
6 if (eleman != null)
7 tablo [adres]. listedenSil (eleman);
8 }
```

Açık Adresleme

Açık adreslemede bir çakışma oldugunda boş bir pozisyon bulunana kadar alternatif pozisyonlar denenmektedir.Matematiksel olarak ifade edersek ,bir çakışma oldugunda sırasıyla karmaFonksiyon(x)+f(1)%N,karmaFonksiyon(x)+f(2)%N,karmafonksiyon(x)+f(3)% N,.. adresleri sırasıyla denenmektedir.

Dogrusal Strateji

F bir fonksiyondur ve tipik olarak f(i)=i dir.Böylece bu stratej ile aranan pozisyonun ardındaki hücreler sırasıyla denenir.



0				49	49	49
1					58	58
2						69
3						
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

29,18,49,58 ve 69 sayları girildiğinde oluşan durum gösterilmiştir. 89 ve 18 sayıları herhangi bir çakışmaya neden olmaz.49 sayı ilk çakışmaya neden olur.(49%10=9 ve 9 numaralı pozisyon dolu). 9 numaralı pozisyondan sonra gelen ilk pozisyon 0 numaralı pozisyondur ve boştur.Bu sebeple 49 bu pozisyona eklenır.58 sayısı ikinci çakışmaya neden olur(58%10=8).8 numaralı pozisyondan sonra gelen 9 ve 0 numaralı pozisyonlar dolu oldugu için ilk boş pozisyon 1 numaralı pozisyondur ve 58 bu pozisyona yerleştırlır. Sonrakiler de bu mantık ile eklenır.

Karma tablosunda bir sayı arama (dogrusal strateji)

```
1 Ornek karmaAra(int icerik){
2 int adres;
3 adres = karma(icerik);
4 while (tablo[adres] != null){
5 if (!( silindi [adres]) && tablo[adres]. icerik == icerik )
6 break;
7 adres = (adres + 1) % N;
8 }
9 return tablo[adres];
10 }
```

Karma tablosunda yeni bir eleman ekleme (dogrusal strateji)

```
1 void karmaEkle(Ornek ornek){
2 int adres;
3 adres = karma(ornek.icerik);
4 while (tablo[adres] != null && !( silindi [adres]))
5 adres = (adres + 1) % N;
6 if (tablo[adres] != null)
7 silindi [adres] = false;
8 tablo [adres] = ornek;
9 }
```

Karma tablosundan eleman silme (dogrusal strateji)

```
1 void karmaSil(int icerik){
2 int adres;
3 adres = karma(icerik);
4 while (tablo[adres] != null){
5 if (!( silindi [adres]) && tablo[adres]. icerik == icerik )
6 break;
7 adres = (adres + 1) % N;
8 }
9 silindi [adres] = true;
10 }
```

İkinci Derece Strateji

Ikinci derece strateji,doğrusal stratejinin öbeklenme problemine çözüm getirmek için önerilmiş bir yöntemdir.İkinci deece stratejide çakışma fonksiyonu ikinci dereceden bir fonksiyon olup yipik olarak $f(i)=i^2$ dir.

0				49	49	49
1						
2					58	58
3						69
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

Eklenme sırası : 89,18,49,58,69
49 sayısı ilk çakışamya neden olur .(49%10=9 ve 9 numaralı pozisyon dolu.) 9 numaralı pozisyondan sonra ilk denenen pozisyo $(9 + 1^2)\%10=0$ numaralı pozisyondur ve boştur.Bu sebeple 49 bu pozisyona eklenır. 58%10=8 ve dolu. 8 numaralı pozisyondan sonra gelen $(8+1^2)\%10=9$ numaralı pozisyon dolu oldugu için sonraki boş pozisyon $(8+2^2)\%10=2$ numaralı pozisyondur ve 58 bu pozisyona yerleştıırlır.

Çift Karma Stratejisi

Çift karma stratejide aynı pozisyona karma fonksiyonu ile adreslenen elemanların deneyecekleri pozisyonlarr yine bir karma fonksiyonu ile belirlendiği için bu stratejiye çift karma stratejisi denir.

Çift karma stratejisinde kullanlan tipk bir çakışma fonksiyonu f(i)=i X karmaFonksiyonu$_2$(x) dir.

$$karmaFonksiyonu_2(x)=S-(x\%S)$$

S karma tablosunun büyüklüğünden küçük bir asal sayı olmak üzere seçilebilir.

89,18,49,58,69 sırası ile
Ikincikarma sonksiyonu olarak 7-(x%7) Kullanılmıştır.89 ve 18 sayıları herhangi Bir çakışmaya neden olmaz.49 sayısı ilk Çakışmaya neden olur.(49%10=9 ve 9 Numaralı pozisyon dolu).9 numaralı pozis-Yondan sonra denenen ilk pozisyon (9+(7-(49%7)))%10=6 numaralı pozisyon Ve boş.49 buraya eklenir.

0						69
1						
2						
3					58	58
4						
5						
6				49	49	49
7						
8			18	18	18	18
9		89	89	89	89	89

Tekrar Karma

Karma tablosu dolmaya başladıgında ,ekleme işlemleri daha fzla zaman almaya başlayacak ve hatta ikinci dereceden strateji ile karma tablosuna ekleme yapmak mümkün olmayabilecektir.Bu durumda çözüm ,eski tablonun iki katı büyüklüğünde yeni bir karma tablosu yaratmak ve eski tablodaki tüm elemanları yeni tablodaki karma fonksiyonu ile hesaplanmış yeni yerlerına yerleştırmektır.

```
1 void tekrarKarma(){
2 int i;
3 Ornek[] tablo ;
4 boolean[] silindi ;
5 tablo = new Ornek[N];
6 silindi = new boolean[N];
7 for (i = 0; i < N; i++){
8 tablo [ i ] = this.tablo[i ];
9 silindi [ i ] = this. silindi [ i ];
10 }
11 this.tablo = new Ornek[2 * N];
12 this. silindi = new boolean[2 * N];
13 N = 2 * N;
14 for (i = 0; i < N / 2; i++)
15 if (tablo[ i ] != null && !silindi [ i ])
16 karmaEkle(tablo[i ]);
17 }
```

Önce karma tablosu ve tablodaki elemanların silinip silinmediği bilgisi gecici dizilere transfer edilmektedir(5-10).Daha sonra orjinal karma tablosu ve **silindi** dizisi iki katına büyütülmektedir(11-12).Yeni tablo büyüklüğü belirlenmekte (13),bu yeni karma tablosuna gecici diziye önceden tarnsfer edilmiş elemanlar tekrar eklenmektedir(14-13).