فصل نهم

درهم سازی

مدرس:
فرشید شیرافکن

دانشجوی دکتری دانشگاه تهران
(کارشناسی و کارشناسی ارشد: کامپیوتر نرم افزار) (دکتری: بیو انفورماتیک)

جدول آدرس دهی مستقیم (direct address)

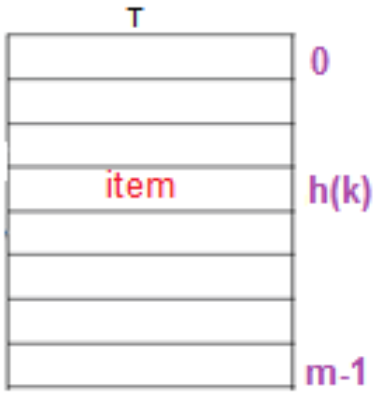
جدول آدرس دهی مستقیم به صورت T[0..m-1] تعریف می شود.

عنصر item با کلید k در خانه T(k) ذخیره می شود.

مثال: اگر کلید عنصر ۳۰ برابر ۲ باشد، (key[30]=2) ، آنگاه T(2)=30 .

مشکل آدرس دهی مستقیم : اگر m بسیار بزرگ باشد، ساخت جدول ناممکن است.
برای حل این مشکل از جدول درهم سازی استفاده می کنیم.

جدول درهم سازی



T : جدول درهم سازی به اندازه m، که n عنصر را در می توان در آن ذخیره کرد.

عنصر item با کلید k در درایه T(h(k)) ذخیره می شود.

h : تابع درهم سازی (hash function)

یک تابع درهم ساز یک کلید را به عنوان ورودی گرفته و یک آدرس بین 0 و m-1 بر می گرداند.

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

ضریب بارگذاری (load facor) : $\alpha = \frac{n}{m}$

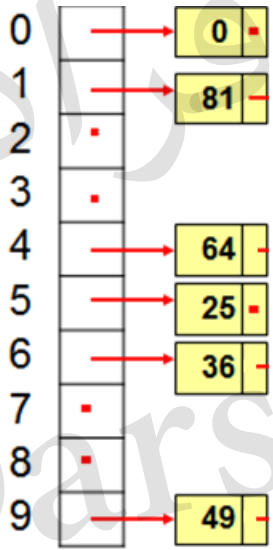
مثال

Insert Keys:

0, 25, 36, 49, 64, 81

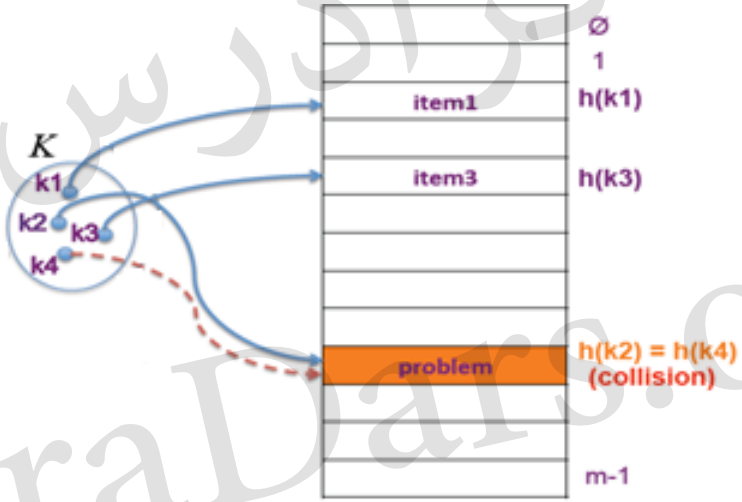
hash function :

h(k) = k mod 10



برخورد (collision)

برخورد : در جدول درهم سازی، ممکن است بیش از یک عنصر به یک درایه خاص نگاشته شود. به عبارتی، به ازای دو کلید متفاوت، آدرس یکسانی تولید شود.



نمی توان تابع درهم سازی انتخاب کرد که اصلا برخوردی نداشته باشد.

روش های حل مشکل برخورد

۱- روش زنجیره ای

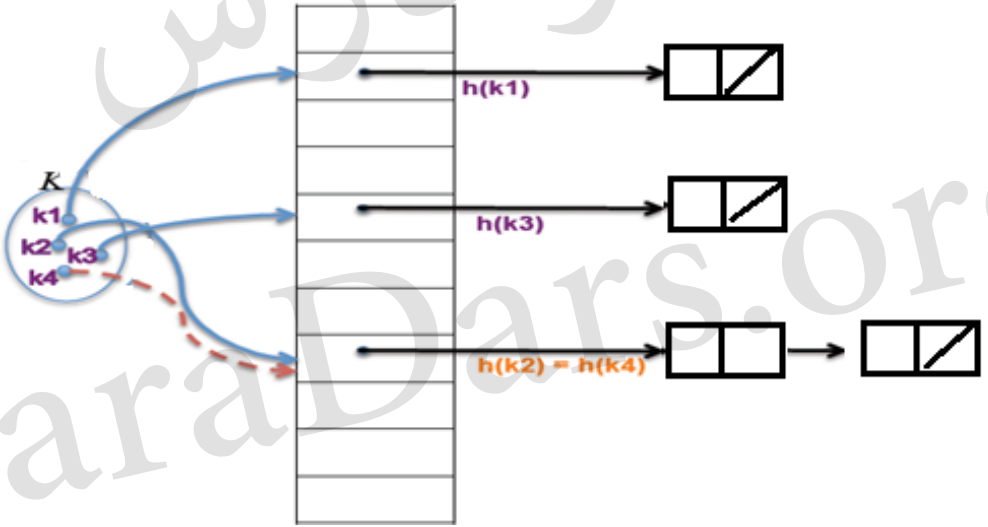
۲- روش آدرس دهی باز

الف: حداکثر تعداد عناصر را از قبل بدانیم.

ب : حداکثر تعداد عناصر را از قبل ندانیم. (جدول درهم سازی **پویا**)

روش زنجیره ای برای حل برخورد

در روش زنجیره ای (chaining)، رکوردهای تصادفی به یکدیگر زنجیر می‌شوند.
عناصر با کلیدهای مختلف k که مقدار h(k) آنها برابر است در لیست پیوندی T[h(k)] قرار می گیرند.



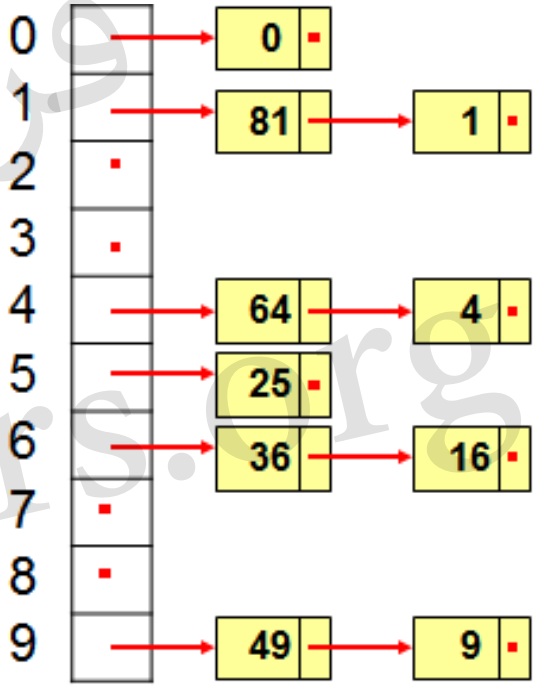
مثال

Insert Keys:

0, 1, 4, 9, 16, 25, 36, 49, 64, 81

hash function :

$h(k) = k \bmod 10$



زمان اجرای عملیات «درج، حذف و جستجو» در روش زنجیره ای

بدترین حالت زمان اجرا:

درج : O(1)

جستجو : متناسب با طول لیست است.

حذف : متناسب با طول لیست است. چون اول باید با انجام یک جستجو،عنصر مورد نظر را پیدا کرد.

Insert(T,x) : Insert **x** at the head of list **T[h(k)]**

Search(T,k) : Search for an element with key **k** in list **T[h(k)]**

Delete(T,x) : Delete **x** from the list **T[h(k)]**

روش آدرس دهی باز برای حل مشکل برخورد

در آدرس دهی باز، از یک آرایه به اندازه m به عنوان جدول درهم سازی استفاده می شود که حداکثر m عنصر در آن جای می گیرد و در هر درایه، فقط یک عنصر ذخیره می شود.

با شروع از محل تصادف، جستجوی خطی به سمت انتهای فایل شروع شده و رکورد تصادفی در اولین محل جادار درج می‌شود. وارسی به روش **حلقوی** انجام می گیرد.

حداکثر تعداد عناصر را باید از قبل بدانیم یا از جدول درهم سازی **پویا** (dynamic hash table) استفاده کنیم.

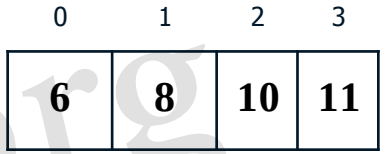
جدول درهم سازی پویا، جدولی است که اندازه آن بر حسب نیاز، کم یا زیاد می شود. (معمولا نصف یا دو برابر)

مثال

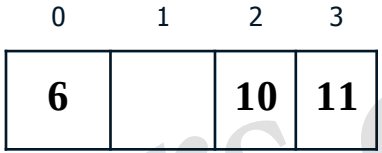
درج عناصر 8و 6 و 10 و 11 در جدول درهم سازی H[0..3] :

تابع Hash : باقیمانده تقسیم بر 4

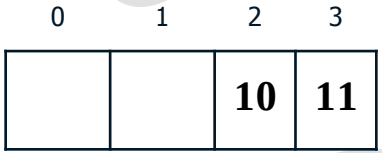
8 % 4 = 0



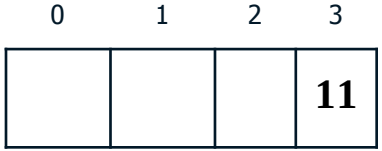
6 % 4 = 2



10 % 4 = 2



11 % 4 = 3

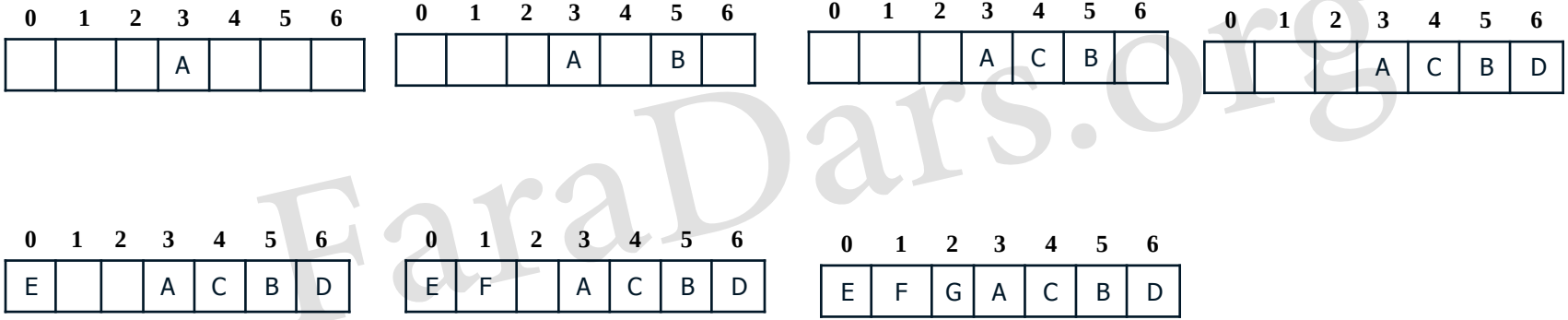


مثال

درج {A , C , B , D , E , F , G} در جدول درهم سازی H[0..6]

خروجی تابع درهم سازی:

key	A	B	C	D	E	F	G
hash	3	5	3	4	5	6	3



مثال

تعداد وارسی های مورد نیاز(شامل وارسی هایی که موجب درج می شود) برای درج هر یک از عناصر در زیر آن نوشته شده است.

E	F	G	A	C	B	D
3	3	7	1	2	1	3

مجموع تعداد وارسی ها در هر ترتیب مورد قبولی نیز برابر 20 می باشد.

درهم سازی پویا

در بیشتر کاربردها، تعداد عناصر از قبل مشخص نمی باشند و با اعمال درج و حذف در هر لحظه، تغییر می کند.
در روش درهم سازی پویا (dynamic hashing) اندازه جدول درهم سازی بر حسب نیاز کم و زیاد می شود.
در این جدول، علاوه بر درج و حذف ساده، دو عمل زیر نیز قابل انجام است:

درج با گسترش	**قبل از درج، جدول پر است و اندازه جدول دو برابر می شود و عناصر موجود به جدول جدید منتقل شده و سپس عنصر مورد نظر در جدول جدید درج می شود.**
حذف و فشرده سازی	**بعد از حذف، نسبت تعداد عناصر به اندازه جدول کم می شود. بنابراین اندازه جدول را نصف کرده و همه عناصر موجود به جدول جدید منتقل می شود.**

مثال

یک جدول درهم سازی را در نظر بگیرید که اندازه ی آن در ابتدا 1 است و آن درایه هم خالی است. هزینه کل (تعداد نوشتن در جدول) درج ۶ عنصر در جدول را مشخص کنید.

عنصر اول را درج می کنیم. درج عنصر دوم : چون جدول جا ندارد، اندازه جدول دو برابر شده و سپس عنصر جدید را درج کرده و عنصر موجود در جدول قبل را به جدول جدید منتقل می کنیم.

درج عنصر سوم: چون جدول جا ندارد، اندازه جدول دو برابر شده و دو عنصر قبلی با هزینه 2 به جدول جدید منتقل شده و عنصر جدید نیز درج می شود.

درج عنصر چهارم : نیازی به دو برابر کردن اندازه جدول نیست، چون یک جای خالی در جدول موجود است.

درج عنصر پنجم : چون جدول جا ندارد، اندازه جدول دو برابر شده و چهار عنصر موجود در جدول قبل به جدول جدید منتقل شده و عنصر پنجم درج می شود. و....

پاسخ

جدول :

i	1	2	3	4	5	6
اندازه جدول	1	2	4	4	8	8
هزینه	1	1+1	1+2	1	1+4	1

$$6 + (1 + 2 + 4) = 13$$

هزینه کل نوشتن n عدد :

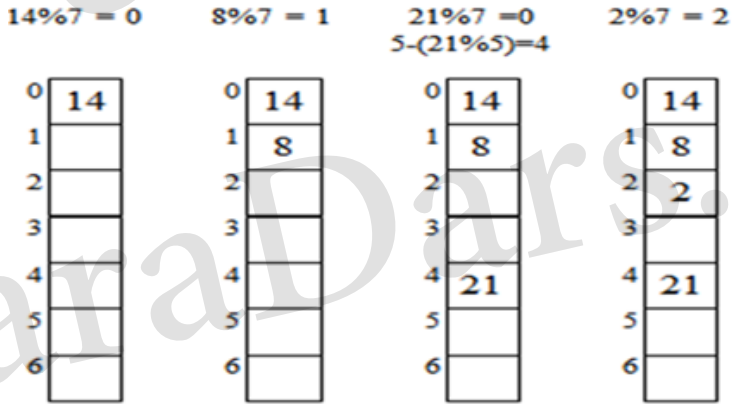
$$n + \sum_{k=0}^{\log n} 2^k \approx 3n - 1$$

مثال double hashing

$h(k) = k \bmod 7$

$d(k) = 5 - k \bmod 5$

Insert keys : 14 , 8 , 21 , 2



مطالب ساختمان داده پیشرفته

۱- مجموعه های مجزا
۲- درخت دودویی جست و جوی بهینه
۳- درخت‌های دودویی جست و جو با ارتفاع لگاریتمی
(قرمز-سیاه ، مرتبه آماری ، AVL)
۴- درخت ۳-۲
۵- درخت بی (B-tree)
۶- تحلیل سرشکن

این اسلاید ها بر مبنای نکات مطرح شده در فرادرس
«مجموعه فرادرس های ساختمان داده ها»
تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید.

faradars.org/fvds9402