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 $\{0, 1, \cdots, m-1\}$ $K \subseteq$،
  $\Longleftarrow$ 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 $\mathcal{O}(1)$

جدول آدرس دهی مستقیم (اعمال)

$\underline{\text{Direct-Address-Search}}(T, k)$
1 **return** $T[k]$

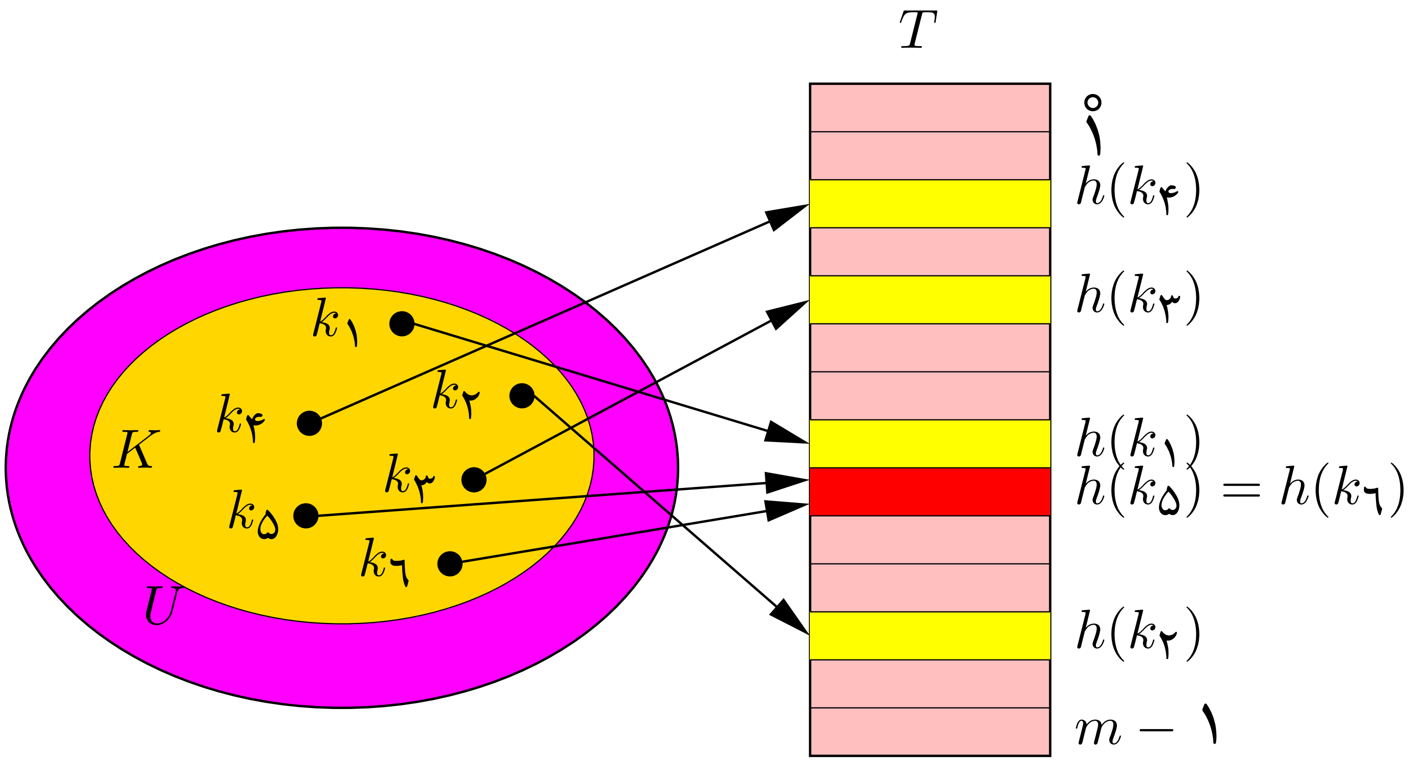
$\underline{\text{Direct-Address-Insert}}(T, x)$
1 $T[key(x)] \leftarrow x$

$\underline{\text{Direct-Address-Delete}}(T, x)$
1 $T[key(x)] \leftarrow$ **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)



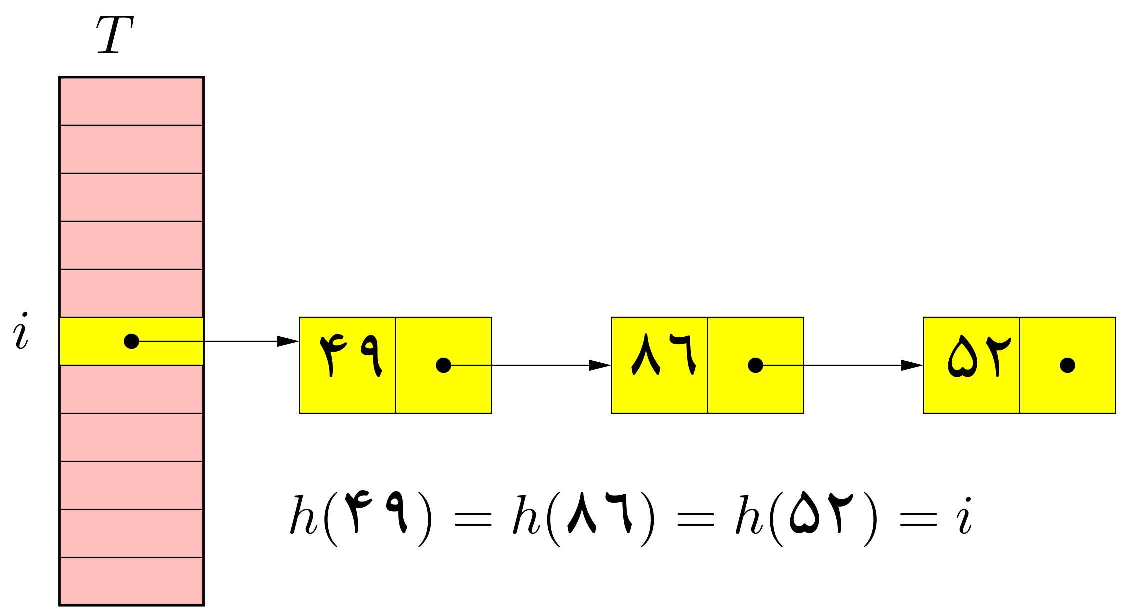
عنصر x با کلید $k = key[x]$ در درایه‌ی $h(k)$ ذخیره می شود. $\Longleftarrow$ برخورد (collision)

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 برخورد (collision) روی داده است.

روش زنجیره‌ای (chaining) برای حل برخورد



$$h(۴۹) = h(۸۶) = h(۵۲) = i$$

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

لیست تهی (null)

روش زنجیره‌ای (اعمال)

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

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

<u>CHAINED-HASH-DELETE</u> (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: α ضریب بارگذاری (load factor) T، یعنی $\alpha = \frac{n}{m}$
- α میانگین 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:**

$\mathcal{O}(n)$ به‌علاوه زمان لازم برای محاسبه تابع درهم‌سازی.

تحلیل روش زنجیره‌ای (در حالت میانگین)

- فرض: تابع درهم‌سازی ساده و یکنواحت است (simple uniform hashing).
یعنی هر عنصر با احتمال مساوی در هر یک از درایه‌های جدول قرار می‌گیرد.
- n_j: اندازه‌ی لیست T_j $(j = ۰ \ldots, m - ۱)$
- داریم: $n = n_۰ + n_۱ + \ldots + n_{m-۱}$
- مقدار میانگین n_j برابر است با $\alpha = \frac{n}{m}$
- فرض: مقدار $h(k)$ را در $\mathcal{O}(۱)$ حساب می‌کنیم.
- زمان مورد نیاز برای جست‌وجوی یک عنصر با کلید k به‌صورت خطی به $n_{h(k)}$ بستگی دارد.

تحلیل میانگین روش زنجیره‌ای (جست‌وجو)

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

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

تحلیل میانگین روش زنجیره‌ای (جست‌وجوی ناموفق)

اثبات:

- طبق فرض هر کلید k که هنوز در جدول ذخیره نشده باشد، با احتمال یکسان می‌تواند در هر یک از درایه‌های جدول قرار گیرد.
- پس زمان لازم برای انجام یک جست‌وجوی ناموفق برابر زمان لازم برای جست‌وجو تا انتهای لیست $T[h(k)]$ می‌باشد، که دارای طول میانگین α است.
- بنابراین زمان جست‌وجوی ناموفق برابر $\mathcal{O}(۱+\alpha)$ است.
(زمان محاسبه $h(k)$، $\mathcal{O}(۱)$ فرض شده است.)

تحلیل میانگین روش زنجیره‌ای (جست‌وجوی موفق)

قضیه:

دریک جدول درهم‌سازی که برخوردها به به روش زنجیره‌ای حل شده‌اند، با فرض استفاده از تابع درهم‌سازی ساده‌ی یک‌نواخت، یک جست‌وجوی موفق زمان میانگین $\mathcal{O}(۱+\alpha)$ نیاز دارد.

ادامه‌ی تحلیل میانگین روش زنجیره‌ای (جست‌وجوی موفق)

اثبات:

- فرض: x_i، iامین عنصری باشد که در جدول درج می‌شود ($i = ۱, ۲, \ldots, n$)، و $k_i = key[x_i]$.
- متغیر تصادفی X_{ij}: $X_{ij} = I(h(k_i) = h(k_j))$
- داریم: $pr(h(k_i) = h(k_j)) = \frac{۱}{m}$
- بنابراین $E[X_{ij}] = \frac{۱}{m}$

تعداد میانگین 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

$$\Theta(۲ + \alpha/۲ - \alpha/۲n) = \Theta(۱ + \alpha)$$

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 $m = ۲^p$، $h(k)$ 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 m 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
٦	٦	۱
۷	۰	۰

$۱۴ * ۷ + ۲ = ۱۰۰$

۱۴ = برخورد در درایه ۰

۲۹ = ۱ + ۲ * ۱۴ برخورد در درایه ۱

۲۸ = ۲ * ۱۴برخورد در درایه ۲

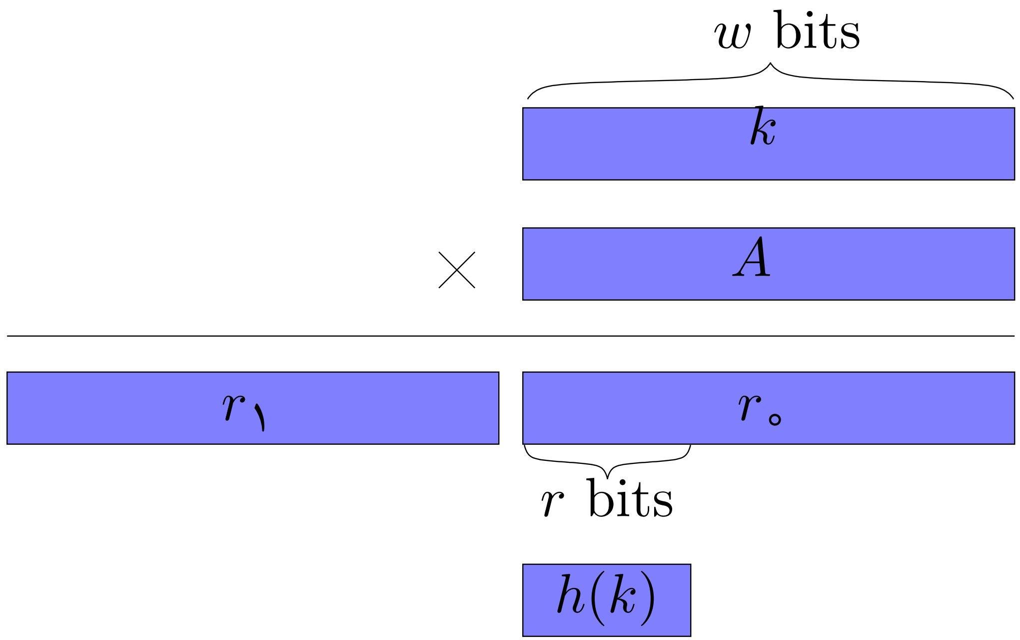
۲۹ = ۱ + ۲ * ۱۴ برخورد در درایه ۴

توابع درهم‌سازی (روش ضرب)

- فرض: کلیدها اعداد صحیح، $m = ۲^r$ و کلمه‌ی کامپیوتر w بیتی است.
- $h(k) = (A.k \bmod ۲^w) \ \mathrm{rsh} \ (w - r)$
- 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)

A	1 0 1 1 0 0 1
k	1 1 0 1 0 1 1
	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 که میانگین 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 از کلیدها انتخاب کن
  - حتا اگر حریف 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:

$$h : U * \{۰, ۱, ..., m - ۱\} \to \{۰, ۱, ..., m - ۱\}$$

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

$$< h(k, ۰), h(k, ۱), ..., h(k, m - ۱) >$$

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

<u>HASH-SEARCH</u> (T, k)

$i \leftarrow 0$
repeat $j \leftarrow h(k, i)$
        if $T[j] = k$
            then return j
        $i \leftarrow i + 1$
until $T[j] =$ **null** **or** $i = m$
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)
if HASH-SEARCH(T, k) ≠ null
   then error "item already exits"
i ← 0
repeat j ← h(k, i)
        if T[j] = null  or T[j] = "deleted"
           then T[j] ← k
                return j
           else i ← i + 1
until i = m
 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 $i = ۰, ۱, ..., m - ۱$

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:

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

که h_2, h_1 توابع درهم‌سازی کمکی هستند.

تحلیل آدرس‌دهی باز

- در آدرس‌دهی باز در هر درایه جدول حداکثر یک عنصر قرار می‌گیرد،
- $n \leq m \Longrightarrow \alpha = \frac{n}{m} \leq ۱$
- 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 به‌طور میانگین 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 جست‌وجوی ناموفق که حتماً $i - ۱$ وارسی اولش پر بوده است و پس از آن یکی از وارسی‌هایش (iام یا بیش‌تر) تهی بوده است. پس

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

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

$$
\begin{aligned}
E[X] &= \sum_{i=۱}^{\infty} i \Pr\{X = i\} \\
&= \sum_{i=۱}^{\infty} i(\Pr\{X \geq i\} - \Pr\{X \geq i + ۱\}) \\
&= \sum_{i=۱}^{\infty} \Pr\{X \geq i\} \\
&\leq \sum_{i=۱}^{\infty} \alpha^{i-۱} \\
&= \sum_{i=۰}^{\infty} \alpha^{i} \\
&= \frac{۱}{۱ - \alpha}
\end{aligned}
$$

تحلیل آدرس دهی باز (درج)

نتیجه:

درج یک عنصر دریک جدول درهم‌سازی باز در حالت میانگین حداکثر $\frac{۱}{۱-\alpha}$ وارسی نیاز دارد.

اثبات:

یک عنصر درج می‌شود اگر حداقل یک فضای خالی وجود داشته باشد. بنابراین $\alpha < ۱$. درضمن درج یک عنصر مستلزم انجام یک جست‌وجو و قرار دادن آن در اولین درایه‌ی خالی است. پس تعداد میانگین وارسی‌ها با توجه به قضیه‌ی قبل برابر $\frac{۱}{۱-\alpha}$ خواهد بود.

تحلیل آدرس دهی باز (جست‌وجوی موفق)

قضیه:

مقدار میانگین تعداد وارسی‌ها در یک جدول درهم‌سازی باز با $\alpha < ۱$ برای یک جست‌وجوی موفق حداکثر برابر است با

$$\frac{۱}{\alpha} \ln \frac{۱}{۱ - \alpha}$$

اثبات:

جست‌وجو برای k همان دنباله‌ی وارسی‌ها را دارد که اگر k را بخواهیم درج کنیم.

اگر k $(i+۱)$ امین کلیدی باشد که درج می‌شود، میانگین وارسی‌ها برابر است با:

$$۱/(۱-i/m) = m/(m-i)$$

پس

$$\begin{aligned} \frac{۱}{n}\sum_{i=۰}^{n-۱}\frac{m}{m-i} &= \frac{m}{n}\sum_{i=۰}^{n-۱}\frac{۱}{m-i} \\ &= \frac{۱}{\alpha}(H_m - H_{m-n}) \end{aligned}$$

که $H_i = \Sigma_{j=۱}^{i} ۱/j$ تابع هارمونی i ام است.

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