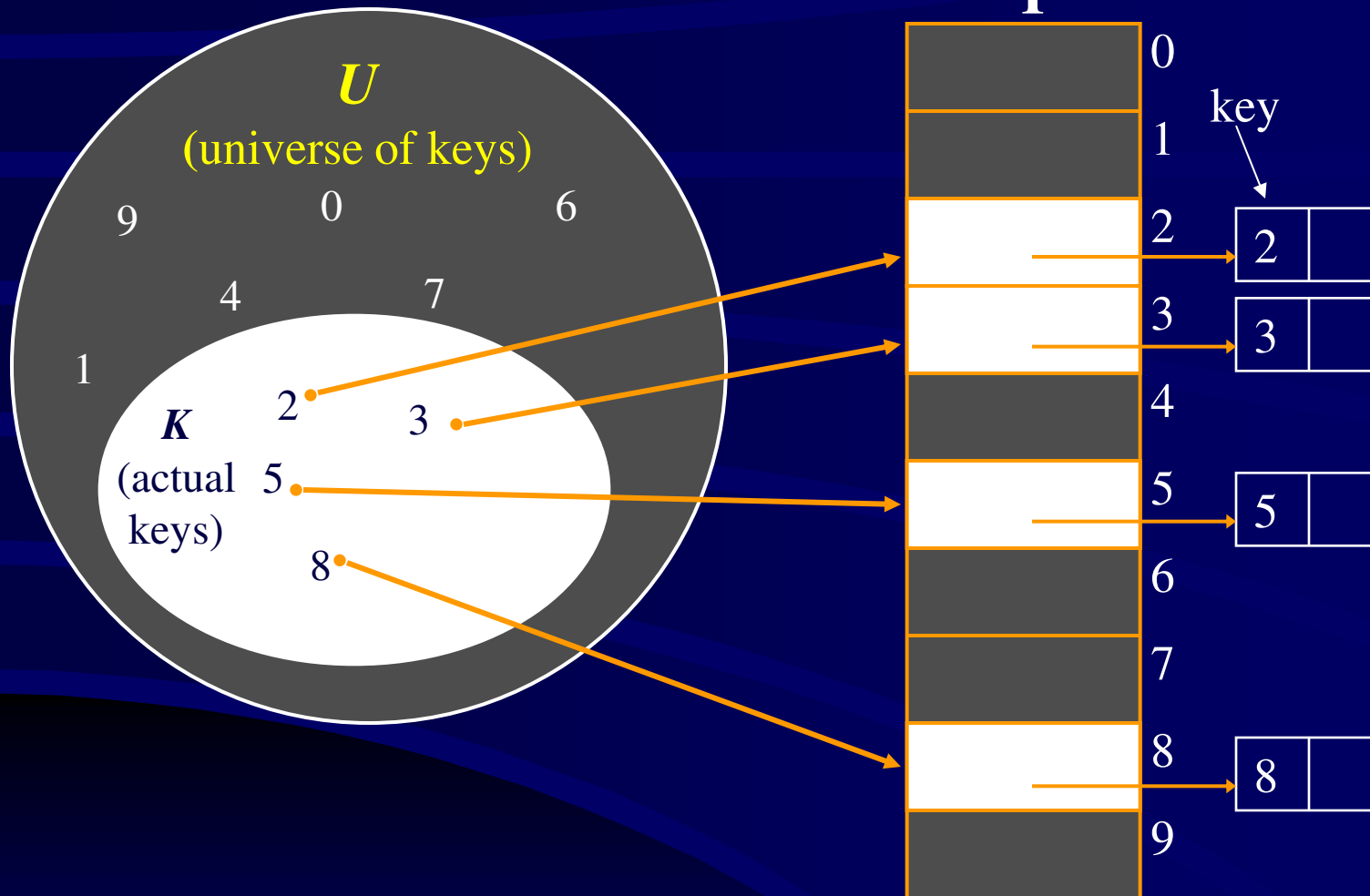


Hashing



Direct-address Table



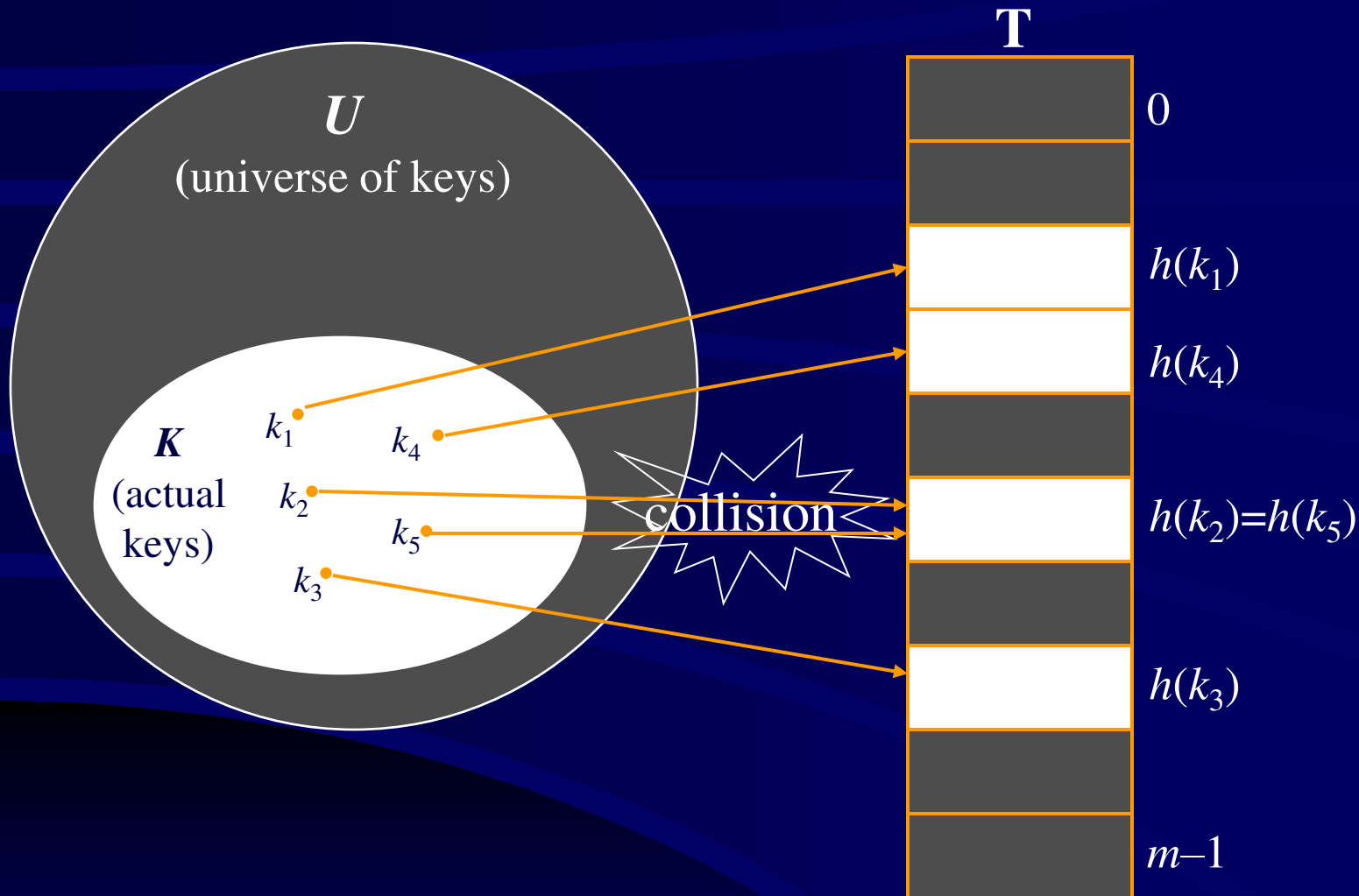
Search: $O(1)$

Insert: $O(1)$

Delete: $O(1)$

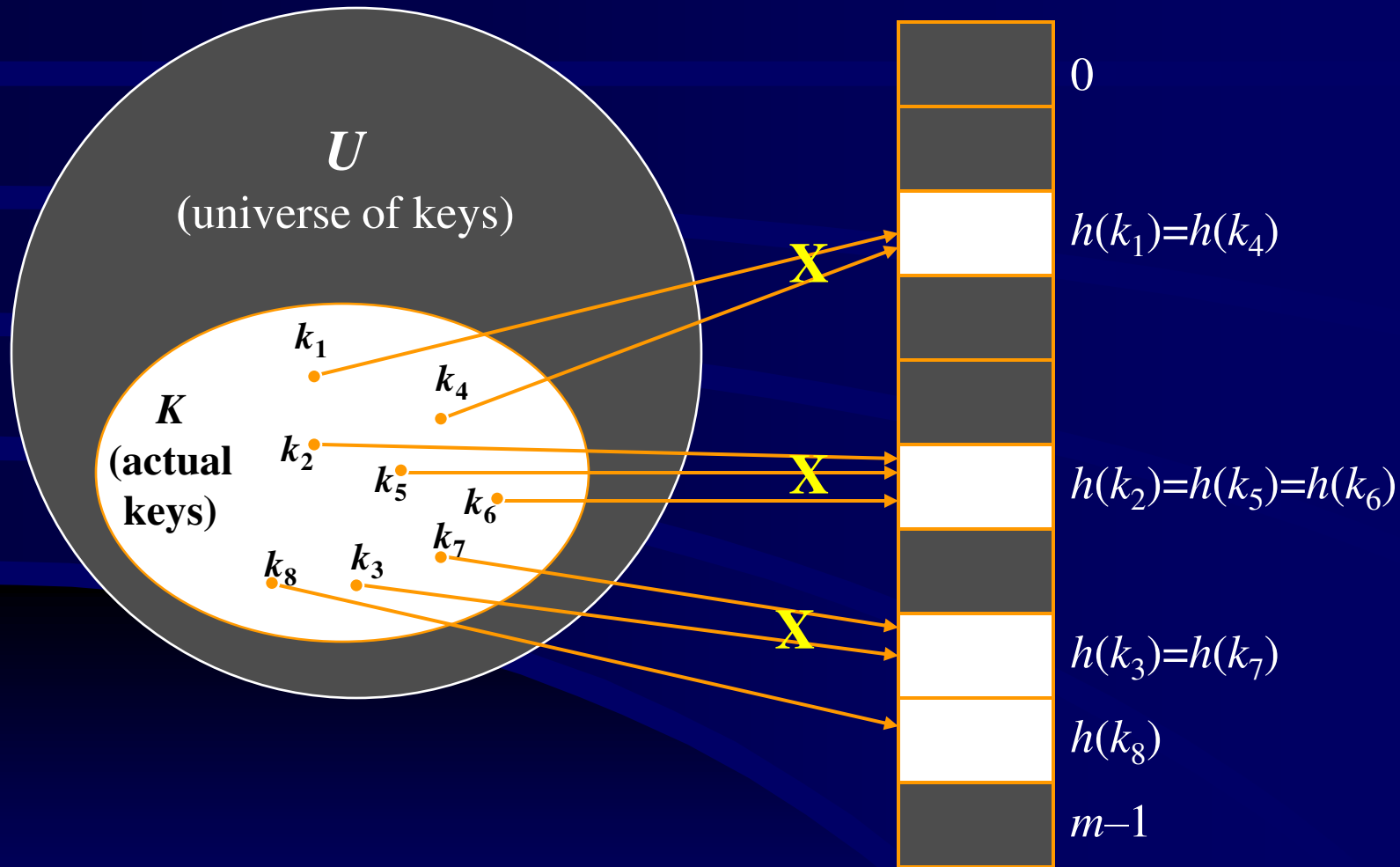
وقتي براي هر كليد از U جايي در جدول داشته باشيم ساختار مناسب است

Hash Table

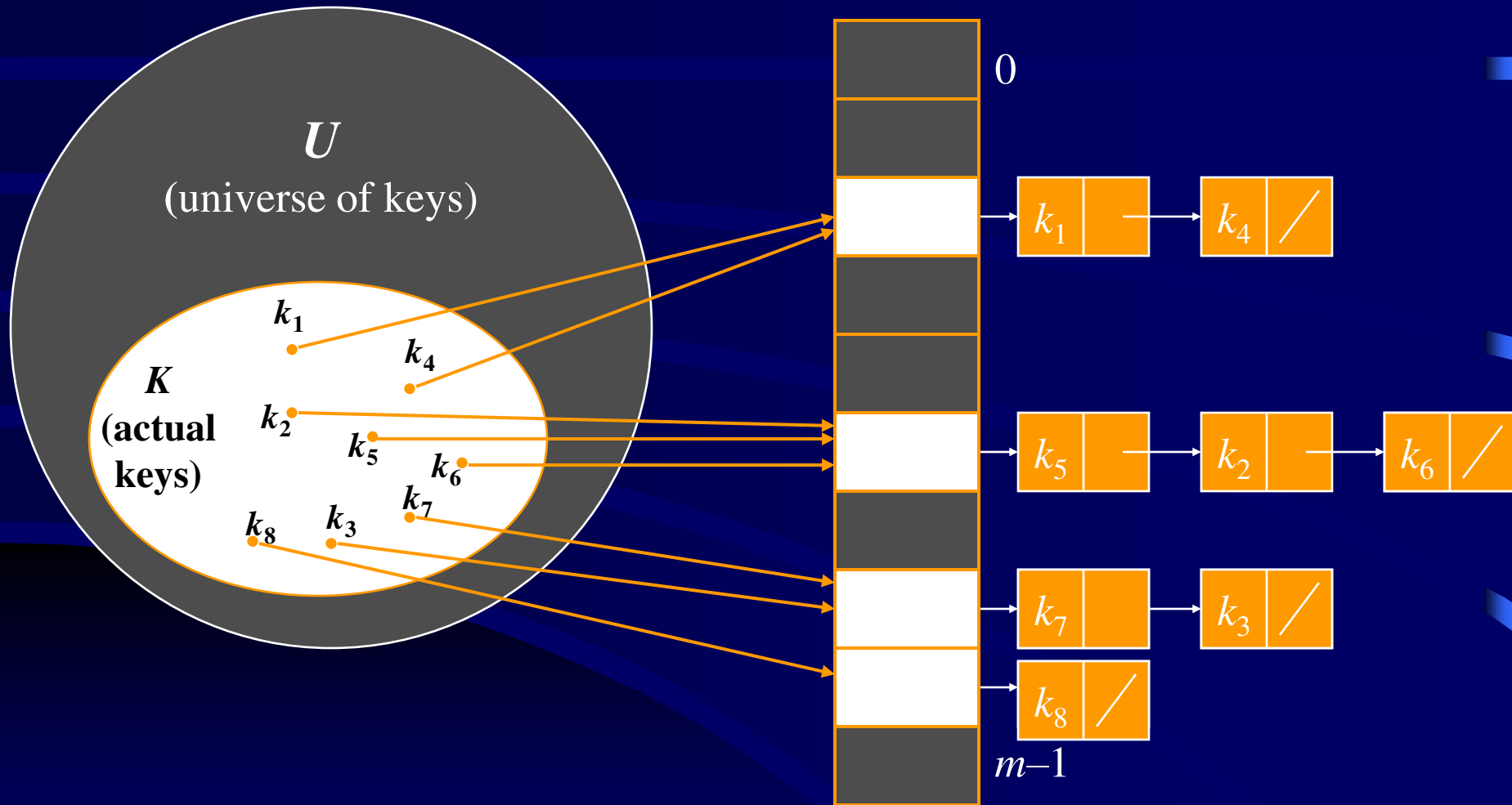


عنصر با کلید k در slot ای که با $h(k)$ تعیین می شود قرار می گیرد.
 h ، hash function است که کلیدهای U را به slot های hash table ($T[0..m-1]$) نگاشت می کند
 $h: U \rightarrow \{0, 1, \dots, m-1\}$

Collision Resolution by Chaining



Collision Resolution by Chaining



انتخاب تابع مناسب براي پراکندگي

- توزيع يکنواخت کلیدها در جدول
- « در عمل بطور کامل امکان پذیر نیست
- معمولاً توابع، با استفاده از روشهاي مکاشفه اي (heuristic)، براساس دامنه کلیدها انتخاب مي شوند.
- اغلب، توابع پراکندگي با فرض اینکه کلیدها اعداد طبيعي هستند طراحي مي شوند.
- اگر کلیدها اعداد طبيعي نیستند، بايد به گونه اي آنها را به اعداد طبيعي تبديل کرد.
- مثلاً براي CLRS به عنوان کلید:

» مقادير ASCII : C=67, L=76, R=82, S=83

» تعداد کل مقادير ASCII : 128

$$CLRS = 67 \cdot 128^3 + 76 \cdot 128^2 + 82 \cdot 128^1 + 83 \cdot 128^0 =$$

141 764 947

روش تقسیم

- تقسیم صحیح کلید k بر m (تعداد slot ها) و استفاده از باقیمانده تقسیم برای نگاشت k به $(0..m-1)$

$$h(k) = k \bmod m$$

- مثال: $m=10, k=22 \Rightarrow h(k) = 2$

$$m = 31, k = 78 \Rightarrow h(k) = 16$$

- روش سریعی است، فقط با يك عمل تقسیم انجام می شود
- مقدار m ؟

روش تقسیم

- تقسیم صحیح کلید k بر m (تعداد slot ها) و استفاده از باقیمانده تقسیم برای نگاشت k به $(0..m-1)$

$$h(k) = k \bmod m$$

- مثال: $m=10, k=22 \Rightarrow h(k) = 2$

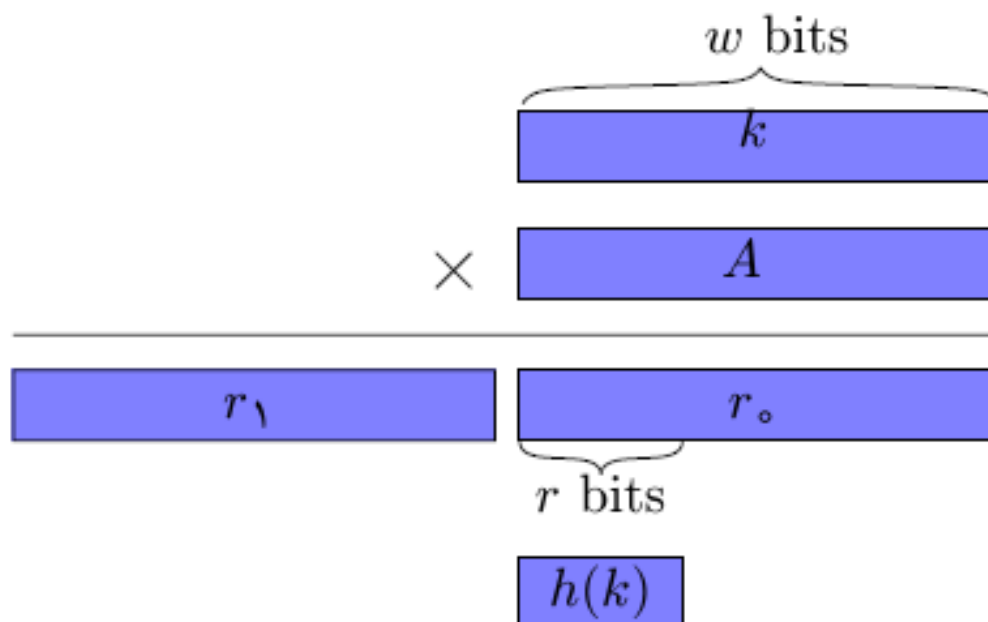
$$m = 31, k = 78 \Rightarrow h(k) = 16$$

- روش سریعی است، فقط با یک عمل تقسیم انجام می شود
- اعداد اولی که نزدیک به توانی از 2 نیستند مقادیر مناسبی برای m هستند

توابع درهم‌سازی (روش ضرب)

- فرض: کلیدها اعداد صحیح، $m = 2^r$ و کلمه‌ی کامپیوتر w بیتی است.
- $$h(k) = (A.k \bmod 2^w) \text{ rsh } (w - r)$$
- rsh یعنی شیفت راست بیتی
- A یک عدد صحیح فرد است به‌طوری‌که $2^{w-1} < A < 2^w$.
- توصیه‌ها و نکات:
 - A نزدیک به 2^w نباشد.
 - عمل ضرب در $2^w \bmod$ سریع است.
 - عمل rsh هم سریع است.

توابع درهم سازی به روش ضرب (مثال)



توابع درهم سازی 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

رفع برخورد با open addressing

- برای تمام کلیدها، عناصر در خود جدول hash نگهداری می شود
- هر slot حاوی یک کلید یا NIL (یا Deleted) است

الگوریتمهای اضافه کردن و جستجو کردن

Hash-Search (T, k)

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

Hash-Insert(T, k)

1. $i \leftarrow 0$
2. **repeat** $j \leftarrow h(k, i)$
3. **if** $T[j] = \text{NIL}$
4. **then** $T[j] \leftarrow k$
5. **return** j
6. **else** $i \leftarrow i + 1$
7. **until** $i = m$
8. **error** “hash table overflow”

حذف عنصر

- نمی توان به راحتی کلید مورد نظر را به NIL تغییر داد
- در Slot ای که باید حذف شود، مقدار خاصی به عنوان DELETED قرار می گیرد.
- در الگوریتم Search این مقدار به عنوان کلیدی که مساوی کلید مورد جستجو نیست تفسیر می شود
- در الگوریتم Insert این مقدار به عنوان slot خالی تفسیر می شود

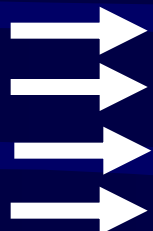
Linear probing

$$h(k, i) = (h'(k) + i) \bmod m. \quad i=0,1,\dots,m-1$$

- ترتیب slot‌هایی که مورد بررسی قرار می‌گیرند
- : $T[h'(k)], T[h'(k)+1], \dots, T[m-1], T[0], T[1], \dots, T[h'(k)-1]$
- پیاده‌سازی این روش ساده است
- **primary clustering** : ممکن است تعداد زیادی از کلیدها در یک محدوده از جدول قرار گیرند و زمان جستجو و اضافه کردن افزایش می‌یابد (به جستجوی خطی نزدیک می‌شود)

Linear Probing (insert 12)

$$12 = 1 \times 11 + 1$$
$$12 \bmod 11 = 1$$



0	42
1	1
2	24
3	14
4	
5	16
6	28
7	7
8	
9	31
10	9

0	42
1	1
2	24
3	14
4	12
5	16
6	28
7	7
8	
9	31
10	9

Quadratic probing

- $$h(k,i) = (h'(k) + c_1i + c_2i^2) \bmod m \quad c_2 \neq 0 \quad i=0,1,..m-1$$

- با انتخاب درست c_1 ، c_2 و m باید مطمئن بود که تابع پراکندگی به ترتیب اندیس تمام slot ها را تولید می کند

- **secondary clustering** : اگر $h(k_1)=h(k_2)$ ، برای k_1 و k_2 ترتیب یکسانی از slot های جدول hash مورد بررسی قرار می گیرند.

- مثال:
$$h(k,i) = (h'(k) + i^2) \bmod m$$

- $m=5$ اگر $h'(k)=0$ ترتیبی که تولید می شود:

Quadratic probing

- $$h(k,i) = (h'(k) + c_1i + c_2i^2) \bmod m \quad c_2 \neq 0 \quad i=0,1,..m-1$$

- با انتخاب درست c_1 ، c_2 و m باید مطمئن بود که تابع پراکندگی به ترتیب اندیس تمام slot ها را تولید می کند

- **secondary clustering** : اگر $h(k_1)=h(k_2)$ ، برای k_1 و k_2 ترتیب یکسانی از slot های جدول hash مورد بررسی قرار می گیرند.

- مثال:
$$h(k,i) = (h'(k) + i^2) \bmod m$$

- $m=5$ اگر $h'(k)=0$ ترتیبی که تولید می شود: $0,1, 4,4,1$