## روش درهم‌سازی

از فصل ۱۱ کتاب CLRS





## مسئله‌ی جدول نمادها (Symbol Table Problem)

- جدول نمادهای $T$ که $n$ رکورد را ذخیره می‌کند.
- هر رکورد علاوه بر مولفه‌ی کلید (key) مولفه‌های دیگر هم دارد.
- اعمال: درج، حذف و جست‌وجو (اعمال فرهنگ داده‌ای)
- داده چه‌گونه باید ذخیره شود؟





## جدول آدرس دهی مستقیم (Direct Access Table)

- اگر کلیدها متمایز و $K \subseteq \{0, 1, \cdots, m-1\}$،
  $\Longleftarrow$ آرایه‌ی $T[0..m-1]$

$$T[k] = \begin{cases} x & \text{if } k \in K \text{ and } key[x] = k \\ \textbf{null} & \text{otherwise} \end{cases}$$

- اعمال در $O(1)$





## جدول آدرس دهی مستقیم (اعمال)

```
DIRECT-ADDRESS-SEARCH(T, k)
1  return T[k]

DIRECT-ADDRESS-INSERT(T, x)
1  T[key(x)] ← x

DIRECT-ADDRESS-DELETE(T, x)
1  T[key(x)] ← null
```

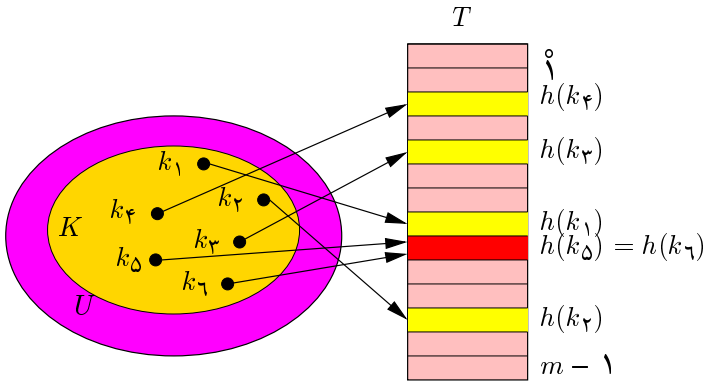
## جدول آدرس دهی مستقیم (مشکل)

- $m$ می‌تواند بسیار بزرگ باشد
- کلید ۶۴ بیتی می‌تواند ۱۸,۴۴۶,۷۴۴,۰۷۳,۷۰۹,۵۵۱,۶۱۶ مقدار متفاوت داشته باشد!
- کلید اگر رشته باشد، تعداد حالت‌ها حتا بیش‌تر از این می‌تواند باشد.





## جدول‌های درهم‌سازی (hashing tables)




عنصر $x$ با کلید $k = key(x)$ در درایه‌ی $h(k)$ ذخیره می شود. $\Longleftarrow$ تصادم (collision)





## تابع درهم‌سازی (hashing function)

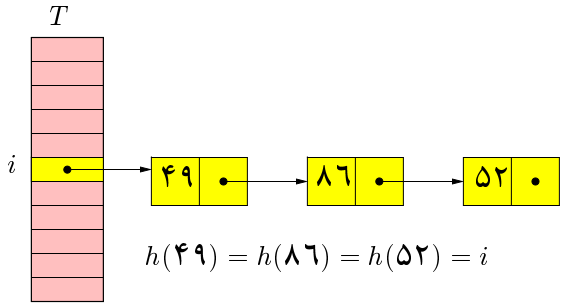
$$h : U \to \{۰, ۱, \ldots, m-۱\}$$

تعریف: اگر دو عنصر توسط تابع درهم‌سازی به یک درایه از جدول متناظر شوند، می‌گوییم یک تصادم (collision) روی داده است.





## روش زنجیره‌ای (chaining) برای حل تصادم




عناصری که به یک درایه نگاشت می‌شوند ($h(k)$ برابر) در لیست پیوندی $T[h(k)]$

لیست تهی (null)

## روش زنجیره‌ای (اعمال)

CHAINED-HASH-SEARCH$(T, k)$
1 search an element whith key $k$ in list $T[h(k)]$

CHAINED-HASH-INSERT$(T, x)$
1 insert $x$ at the head of list $T[h(key[x])]$

CHAINED-HASH-DELETE$(T, x)$
1 delete $x$ from the list $T[h(key[x])]$





## روش زنجیره‌ای (تحلیل)

- فرض: جدول درهم‌سازی $(T)$ به‌اندازه‌ی $m$، $n$ عنصر را ذخیره می‌کند.
- تعریف: $\alpha$ ضریب بارگذاری (load factor) $T$، یعنی $\alpha = \frac{n}{m}$
- $\alpha$ متوسط تعداد عناصری است که در یک زنجیره قرار می‌گیرند.





## تحلیل روش زنجیره‌ای (بدترین حالت)

- بدترین حالت: $n$ عنصر در یک درایه قرار گیرند.
- بدترین زمان برای جست‌وجو:

$O(n)$ به‌علاوه زمان لازم برای محاسبه تابع درهم‌سازی.





## تحلیل روش زنجیره‌ای (در حالت متوسط)

- فرض: تابع درهم‌سازی ساده و یکنواخت است (simple uniform hashing).
  یعنی هر عنصر با احتمال مساوی در هر یک از درایه‌های جدول قرار می‌گیرد.
- $n_j$: اندازه‌ی لیست $T_j$ $(j = 0, \ldots, m-1)$
- داریم: $n = n_0 + n_1 + \ldots + n_{m-1}$
- مقدار متوسط $n_j$ برابر است با $\alpha = \frac{n}{m}$
- فرض: مقدار $h(k)$ را در $O(1)$ حساب می‌کنیم.
- زمان مورد نیاز برای جست‌وجوی یک عنصر با کلید $k$ به‌صورت خطی به $h(k)$ بستگی دارد.

## تحلیل متوسط روش زنجیره‌ای (جست‌وجوی)

دو حالت برای یک عمل جست‌وجو: جست‌وجوی موفق یا ناموفق ندارد.

قضیه:
دریک جدول درهم‌سازی که تصادم‌ها به روش زنجیره‌ای حل شده‌اند، با فرض استفاده از تابع درهم‌سازی ساده‌ی یک‌نواخت، یک جست‌وجوی ناموفق به زمان متوسط $O(1+\alpha)$ نیاز دارد.





## تحلیل متوسط روش زنجیره‌ای (جست‌وجوی ناموفق)

اثبات:

- طبق فرض هر کلید $k$ که هنوز در جدول ذخیره نشده باشد، با احتمال یکسا
  می‌تواند در هر یک از درایه‌های جدول قرار گیرد.
- پس زمان لازم برای انجام یک جست‌وجوی ناموفق برابر زمان لازم برای جست‌وج
  تا انتهای لیست $T[h(k)]$ می‌باشد، که دارای طول متوسط $\alpha$ است.
- بنابراین زمان جست‌وجوی ناموفق برابر $O(1+\alpha)$ است.
  (زمان محاسبه $h(k)$, $O(1)$ فرض شده است.)





## تحلیل متوسط روش زنجیره‌ای (جست‌وجوی موفق)

قضیه:

دریک جدول درهم‌سازی که تصادم‌ها به به روش زنجیره‌ای حل شده‌اند، با فرض استفاده از تابع درهم‌سازی ساده‌ی یک‌نواخت، یک جست‌وجوی موفق زمان متوسط $O(1+\alpha)$ نیاز دارد.





## ادامه‌ی تحلیل متوسط روش زنجیره‌ای (جست‌وجوی موفق)

اثبات:

- فرض: $x_i$ $i$ امین عنصری باشد که در جدول درج می‌شود
  $(i = 1, 2, \ldots, n)$، و $k_i = key[x_i]$.
- متغیر تصادفی $X_{ij}$: $X_{ij} = I(h(k_i) = h(k_j))$
- داریم: $pr(h(k_i) = h(k_j)) = \frac{1}{m}$
- بنابراین $E[Xij] = \frac{1}{m}$

## تعداد متوسط عناصر مورد بررسی برای جست‌وجوی موفق

$$
\begin{aligned}
E\left[\frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}X_{ij}\right)\right] &= \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}E[X_{ij}]\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n}\frac{1}{m}\right) \\
&= 1+\frac{1}{nm}\sum_{i=1}^{n}(n-i) \\
&= 1+\frac{1}{nm}\left(n^2-\frac{n(n+1)}{2}\right) \\
&= 1+\frac{n-1}{2m} = 1+\frac{\alpha}{2}-\frac{\alpha}{2n}
\end{aligned}
$$





**بنابراین**

**زمان جست‌وجوی‌موفق (با احتساب زمان لازم برای محاسبه تابع درهمسازی) برابر است با**

$$\Theta(2+\alpha/2-\alpha/2n) = \Theta(1+\alpha)$$





## توابع درهم‌سازی

- یک تابع درهم‌سازی خوب باید یکنواخت و ساده simlpe uniform hashing
- بسیاری از توابع درهم‌سازی فرض می‌کنند کلید عناصر اعداد طبیعی هستند،
- بنابراین اگر کلید عناصری که می‌خواهیم ذخیره کنیم طبیعی نباشد باید روشی برای متناظر کردن آن‌ها با اعداد طبیعی بیابیم.
- برای مثال، اگر کلید عناصر رشته‌هایی از نویسه‌ها باشد، می‌توانیم کلید را مجموع کد اسکی نویسه‌های آن رشته در نظر بگیریم.





## توابع درهم‌سازی (روش تقسیم)

- $h(k) = k \bmod m$.
- باید از انتخاب مقادیر خاصی برای $m$ اجتناب کنیم.
- مثلاً $m$ نباید توانی از ۲ باشد، زیرا اگر $m = 2^p$، $h(k)$ معادل $p$ بیت کم‌ارزش $k$ است
- به‌تر است از مقادیر همه‌ی بیت‌ها برای محاسبه‌ی $h(k)$ استفاده شود.
- به‌طور کلی یک عدد اول که نزدیک به توانی از ۲ نباشد انتخاب مناسبی برای $n$ است.

## توابع درهم‌سازی (روش تقسیم)

توزیع و مقدار کلیدها در خوبی یا بدی تابع درهم‌سازی نقش دارند:

### مثال

- عناصر مجموعه، اعداد مجذور کامل باشند یعنی $X_i = i^۲$ $(i = ۱..۱۰۰)$ و
- $m = ۷$
- می‌دانیم $xy \bmod m = (x \bmod m)(y \bmod m) \bmod m$
- $h(x) = i^۲ \bmod ۷ = (i \bmod ۷)^۲ \bmod ۷$





| $i$ | $i \bmod ۷$ | $(i \bmod ۷)^۲ \bmod ۷$ |
|---|---|---|
| ۱ | ۱ | ۱ |
| ۲ | ۲ | ۴ |
| ۳ | ۳ | ۲ |
| ۴ | ۴ | ۲ |
| ۵ | ۵ | ۴ |
| ۶ | ۶ | ۱ |
| ۷ | ۰ | ۰ |

$۱۰۰ = ۱۴ * ۷ + ۲$

۱۴ = تصادم در درایه ۰

$۲۹ = ۱ + ۲ * ۱۴$ تصادم در درایه ۱

$۲۸ = ۲ * ۱۴$تصادم در درایه ۲

$۲۹ = ۱ + ۲ * ۱۴$ تصادم در درایه ۴





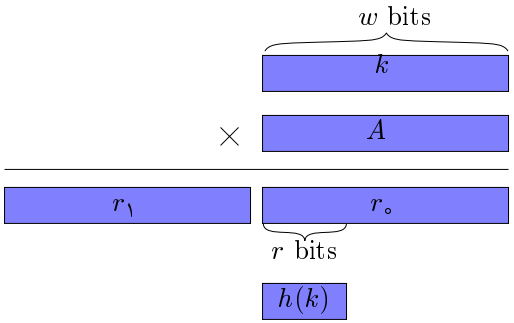
## توابع درهم‌سازی (روش ضرب)

- فرض: کلیدها اعداد صحیح، $m = ۲^r$ و کلمه‌ی کامپیوتر $w$ بیتی است.
- $h(k) = (A.k \bmod ۲^w) \text{ rsh } (w - r)$
- rsh یعنی شیفت راست بیتی
- $A$ یک عدد صحیح فرد است به‌طوری‌که $۲^{w-۱} < A < ۲^w$.
- توصیه‌ها و نکات:
  - -- $A$ نزدیک به $۲^w$ نباشد.
  - -- عمل ضرب در $\bmod ۲^w$ سریع است.
  - -- عمل rsh هم سریع است.





## توابع درهم‌سازی به روش ضرب (مثال)



## توابع درهم‌سازی به روش ضرب (مثال)

1 0 1 1 0 0 1 A

1 1 0 1 0 1 1 k

---

1 0 0 1 0 1 0 0 1 1 0 0 1 1





## مشکل توابع درهم‌سازی

- برای هر تابع درهم‌سازی $h$ که دیده‌ایم، می‌توان مجموعه‌ای از کلیدها پیدا کرد ک
  متوسط زمان جست‌وجو بسیار زیاد شود.
- مثل یک بازی: شما تابع و من کلید
- ایده‌ی راه‌حل: درهم‌سازی سراسری (universal hashing)
  -- از بین یک مجموعه از توابع درهم‌سازی یکی را به‌صورت تصادفی و مستقل ا
  کلیدها انتخاب کن
  -- حتا اگر حربق همه‌ی توابع را از قبل بداند، نمی‌تواند کلیدهای بد را انتخاب کند.
  -- چون نمی‌داند کدام تابع درهم‌سازی انتخاب خواهد شد.





## آدرس‌دهی باز (open hashing)

در آدرس‌دهی باز هر عنصر در یک درایه جدول ذخیره می‌شود

### درج

برای مشخص کردن این‌که کدام درایه‌ها باید بررسی شوند تابع درهم‌سازی را به این صورت تعریف می‌کنیم:

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

در آدرس‌دهی باز برای هر کلید $k$ نیاز به چک کردن متوالی درایه‌های زیر داریم:

$$< h(k, 0), h(k, 1), ..., h(k, m-1) >$$

```
HASH-INSERT(T, k)
 1  i ← 0
 2  repeat j ← h(k, i)
 3         if T[j] = null
 4            then T[j] ← k
 5                 return j
 6            else   i ← i + 1
 7  until i = m
 8   error "hash table overflow"
```





```
HASH-SEARCH(T, k)
 1  i ← 0
 2  repeat j ← h(k, i)
 3         if T[j] = k
 4            then return j
 5         i ← i + 1
 6  until T[j] = null   or i = m
 7  return null
```





**برای انجام عمل حذف، به‌جای این‌که در درایه عنصر حذف شده** null **قرار دهیم مقدار خاص** "deleted" **را قرار می‌دهیم (بنابراین الگوریتم درج هم نیاز به اندکی اصلاح دارد به این صورت که آن درایه می‌توان عنصر جدیدی درج کرد.)**

```
HASH-DELETE(T, k)
 1  i ← HASH-SEARCH (T, k)
 2  if i ≠ null
 3     then T[j] ← "eleted"
```





```
HASH-INSERT(T, k)
 1  i ← 0
 2  repeat j ← h(k, i)
 3         if T[j] = null   or T[j] = "deleted"
 4            then T[j] ← k
 5                 return j
 6            else   i ← i + 1
 7  until i = m
 8   error "hash table overflow"
```

## روش‌های معمول برای آدرس‌دهی باز

- **وارسی خطی** (linear probing)
- **وارسی درجه ۲ و** (quadratic probing)
- **وارسی دوگانه** (double hashing)





## وارسی خطی

$$h(k,i) = (h'(k) + i) \bmod m$$

**برای** $i = 0, 1, ..., m-1$





## وارسی درجه دو

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

$h'$ **یک تابع درهم‌سازی کمکی است و**

$c_1, c_2$ $(c_2 \neq 0)$ **ثابت‌های کمکی هستند**

$(i = 0, 1, ..., m-1)$

**اگر دو کلید وارسی اولیه مشابه داشته باشند رشته وارسی‌های آن‌ها مثل هم است:**

$$h(k_1, 0) = h(k_2, 0) \Longrightarrow h(k_1, i) = h(k_2, i)$$





## درهم‌سازی دوگانه

**درهم‌سازی دوگانه تابع درهم‌سازی به این شکل دارد:**

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

**که** $h_2, h_1$ **توابع درهم‌سازی کمکی هستند.**

## تحلیل آدرس‌دهی باز

- در آدرس‌دهی باز در هر درایه جدول حداکثر یک عنصر قرار می‌گیرد،
- $\alpha = \frac{n}{m} \leq ۱ \Longleftarrow n \leq m$
- فرض: درهم‌سازی یکنواخت
- دراین صورت رشته وارسی: $< h(k,۰), h(k,۱), \ldots, h(k,m-۱) >$
- درج یا جست‌وجوی کلید $k$: معادل یکی از جایگشت‌های $< ۰, ۱, \ldots, m-۱ >$





## تحلیل آدرس‌دهی باز (جست‌وجوی ناموفق)

**قضیه:**

**حداکثر تعداد وارسی‌ها در یک جدول درهم‌سازی باز با $\alpha < ۱$ برای یک جست‌وجوی ناموفق برابر است با $\frac{۱}{۱-\alpha}$**





## اثبات

$X$: یک متغیر تصادفی برابر با تعداد وارسی‌ها در یک جست‌وجوی ناموفق

$A_i$: رخداد این‌که $i$ امین وارسی به درایه‌ی پر برود.

رخداد $\{X \geq i\}$ یعنی: یک جست‌وحوی ناموفق که حتماً $i-۱$ وارسی اولش پر بوده است و پس از آن یکی از وارسی‌هایش ($i$ ام یا بیش‌تر) تهی بوده است. پس

$$\{X \geq i\} = zA_۱ \cap A_۲ \cap \ldots \cap A_{i-۱}$$





$$\begin{aligned}
\Pr\{X \geq i\} &= \Pr\{A_۱ \cap A_۲ \cap \ldots \cap A_{i-۱}\} \\
&= \Pr\{A_۱\} \times \Pr\{A_۲|A_۱\} \times \Pr\{A_۳|A_۱ \cap A_۲\} \ldots \\
&= \frac{n}{m} \times \frac{n-۱}{m-۱} \times \frac{n-۲}{m-۲} \times \ldots \times \frac{n-i+۲}{m-i+۲} \\
&\leq \left(\frac{n}{m}\right)^{i-۱} \\
&= \alpha^{i-۱}
\end{aligned}$$

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{\infty} i\Pr\{X = i\} \\
&= \sum_{i=1}^{\infty} i(\Pr\{X \geq i\} - \Pr\{X \geq i+1\}) \\
&= \sum_{i=1}^{\infty} \Pr\{X \geq i\} \\
&\leq \sum_{i=1}^{\infty} \alpha^{i-1} \\
&= \sum_{i=0}^{\infty} \alpha^{i} \\
&= \frac{1}{1-\alpha}
\end{aligned}
$$





## تحلیل آدرس دهی باز (درج)

**نتیجه:**

درج یک عنصر دریک جدول درهم‌سازی باز در حالت متوسط حداکثر $\frac{1}{1-\alpha}$ وارسی نیا
دارد.

**اثبات:**

یک عنصر درج می‌شود اگر حداقل یک فضای خالی وجود داشته باشد. بنابراین $\alpha < 1$
درضمن درج یک عنصر مستلزم انجام یک جست‌وجو و قرار دادن آن در اولین درایه‌ی
خالی است. پس تعداد متوسط وارسی‌ها با توجه به قضیه‌ی قبل برابر $\frac{1}{1-\alpha}$ خواهد بود.





## تحلیل آدرس دهی باز (جست‌وجوی موفق)

**قضیه:**

مقدار متوسط تعداد وارسی‌ها در یک جدول درهم‌سازی باز با $\alpha < 1$ برای یک جست‌وجوی موفق حداکثر برابر است با

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





**اثبات:**

جست‌وجو برای $k$ همان دنباله‌ی وارسی‌ها را دارد که اگر $k$ را بخواهیم درج کنیم.
اگر $k$ $(i+1)$ امین کلیدی باشد که درج می‌شود، متوسط وارسی‌ها برابر است با:

$$1/(1 - i/m) = m/(m-i)$$

پس

$$
\begin{aligned}
\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
&= \frac{1}{\alpha}(H_m - H_{m-n})
\end{aligned}
$$

که $H_i = \Sigma_{j=1}^{i} 1/j$ تابع هارمونی $i$ ام است.

$$
\begin{aligned}
\frac{1}{n}\sum_{i=0}^{n-1}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=0}^{n-1}\frac{1}{m-i} \\
\frac{1}{\alpha}(H_m - H_{m-n}) &= \frac{1}{\alpha}\sum_{k=m-n+1}^{m} 1/k \\
&\le \frac{1}{\alpha}\int_{m-n}^{m}(1/x)dx \\
&= \frac{1}{\alpha}\ln\frac{m}{m-n} \\
&= \frac{1}{\alpha}\ln\frac{1}{1-\alpha}
\end{aligned}
$$