

Hash Tables

ساختار داده‌ای مجدول درهم‌سازی

- ساختار داده‌ای (نیمه) پوینا برای ذخیره، پیدا کردن و دسترسی به عنصر از طبق مکانیک
- برای ذخیره داده‌ها از حافظه با دسترسی نصادران (مُل آرایه‌ها) استفاده می‌کنیم
- ۱) برای تعیین آینده داده را در کدام موقعیت (آدرس) حافظه پذیرانی کنیم از تابع درهم‌سازی استفاده می‌کنیم.
- بین ترسی ساختار داده‌ای لحیف می‌شود که هزینه هر عمل جستجو، درج و حذف در میان میان

*
 $O(1)$

Hash Tables

ساختار داده ای مجدول درهم سازی

جدول (هم سازی) لیست آرایه مربوطه کردن

Search(T, K) $\rightarrow O(n) \quad O(\lg n) \quad O(n) \quad O(1)$

Insert(T, x) $\rightarrow O(1) \quad O(n) \quad O(1) \quad O(1)$

Delete(T, x) $\rightarrow O(1) \quad O(n) \quad O(1) \quad O(1)^*$

پظر میانیں //

در زمان بروکری
سُر کاس درودی

Phone number 

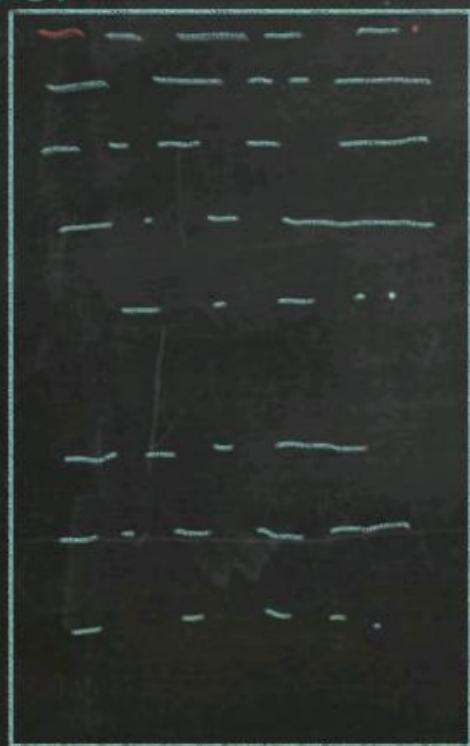
اگر ۷ رنگ کاس گیرند
اگر ۸ رنگ کاس گیرند
۱) فرستن موجوں باش
۲) نام کاس گیرند و لرند

Caller ID (Data)	Name	(key) Phone	فرستن (key)
	Ali	912123456	1
			2
			.
			.
			.
			N

Key-value dictionary
(map)

فراوانی (تعداد مکار) متس کلین (عنده من)

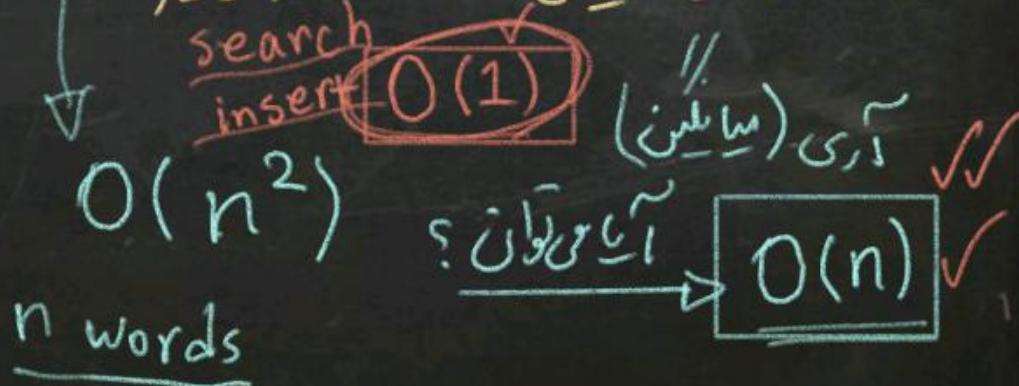
Text



- اسما حواهم مجموعه طبیعی = (متایز)

- کلین (عنده من) را می تئیم.

- فراوانی حرطیه یعنی تعداد مکار حرطیه.



n words

مُسْلِم . سُلَيْهُ . فَرَحِي طَبِين (فَرَحِي طَبِين) Contact book

Name	Tel.
J. Smith	01234567
-	-
-	-
-	-
-	-
-	-
-	-
-	-

n

و خواهیم نام مخالب ب محاسبه سُلَيْهٰ تلفن واردوه

محبَّو سُور

- اوسْ جَبَحَوی عَصْلِ هَرِسِه ($O(n)$)

- آیا می‌دان حَبَحُورَادَر ($O(1)$) با کامرسانی؟

جواب: آری

اوَسْ: فرض کنید حافظه به قَرَطَافِ بَزَرگ در اختیار

داریم به تئمیل کنیم آرایه‌ای به بزرگی کام می‌باشد تلفن‌های یکن لغزنده کنیم

Data Key ↑

روش آدرس دهن سیم

- آن میزان حسینه را در (1) 0 با کامرساند؟

جواب: آری
string name[]
= new string [10000000000]

روش: فرض نمایند حافظه در طرفی بزرگ در اینجا در

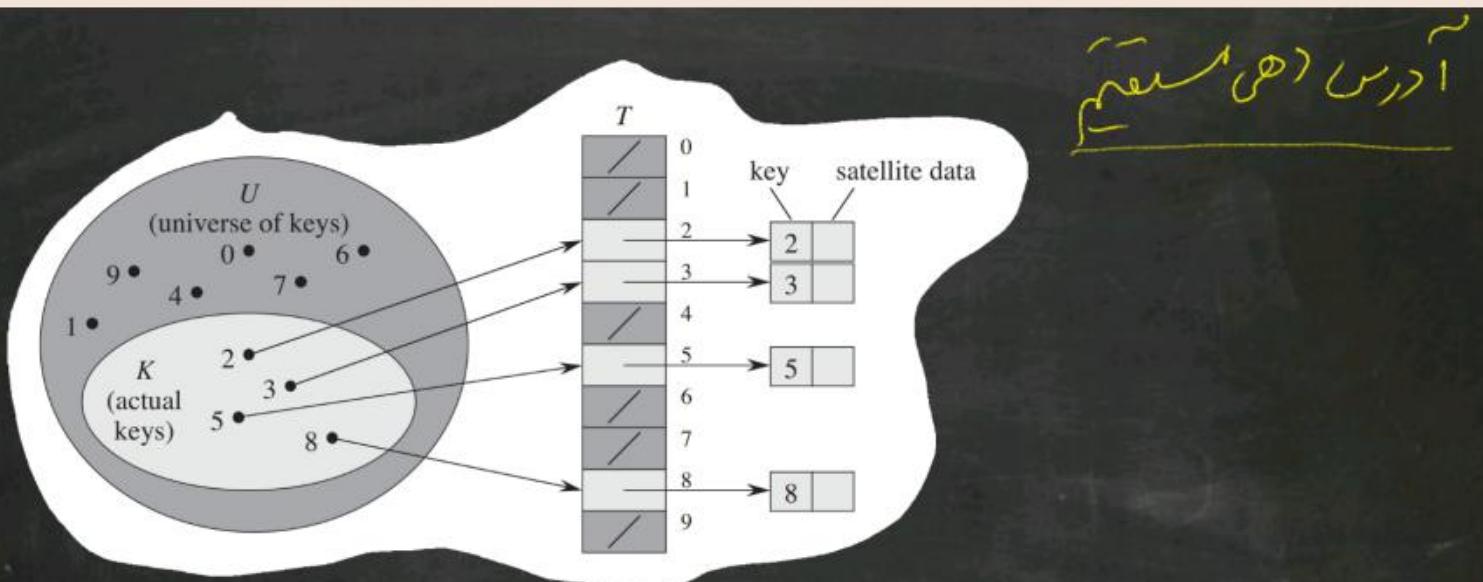
برای هر سیم کوچکی کامپیوتری به بزرگ کامپیوچر تلفنی این تعلق نداشت.
به فضایی که اندیس هر عضور آرایه (یعنی ستاره) با سیم و تلفن
مخاطبی است که نام او را در آن عضور آرایه نگذارد.

T	Name
000...0	/
000...1	/
.	:
0123456	"J.Smith"
.	:
0999999

Search (T, k)

if $T[k] \neq \text{NIL}$ then return $T[k]$

- باید این روش بحث نظری را در حافظه
لمسی و برگ این روش مناسب و کاربردی نیست.



آدرس (حکم)

آن ساختار داده‌ای در عمل کاربردی نیست
چرا که حافظه بسیار بزرگ‌تری از آن خواهد داشت
در عمل سازمانی مصرف نمی‌شود

DIRECT-ADDRESS-SEARCH(T, k)

1 **return** $T[k]$

DIRECT-ADDRESS-INSERT(T, x)

1 $T[x.key] = x$

DIRECT-ADDRESS-DELETE(T, x)

1 $T[x.key] = \text{NIL}$

بعنوان سُل در سیستم کامپیوٹر های تلفن ۱۰ روزه هست

برای آدرس دهنده مسّعی حافظه ای به بزرگی ۱۵ میلیارد رکورد باید کفی خواهد بود

در حالیکه برای فرآیند محاسبین بطور معمول حدود ۱۰۰۰ رکورد مسّرشارد

راه حل جایگزین است این دو از جدول درهم سازی

Chain Table جدول زنجیره‌ها

انواع جدول درهم سازی

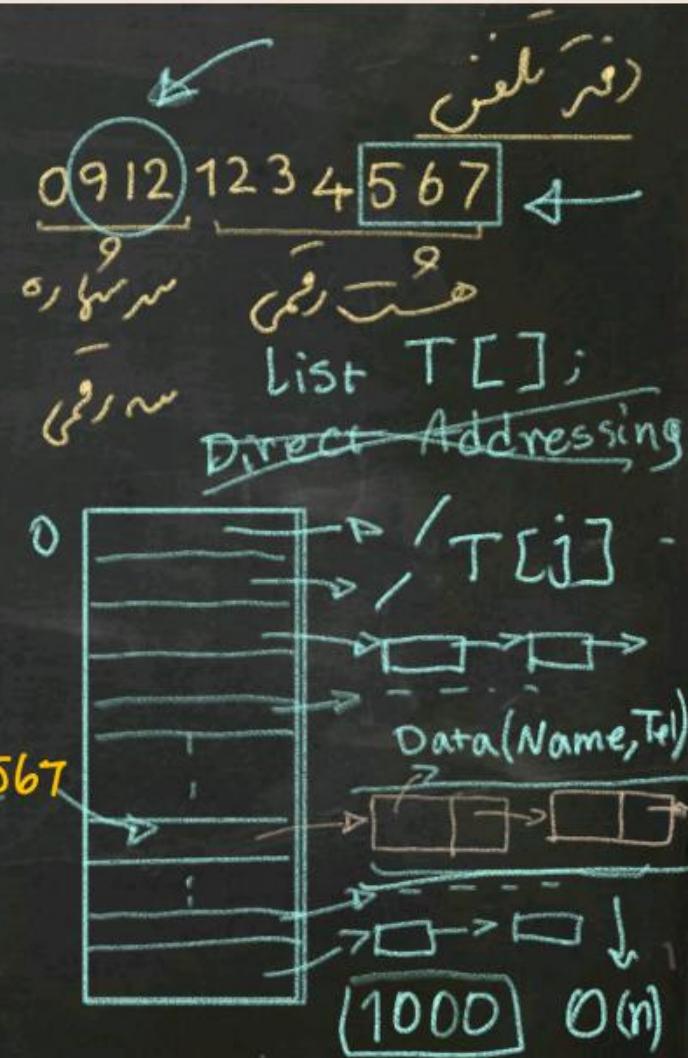
Open Addressing آدرس دهنده باز



برگه ران کے بجاء نہ

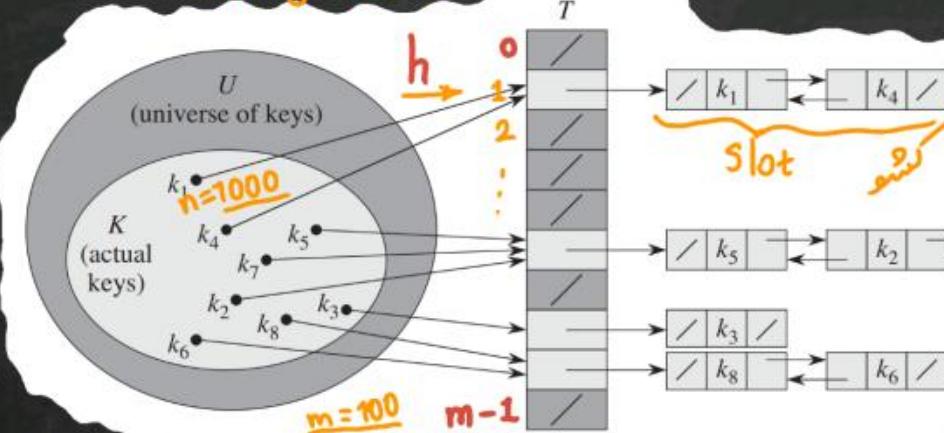
$j \leftarrow tel \% 1000$

$O(1)$



Chain Table

many-to-one



ساختار را درهای مجموعه زنجیره ها

List $T[]$

= new List[m]

: اندازه مجموعه در حالت اولیه m

$m \leq n$

بطور خطی سنبی از
با سیم

: سیار کم عناصر مجموعه

Collision

برخورد

$k, k' \in U, k \neq k'$

$h(k) = h(k')$

نگویند تابع دفعه سری برروی
برخورد k', k

CHAINED-HASH-INSERT(T, x)

1 insert x at the head of list $T[h(x.key)]$ $\rightarrow O(1)$

CHAINED-HASH-SEARCH(T, k) $O(\frac{n}{m})$ در میانگین

1 search for an element with key k in list $T[h(k)]$ $O(1)$

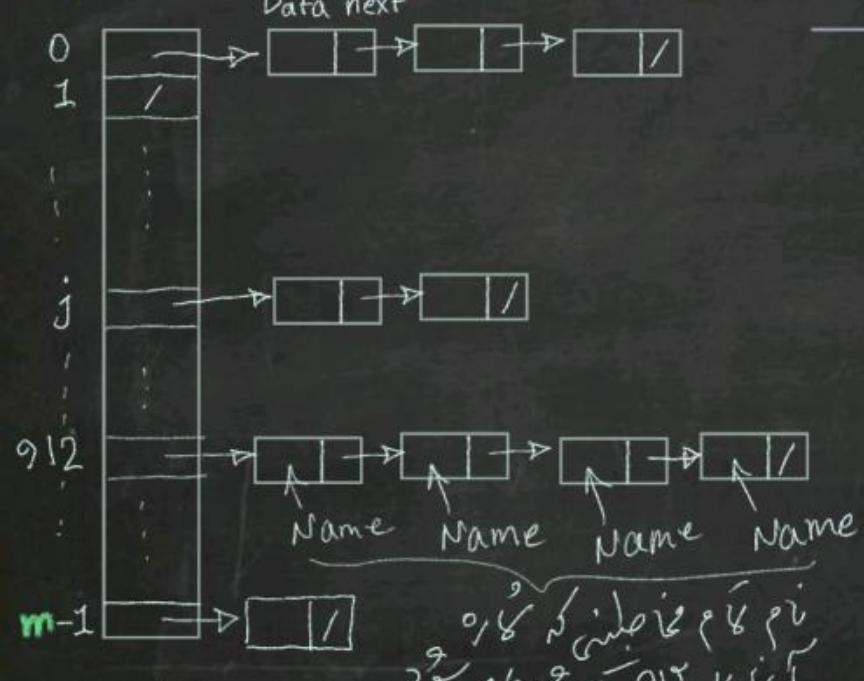
CHAINED-HASH-DELETE(T, x)

1 delete x from the list $T[h(x.key)]$ $O(n)$ در بدترین حالت

$\rightarrow O(1)$

T

Chain Table



ساختار داده‌ای جدول زنجیره‌ها

این ساختار داده‌ای آرایه‌ای از لیست‌های پونزی است.

در واقع، آرایه بجزء بالوه،

لیست پونزی head برای m کوئی را نماید.

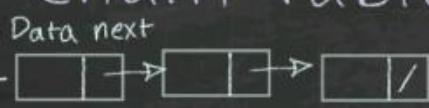
نام کام مخاطبین که گروه
آنها به ۹۱۲ شروع شود

به عنوان سُل درسته CallerId

استفاده ننم و در عوض لیست ۲۷ مخاطبین را در لیست نمایم

T

Chain Table



ساختار را درای جدول زیر نموده

فرض کنید برای تعیین این را درای با همکاری طبقه
 در کدام موقعیت جدول T (عنوان در کامپیوت)
 ذخیره شود از تابع h اینها (جگتم) $h: U \rightarrow \{0, 1, \dots, m-1\}$

$$h: U \rightarrow \{0, 1, \dots, m-1\}$$

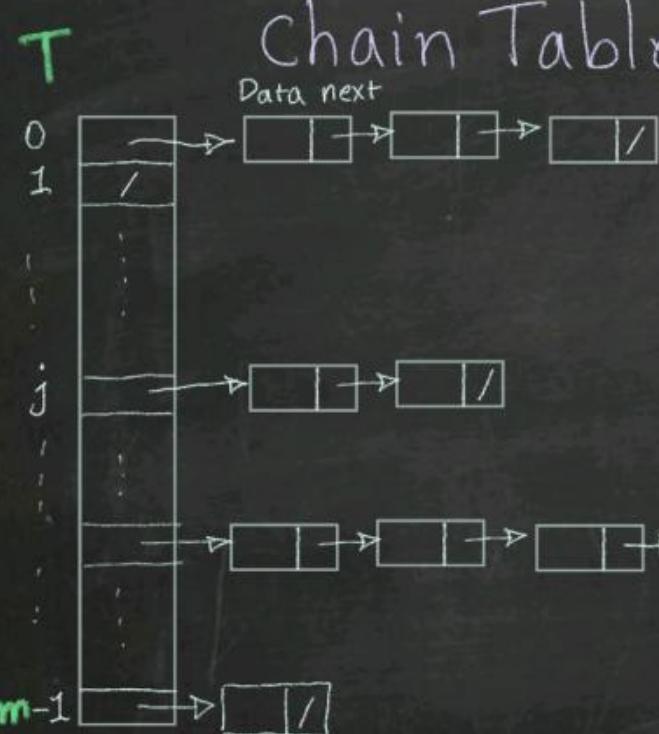
U: مجموعه جزوی معادل گذشته برای کامپیوت

$$U = \{0, 1, \dots, 999999\}$$

و فرض کنید h تابع است که به عنوان یک تابع دو رسم آخر آن سهاره نهان را ساخته است.

بطوری بین تابع یک تابع در خسازی h کوچم h hash function

Chain Table



ساختار را (کوچک) جدول زیرخواهی

$\text{Search}(T, k)$

$$j = h(k)$$

return List-Search($T[j]$, k)

لازم است تابع درخت سازی مزین

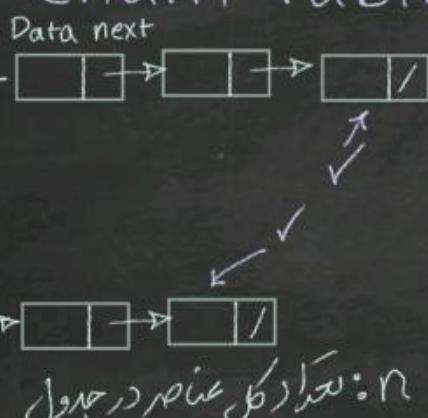
محاسبه از مرتبه $O(1)$ (دسترسی مستقیم)

درست هر سینه محبجو برابر است به هر سینه محبجویی بر روی لیست دومنی $[j]$

* این است فرق در حل لیست دومنی میتوان باشد محبجو زمان میتواند بگیرد.

T

Chain Table



ساختار را (کوچک) جدول زنجیره

Search (T, k)

$$j = h(k)$$

return List-Search($T[j], k$)

m عدد دلخواهی کرده
 n عدد از کام گرفته

هر سه "حذفی و فوچ در برگیرن حالت"

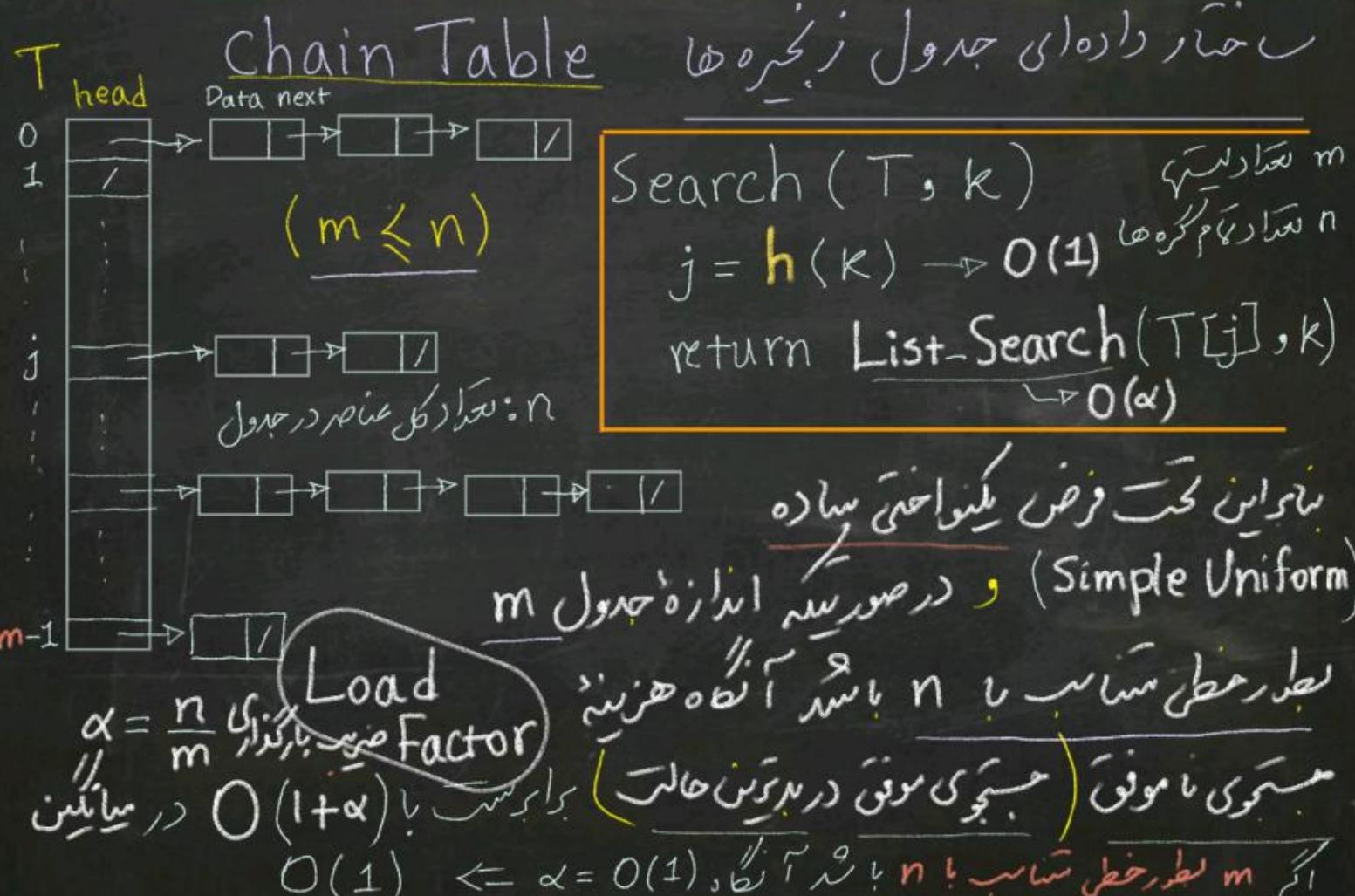
بطور مسائلیں برای است با میانلین طول

لیستی پوندی در جدول زنجیره ها

$$\alpha = \frac{n}{m}$$
 ضرب بارگذاری

طلیلی ترین سایر ایندیکس

* اگر بتوانم فرض کنم تابع دھب ری h فضای ل را به طور متواءخت به مجموع $\{0, 1, \dots, m-1\}$ منطبق کند



تابع درهم زی مطلوب باید ویرگی میتواند ساده را داشته باشد.

* از نظر تئوری در حالت کلی شیوه ای برای تعین درهم زی مناسب $\{0, 1, \dots, m-1\}$ بسته به نوع داروهای دارویی است.

$$h(k) := k \bmod m \quad \text{روش قسم.}$$

این روش زمان مناسب است که m صدر اول باشد و در از توکان کامل 2.

$$h(k) := \left\lfloor m * (\underbrace{k * A \bmod 1}_{\text{frac-part}(k * A)}) \right\rfloor \quad \text{روش ضرب.}$$

$$\begin{aligned} A &= \pi \\ A &= \sqrt{2} \\ A &= \frac{\sqrt{5}-1}{2} \end{aligned} \quad \left\{ \begin{array}{l} \text{اعداد نسبت.} \\ \text{اعداد مخلص.} \end{array} \right.$$

داده دری . تابع درهم سازی $h: U \rightarrow \{0, 1, \dots, m-1\}$

* ویرگی میتواند ساده Simple Uniform

میگویند تابع درهم سازی h ویرگی میتواند دارد اگر h نجوعه U را به طور میتواند با اعداد $\{0, 1, \dots, m-1\}$ نهاده

$$h^{-1}(j) = \{k \in U \mid h(k) = j\}$$

اگر از ای هر چند $j, j' \in \{0, \dots, m-1\}$

$$|h^{-1}(j)| = |h^{-1}(j')|$$

در حالت کلی برای درهم سازی را (۵۰) های غیر عدد صحیح

- اعداد راسخ ری

- رشته های کاراکتری

file(binary) \rightarrow string
(64char) $\xrightarrow{h \in \{0, \dots, m-1\}}$

{ string \rightarrow uint32 \rightarrow
 file(binary) \rightarrow uint32

ما داشته ایم
ما داشته ایم

که سبک آن بسیار صحیح است. (در راستای لحاظ

از الگوریتم درهم سازی مناسب است و می تشم *

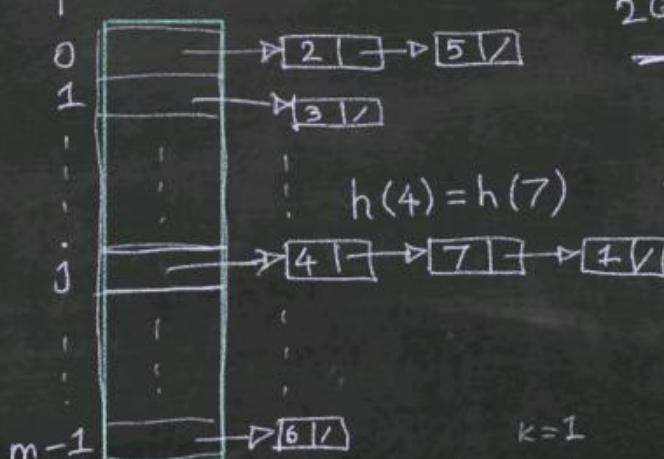
حال عدد حاصل را با اینکلاین داشتیم (شروع می شود)

ساختار را در ای آدرس (هی) باز

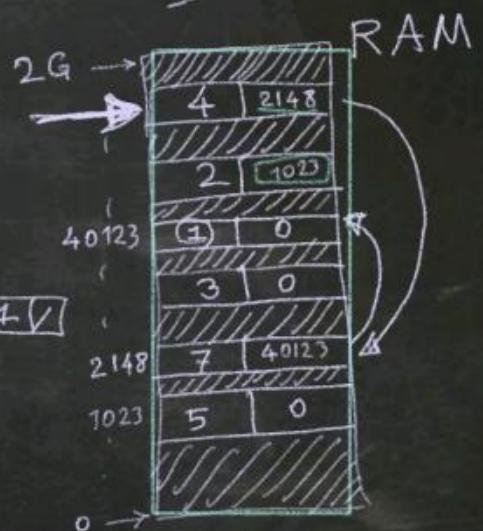
$h(k, i)$

ترکیبی آرایه پیاده سازی می شود (بعدن سازن لیست پیوندی)

chain table



$$h(4) = h(7)$$



Search(T, k)

$i=0$

$x=T[h(k, 0)]$

while ($x \neq \text{NIL}$ and $x.\text{key} \neq k$)

$i++$
 $x=T[h(k, i)]$

return x

Search(T, k)

$p=T[h(k)] \cdot \text{head}$

while ($p \neq \text{NIL}$ and $p.\text{key} \neq k$)

$\leftarrow p=p.\text{next}$

$h(k, i)$

probe

ساختار داده‌ای آدرس‌دهی باز

	T
0	79
1	
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
m-1	

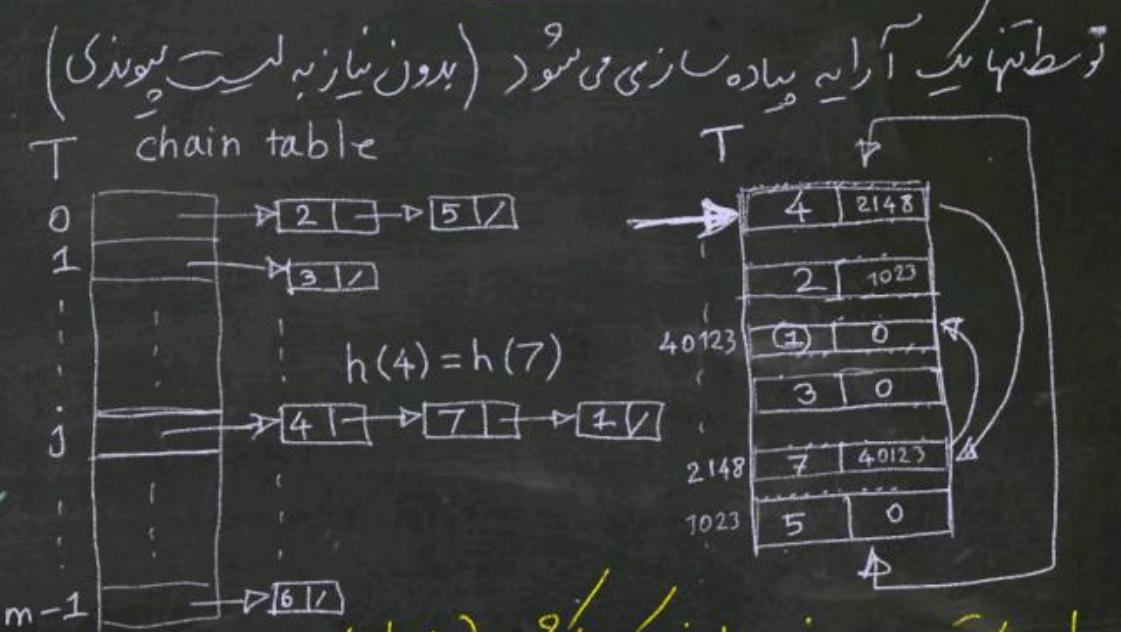
probe چنان

$i = 0$

while $i < m$

$j = h(k, i)$

$i++$



برای دسترسی به عنصر داخل سلوت (slot) در $\{0, 1, \dots, m-1\}$ از یک باعث درخواستی به فرم زیر استفاده می‌کنیم

$h : U \times \{0, 1, \dots, m-1\} \rightarrow \{0, 1, \dots, m-1\}$

ساختار داده‌ای آدرس (هی) باز

T	0	79
1		
2		
3		
4		69
5		98
6		
7		72
8		
9		14
10		
11		50
m-1		
12		

$m > n$

- ترتیبی آرایه ساده سازی می‌شود (بدون نیاز به لیست پیوندی)
- برای اندیکاتوری آرایه T از تابع درخواست سازی به فرم

$$h : U \times \{0, 1, \dots, m-1\} \rightarrow \{0, 1, \dots, m-1\}$$

$$j = h(k, i)$$

$$\begin{matrix} \text{نام} \\ \text{نام} \\ \vdots \\ \vdots \end{matrix} \quad \begin{matrix} \text{ردیف} \\ \text{ردیف} \\ \vdots \\ \vdots \end{matrix} \quad \begin{matrix} \text{کل} \\ \text{کل} \\ \vdots \\ \vdots \end{matrix} \quad \begin{matrix} \text{ک} \\ \text{ک} \\ \vdots \\ \vdots \end{matrix}$$

① $k \neq k'$, $h(k, 0) = h(k', 0)$, ② $k \neq k'$, $i \neq i'$, $h(k, i) = h(k', i')$: نتیجه -

- ورودی ضروری برای تابع درخواست سازی

نام $\in \{0, 1, \dots, m-1\}$ باشد از اعمال تابع درخواست سازی کاملاً متمم، یعنی جایگشت از اعداد

$\sigma = \langle \sigma_0, \sigma_1, \dots, \sigma_{m-1} \rangle \in S^m$: ورودی تابع احتمال کامل

$\sigma' = \langle \sigma'_0, \sigma'_1, \dots, \sigma'_{m-1} \rangle \in S^m$

$h^{-1}(\sigma) := \{k \in U \mid \langle h(k, 0), \dots, h(k, m-1) \rangle = \sigma\}$

$\forall \sigma \neq \sigma' \quad |h^{-1}(\sigma)| = |h^{-1}(\sigma')|$

عملگر درج در صارداهای آدرس (ف)

HASH-INSERT(T, k)

```
1   $i = 0$  ← سازنده
2  repeat
3       $j = h(k, i)$  بار اول بدون بررسی
4      if  $T[j] == \text{NIL}$  همچ سرت
5           $T[j] = k$ 
6          return  $j$ 
7      else  $i = i + 1$  سرت خروج از حلقة
8  until  $i == m$  ↗
9  error "hash table overflow"
```

Insert(S, x)

مجموعه دوای

Object نیز سُن x

do

- - -
- - -

while($i != m$)

عملگر محاسبه جو (رسانه های آموزشی درس دسی باز)

HASH-SEARCH(T, k)

```
1    $i = 0$ 
2   repeat
3        $j = h(k, i)$ 
4       if  $T[j] == k$ 
5           return  $j$ 
6        $i = i + 1$ 
7   until  $T[j] == \text{NIL}$  or  $i == m$ 
8   return  $\text{NIL}$ 
```

الgoritم از $h(k, i)$ مروع می شود
ورونی آرایه T "جلوی رود"

(براسن کنند ها و سی کنند)

اگر $T[j]$ برابر طبقه مورد محاسبه جو

بود \leftarrow ف را برمی گرداند

اگر $T[j] = \text{NIL}$ آنگاه بخط

واردادی برای الgoritم Insert

داشتم تصحیح دارم که کلی در T موجود است

تحلیل هزینه جستجو در صفات راه ای آدرس دهنده باز

0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
12	

m : اندازه جدول

n : تعداد عناصر موجود در جدول

ضریب بارگذاری Load factor

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

لوچنیدر جدول زیرهای $\alpha > 1$

HASH-SEARCH(T, k)

1 $i = 0$

2 repeat

3 $j = h(k, i)$

4 if $T[j] == k$

5 return j

6 $i = i + 1$

7 until $T[j] == \text{NIL}$ or $i == m$

8 return NIL

هزینه فرض نکن
جستجو مابهه

با محاسبه $O(1)$

- هزینه الگوریتم جستجو در بدترین حالت $O(n)$ آدرس دهنده باز

$$O\left(\frac{1}{1-\alpha}\right) = O\left(\frac{m}{m-n}\right)$$

(جدول پراست و جستجو ناموفق)

(برابر است با هزینه عملگر (رج))

- در میان ممکن هزینه الگوریتم جستجو در آدرس دهنده باز $O\left(\frac{1}{\alpha} \ln \frac{1}{1-\alpha}\right)$

محض ایشان تابع درجه زی و درستی میتواند همان را طرابه

جستجوی موفق

تحلیل هزینه جستجو در صفات راه ای آدرس دلخی باز

0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	14
10	
11	50
12	

$$n = 1,000,000 \cdot \text{میلیون}$$

$$m = 1,100,000$$

سیالن تعداد که نزدیک هاست رسانی
با دوچ (جستجوی ناموفق)
برابر با 11 گشته است

HASH-SEARCH(T, k)

```

1    $i = 0$ 
2   repeat
3        $j = h(k, i)$ 
4       if  $T[j] == k$ 
5           return  $j$ 
6        $i = i + 1$ 
7   until  $T[j] == \text{NIL}$  or  $i == m$ 
8   return  $\text{NIL}$ 

```

هزینه فرض ممکن
هزینه محاسبه
تابع هزار
 $O(1)$
است

$$O\left(\frac{1}{1-\alpha}\right) = O\left(\frac{m}{m-n}\right)$$

دو سیالن که نزدیک هاست
جستجوی موفق که ۳ مانند است
جدول پرانت و جستجوی ناموفق

- هزینه الگوریتم جستجو در بدترین حالت $O(n)$
(جدول پرانت و جستجوی ناموفق)

$$O\left(\frac{1}{\alpha} \ln \frac{1}{1-\alpha}\right)$$

- در سیالن هزینه الگوریتم جستجو در آدرس دلخی باز
محض ای تابع در هم زی و در کمی بیشتر از ارابه / جستجوی موفق

برخ روابط ریاضی بای ساخت می تابع در فرم سازی بای آدرس دهی باز

$$h(k, i) := (h'(k) + i) \bmod m \quad \text{Linear hash} \quad (1)$$

$$h(k, i) := (h'(k) + c_1 i + c_2 i^2) \bmod m \quad \text{Quadratic hash} \quad (2)$$

ضرایس ثابت c_1, c_2 نسبت (با استفاده از
نظریه اعداد)

Double hash (در فرم سازی صناعت)

$$h(k, i) := (h_1(k) + i \cdot h_2(k)) \bmod m$$

(رانجی) $h_1(k), h_2(k)$ دو باره (در فرم سازی) بر روی مجموع جمله های مورد

صیغه $h_1(k)$: مین ساز و مین مین صفر است (اوین یعنی)

$h_2(k)$: مین کند و فاصله مین مین صفر است (رکن های پیش سفید)

$(\forall k \in U)$. m^2 باشد m نسبت به m اول باشد $\gcd(h_2(k), m) = 1$ ضرور است

برخی روابط ریاضی برای ساختن تابع درخواستی برای آدرس دهی باز

(٣) درخواستی متعاقب Double hash

$$h(k, i) := (h_1(k) + i \cdot h_2(k)) \bmod m$$

(رانجی k ، h_1, h_2 در تابع درخواستی بروی مجموعه جزئی طبقه متن.

$h_1(k)$: تبعین کننده موافق کلیه صفات (اولین نیاز)

$h_2(k)$: تبعین کننده فاصله بین موافقین (رکن های پسرم از m)

ضرورت باشد $\forall k \in U$. $\gcd(h_2(k), m) = 1$ همواره $h_2(k)$ نسبت به m اول باشد

برای تضمین اول بودن $h_2(k)$ عدد اول m را برگیرید

$$h_2(k) \rightarrow \{1, 2, \dots, m'\} \quad m' < m$$

برای $m = 2^s$ را بگیرید کامل ۲ بگیرید

$$h_2(k) := \underbrace{(2h'(k) + 1)}_{\text{هموار}} \bmod m'$$

ملاحظات در خصوص عملگر حذف Delete برای ساختار آدرس دهنده باز

کاربرد عملگر حذف در جدول درهم سازی آدرس دهنده باز به وضویت فاصله تحقیق است (اما هر دو از این دو)

روش ۱) محتوای خانه ای که می خواهیم حذف کنیم بوج می کنیم

ایجاد. این روش باعث می شود وجود راسه باشد

معماری طبقه بسیار R در T اما الگوریتم

محبتو در سیر گانه زدن برای پیدا کردن K به NIL نیست.

روش ۲) محتوای خانه ای که می خواهیم حذف کنیم را باید معکار فکر (اده می کنیم)

که خارج از معادل مجموعه جای طبقه است به همینه "حذف شده" می کنیم.

اصطلاحاتی کوئی خانه حذف شده را علامت زنیم (Mark). معملاً معکار (-1) یعنی

$T[j] = \text{"DELETED"}$

T	4
j →	NIL
	5 -1
	NIL
	2
	7
DELETED	NIL
-1	3
	6

✓ $\text{Search} \leftarrow (\text{روی} \text{Mark} \text{ روس دوم})$ (در صورت حذف با روسی کردن)

✓ $\text{Insert}(T, k) \leftarrow$ Insert را باز نمایی از مسیر الگوریتم

if $T[j] = \text{NIL}$ OR $T[j] == \text{DELETED}$
 $T[j] = k$
return j

جستجوی ناموفق
 $O\left(\frac{m}{m-n}\right)$

اگر Delete باشد $\text{نیز طبق مکانیکی سودا}$

(اصول احذف بینصر از سر ساختار داده ای $\leftarrow n=n-1$)
جستجو سریعتر \rightarrow کاشن