مقدمه ای بر:

توابع درهم سازی

Hash functions

مهدیار افتخاری نیا

.اصطلاح هشینگ (hashing) : این کلمه در لغت به معنی آمیزش چند چیز یا مخلوط کردن چند چیز مختلف است.به علاوه یکی از معانی هشینگ درهم سازی نیز میباشد.

.کاربرد درهم سازی در علوم کامپیوتر:
درهم سازی یا هشینگ در علوم کامپیوتر کاربرد های فراوانی دارد از جمله : شبکه های کامپیوتری وتئوری کدینگ و بررسی صحت داده ها و بهینه سازی حافظه مورد استفاده و... . همچنین این نوع داده ساختار دارای پیچیدگی محاسباتی بسیار پایینی هستند و اثبات شده که بسیار کارا است و به خوبی جواب میدهد.

مثال:
برای آنکه مطمئن شویم اطلاعات به درستی از دست فردی به فرد دیگری انتقال داده شده (دستکاری نشده) از توابع هش استفاده میشود.دلیل آن است که این توابع طول خروجی بسیار کوتاه تری نسبت به ورودی دارند و به درستی اطلاعات ورودی را به خروجی یکتا ارجاع میدهند.

.دلیل اصلی استفاده از توابع هش چیست؟

فرض کنید ما آرایه ای به طول مجموعه اعداد طبیعی داریم ← $N=\{1,2,3,4,5,6,.....\}$

میخواهیم یک سری از داده ها را در خانه های آرایه وارد کنیم که اندیس های آن زیر مجموعه از از مجموعه اعداد طبیعی است
.یک را حل برای آنکه مطمئن شویم تعداد خانه ها کمتر از داده های ما نباشد و داده دقیقا در همان خانه ای قرار میگیرد
که کلید آن اشاره میکند آن است که به ازای هر عضو از مجموعه اعداد طبیعی خانه ای با همان اندیس در آرایه داشته باشیم.

طبیعتا اگر بخواهیم به ازای هر عضو از مجموعه اعداد طبیعی خانه ای از آرایه داشته باشیم که اندیس آن عضوی از مجموعه اعداد طبیعی باشد
این کار امکان ناپذیر است (به دلیل کمبود خانه ها و یا عناصر حافظه).(تصویر ۱)

پس نمیتوانیم به ازای مجموعه بسیار بزرگی از کلید ها آرایه ای با همان اندازه ایجاد کنیم
و مجبوریم راهکار دیگری را در پیش بگیریم . در اینجا عمل هشینگ و توابع هش مختلف وارد کار میشوند.

توجه: در اسلاید بعد تعاریفی با توجه به این تعریف خواهیم داشت که دقیقا به همین موارد اشاره میکند منتها به اسمی متفاوت.

key	data
0	-
1	-
2	-
3	-
4	-
5	-
...	

تصویر ۱

.تعاریف اولیه:

کلید(key):
منظور ما از کلید در حقیقت پارامتر ورودی میباشد که با دادن آن به تابع هش میتوان به اندیس خانه ارایه رسید.
(این تابع میتواند کلید را مستقیما به اندیس آرایه تبدیل نماید که در این صورت نام آن دیگر هش نیست).

مجموعه جامع کلید ها (Universe):
مجموعه ای است که دارای تمام کلید هایی است که ما میتوانیم اتخاذ کنیم که به شکل U نمایش داده میشود.

مجموعه کلید های حقیقی (actual keys):
مجموعه ای از کلید هایی است که از کلید های مجموعه جامع ایجاد و انتخاب شده و قصداستفاده از آن ها را داریم.
(اصطلاحا میخواهیم آن ها را نگاشت کنیم)

جدول هش (hash table): در حقیقت همان آرایه ای است که میخواهیم اطلاعاتمان را درون آن قرار دهیم.

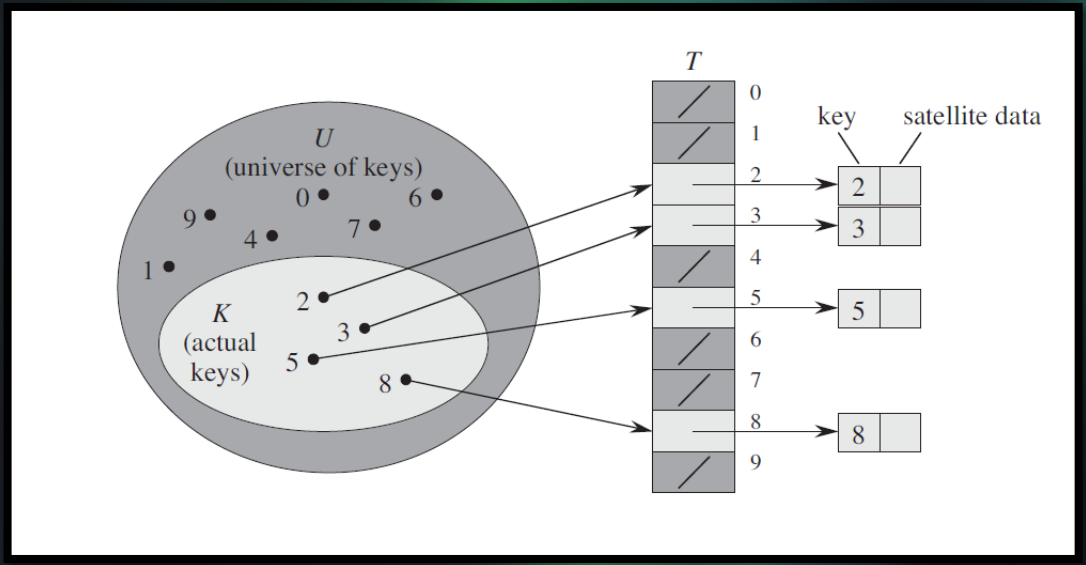
روش direct access table(direct addresing):
این روش دقیقا همانند تعاریف صفحه قبل است و داده با همان کلید مستقیما به خانه با کلید یکسان نگاشت میشود.
مشکل این روش آن است که به ازای مجموعه ای بسیار بزرگ حافظه بسیاری را مصرف میکند و ممکن است امکان ناپذیر باشد
(تصویر ۲)

پیچیدگی زمانی این روش:

پیچیدگی زمانی به ازای جستجو:O(1)

پیچیدگی زمانی به ازای درج یک عنصر: O(1)

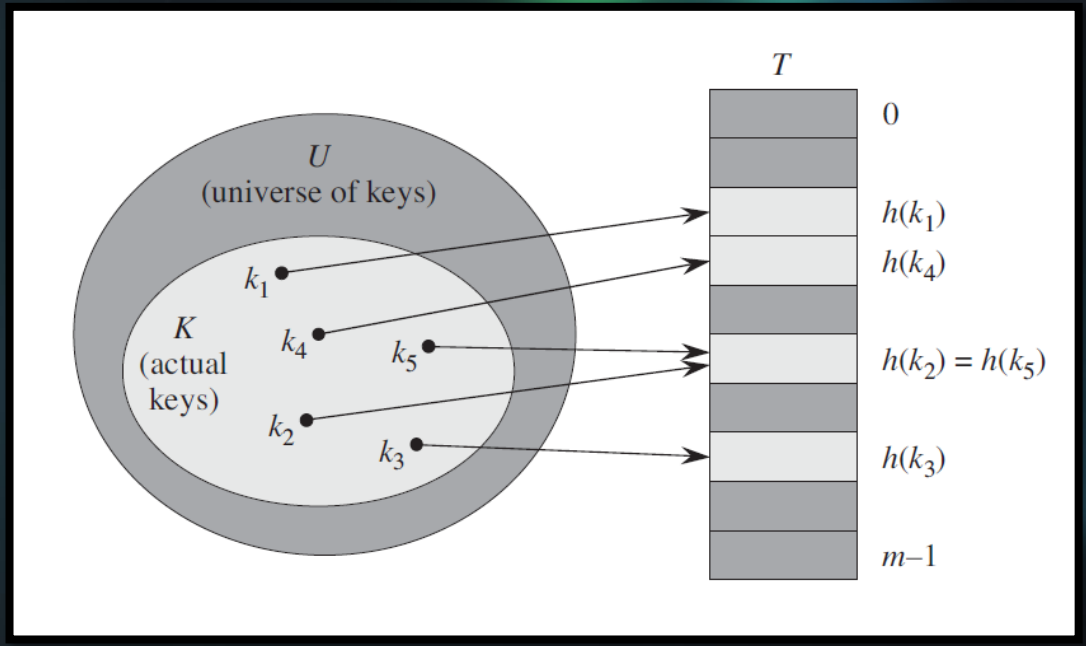
پیچیدگی زمانی به ازای حذف یک عنصر:O(1)



تصویر ۲

.توابع درهم ساز(هش):

تا کنون به شیوه مستقیم آدرس دهی را انجام میدادیم اما میتوانیم با تابعی که ما را به یک خانه از آرایه نگاشت میکند کار کنیم .
ایده به این صورت است که ما کلید هایمان را درون تابعی قرار میدهیم که مارا به یکی از خانه های آرایه نگاشت میکند.



آرایه: T[0,1,...,m-1]

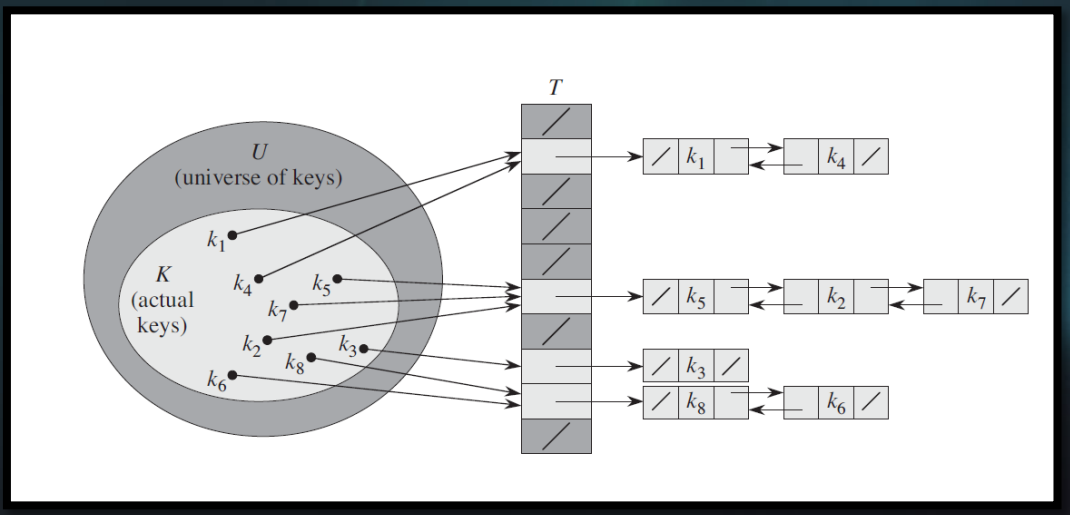
تابع: h(k) | kϵU {0,1,...,m-1}

.برخورد یا (collision):

هنگام درج عناصر در آرایه توسط تابع درهمساز این امکان وجود داره که دو کلید به خانه ای یکسان نگاشت شوند.
به این مشکل برخورد گفته میشود که در نوع خود بسیار مشکل بزرگی است. برای حل این مشکل روش های متعددی وجود دارد که به آن خواهیم پرداخت البته این روش ها پیچیدگی زمانی را بالا خواهند برد
ولی در هر صورت مشکل را حل خواهند کرد. در ضمن هرچقدر تابع به کار رفته قابلیت تصادفی سازی بیشتری داشته باشد برخورد کمتر پیش خواهد آمد ولی میدانیم که تقریبا هیچ تابع هشی وجود ندارد که دارای collision نباشد.

.حل مشکل برخورد به روش زنجیره سازی (CHAINING):

در این روش خانه های آرایه هر کدام یک اشاره گر first از یک لیست پیوندی است که در صورت ایجاد برخورد به اولین گره از لیست پیوندی همان خانه نگاشت میشود.و در صورتی که هیچ کلیدی به یکی از خانه های آرایه نگاشت نشود مقدار آن خانه برابر با NIL است. تصویر (۳)



پیچیدگی زمانی این روش:

پیچیدگی زمانی به ازای جستجو:O(1)

پیچیدگی زمانی به ازای درج یک عنصر: $O(N),\theta(1+a)$

پیچیدگی زمانی به ازای حذف یک عنصر:$O(N),\theta(1+a)$

برای درک بهتر موضوع در اسلاید های بعد توضیحات تکمیلی را خواهیم داد

. تابع درهم سازه ساده یکنواخت:

در اینجا میخواهیم میزان توزیع پذیری و یکنواختی تابع درهم ساز را بررسی کنیم.منظور از توزیع پذیری این است که برای یک خانه از جدول درهم ساز چند کلید به آن نگاشت میشود.در اسلاید قبلی روش زنجیره سازی را معرفی کردیم و گفتیم در بدترین حالت برای جستجو یک عنصر در یک خانه باید به اندازه N خانه جستجو کنیم
که هم به ازای جستجو موفق و هم به ازای جستجو ناموفق برقرار است. اما دلیل آن چیست؟ این موضوع به یکنواختی کلید ها اشاره دارد. فرض کنید ما تعداد N کلید برای درج داریم و قرار است همه این کلید ها به یک خانه نگاشت شود قطعا این تابع هش مناسبی نیست زیرا یکنواختی ندارد و قرار است همه کلید ها را به یک خانه نگاشت کند در نتیجه پیچیدگی برای جستجو در این تابع O(N) است زیرا قرار است N کلید را در یک خانه مورد جستجو قرار بدهد.

در اینجا چه تابعی مناسب است؟
در اینجا تابع درهم سازی مناسب است که یکنواختی را تا حد ممکن حفظ کند.ما در اینجا(در روش زنجیره سازی) فرض را بر این میگذاریم که تابع ما یک تابع درهم ساز یکنواخت است. در اینجا با فرض موجود کلید ها هر یک با احتمال یکسانی به خانه های جدول درهم ساز نگاشت میشوند و به عبارت دیگر احتمال اینکه دو کلید متفاوت به یک خانه نگاشت شوند یکسان است .پس:

احتمال برخورد دو کلید متفاوت : $\rho\big(h(a) = h(b)\big) = \frac{1}{m}$

احتمال برخورد کلید ها به طور متوسط (load factor): $a = \frac{n}{m}$

احتمال تعداد جستجو موفق و ناموفق به طور متوسط به ازای یک عنصر:$\theta(1 + a)$

.توابع درهم ساز:

چه تابع درهم سازی خوب است؟

یک تابع خوب درهم سازی فرض یکنواخت بودن توزیع کلید ها را تحقق میبخشد.بررسی اینکه آیا تابع مورد نظر ما دقیقا توزیع کلید ها را به صورت یکنواخت انجام تقریبا غیر ممکن است و فقط میتوانیم به صورت احتمالاتی با آن برخورد کنیم و احتمال هر یک از عملیات ها را بدست آوریم.

.متد های توابع درهم سازی:

۱ –روش تقسیم:

این روش در توابع درهم سازی بسیار مرسوم است زیرا بسیار ساده است.در این روش ما کلید را به یکی از m خانه جدول هش نگاشت میکنیم با باقیمانده تقسیم کلید بر m.پس:

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

برای مثال اگر k=100 و m=12 حاصل به خانه ۴ نگاشت میشود.

زمانی که از روش تقسیم استفاده میکنیم باید در نظر داشته باشیم که مقدار m نباید توانی از عدد دو باشد و.
و حد الامکان باید از اعداد که توانی از دو هستند دور باشند.دلیل این است که زمانی که ما عددی را انتخاب میکنیم که توانی از دو است یا نزدیک به آن است بیت هایی باارزش بالا در نظر گرفته نمیشوند و بیت هایی با ارزش پایین تر در نظر گرفته میشود (low order bits)

.مثال زیر را را درنظر بگیرید:

اگر m را 2^6 در نظر بگیریم:

```
k:      1111111111010101
m:      0000000001000000   (2^6)
H(k)= 0000000000010101
```

و اگر عدد m را دوباره برای کلیدی استفاده کنیم که بیت های کم ارزش کوچکتر از m آن از کلید قبلی بر گرفته شده باشد شاهد جواب یکسان خواهیم بود:

```
k:      0000000001010101
m:      0000000001000000   (2^6)
H(k)= 0000000000010101
```

پس دیدیم که بیت های با ارزش کلید نادیده در نظر گرفته میشود و امکان برخورد را بالا میبرد.در حالت کلی اگر کلید های ما رندوم باشد و از قاعده خاصی پیروی نکند نیازی به استفاده از اعداد اول نیست و میتوان عدد m را هر عددی در نظر گرفت ولی امکان اینکه کلید ها رندوم باشند بسیار پایین و حتی ناممکن است.

۲-روش ضرب:

این روش دارای دو مرحله است.در مرحله اول کلید (k) را در یک ثابت به نام A ضرب میکنیم که 0<A<1 و قسمت اعشاری KA را استخراج میکنیم.در نهایت قسمت اعشاری را در m ضرب میکنیم و کران پایین جز صحیح ان را محاسبه میکنیم.

$$h(k) = \lfloor m(kA \bmod 1) \rfloor$$

به طوری که $kA \bmod 1$ برابر است با قسمت کسری kA که برابر میشود با :$kA - \lfloor kA \rfloor$

یکی از مزایای این روش این است که نیازی به انتخاب یک m خاص نیست و میتوان از هر عددی برای استفاده به جای m استفاده نمود.