Hello!

I am Esmaeil Kazemi

I'm interested in learning how are you?

You can find me at @eskazemi

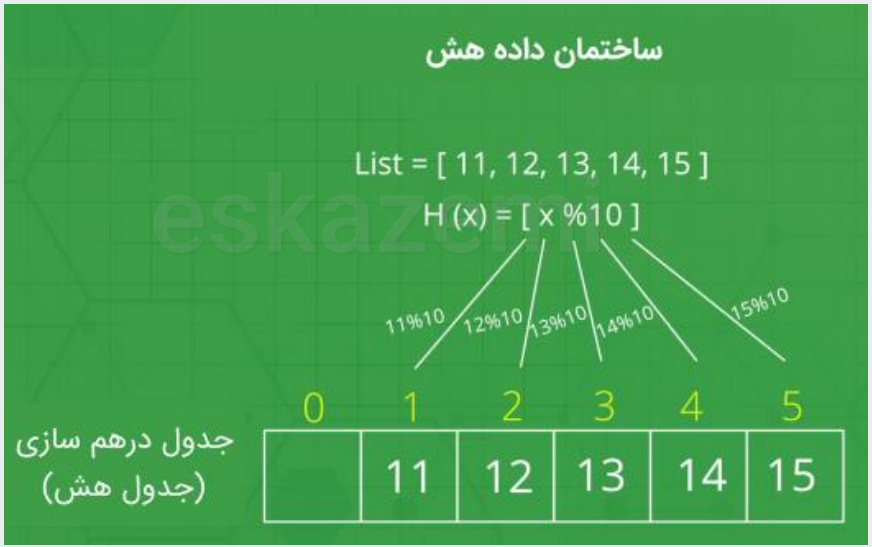
NOSQL

vs

SQL

eskazemi

Hash Table



هشینگ به فرآیند تولید یک خروجی با اندازه ثابت از یک ورودی با اندازه متغیر با استفاده از فرمول های ریاضی معروف به توابع هش (hash functions) اشاره دارد. این تکنیک یک اندیس یا مکان را برای ذخیره سازی یک آیتم در یک ساختار داده تعیین می کند.

Hash Table

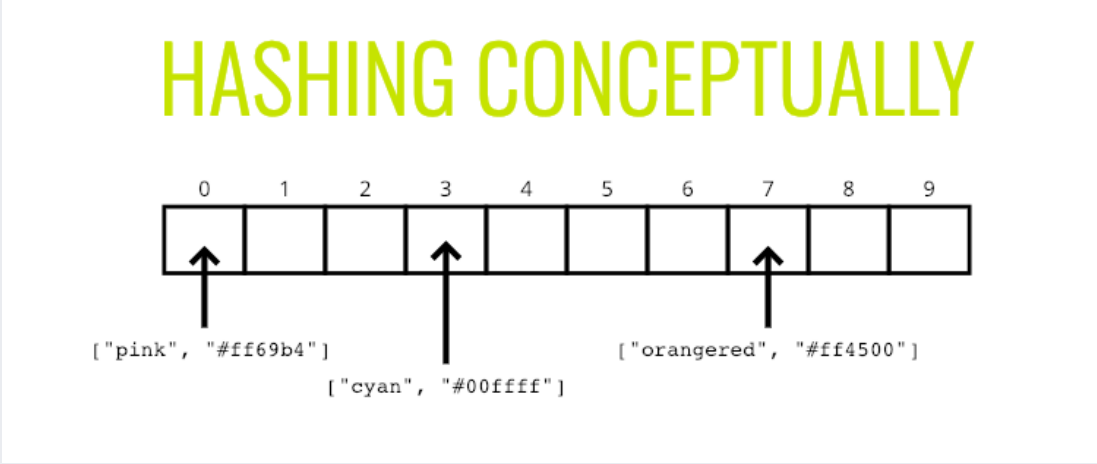
چه نیاز به ساختار داده ای Hash داریم؟

هر روز، داده های موجود در اینترنت چند برابر می شوند و همیشه ذخیره سازی موثر این داده ها کار دشواری است. در برنامه نویسی روزمره، ممکن است این مقدار داده چندان بزرگ نباشد، اما با این حال، باید به راحتی و به طور موثر ذخیره، دسترسی و پردازش شود. یک ساختار داده بسیار رایج که برای چنین هدفی استفاده می شود، ساختار داده Arrayاست.

حال این سوال پیش می آید که اگر Array از قبل وجود داشت، نیاز به یک ساختار داده جدید چه بود؟ پاسخ این سوال در کلمه "کارایی(efficiency)" است. اگرچه ذخیره سازی در Arrayزمان O(1) می برد، اما جستجو در آن حداقل O (log n) زمان می برد. این زمان به نظر کوچک می رسد، اما برای یک مجموعه داده بزرگ، می تواند مشکلات زیادی ایجاد کند و این به نوبه خود، ساختار داده Array را ناکارآمد می کند.

بنابراین اکنون ما به دنبال یک ساختار داده ای هستیم که بتواند داده ها را ذخیره کند و در زمان ثابت، یعنی در زمان O(1) جستجو کند. به این ترتیب ساختار داده هشینگ وارد بازی شد. با معرفی ساختار داده های Hash، اکنون امکان ذخیره سازی آسان داده ها در زمان ثابت و بازیابی آن ها در زمان ثابت نیز وجود دارد.

Hash Table



در جدول Hash از جفت‌های کلید-مقدار برای ذخیره‌سازی داده‌ها استفاده می‌شود. این جداول به دلیل سرعت و کارایی‌شان تقریباً در همه زبان‌های برنامه‌نویسی به نوعی پیاده‌سازی شده‌اند. جداول هَش در جاوا اسکریپت به نام شیء (Object)خوانده می‌شوند و در پایتون دیکشنری نام دارند و در جاوا، اسکالا و Go نیز به صورت Map نامگذاری شده‌اند.

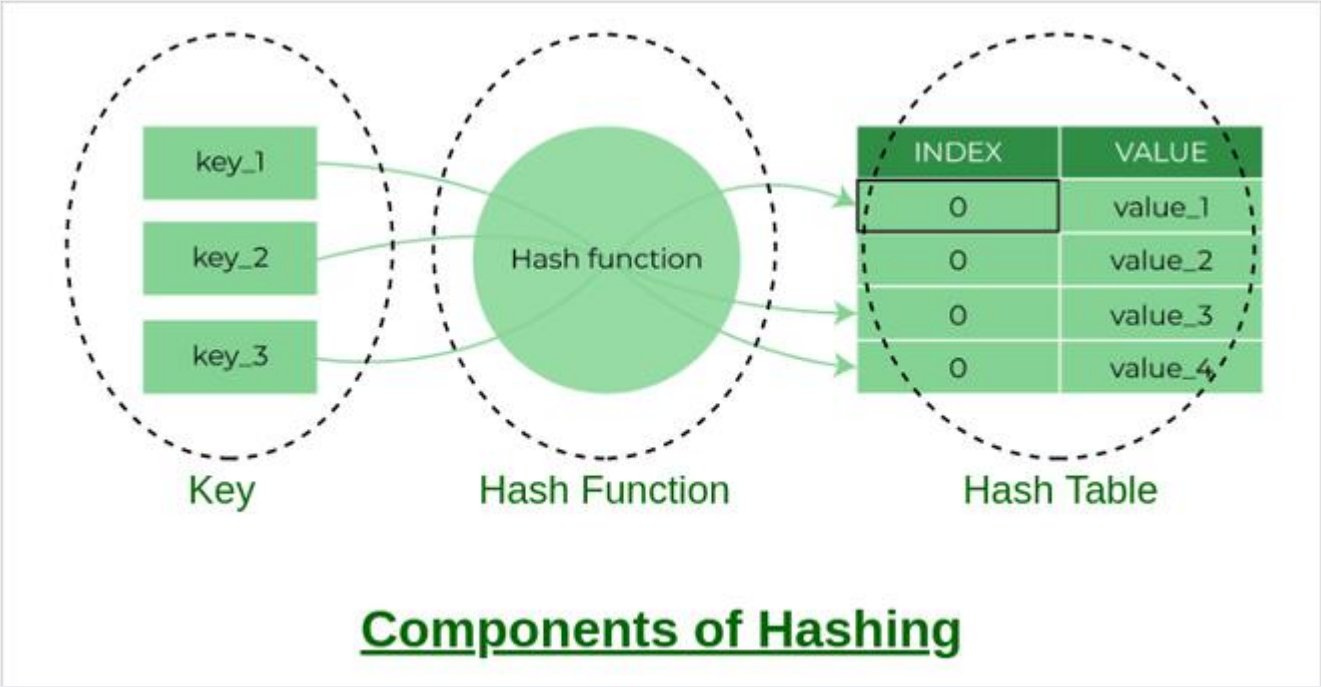
Components of Hashing

کلید: یک کلید می تواند هر رشته یا عدد صحیحی باشد که به عنوان ورودی در تابع درهم سازی استفاده می شود.

تابع هش: توابع هش الگوریتم‌های ریاضی هستند که داده‌ها را با اندازه و طول متغیر دریافت می‌کنند و به خروجی یکسان و قطعی تبدیل‌شان می‌کنند. به عبارتی دیگر هش کلید ورودی را دریافت می کند و اندیس یک عنصر را در آرایه ای به نام جدول هش (hash Table) برمی گرداند. توابع هش رکن اصلی فرایند هشینگ هستند (در فرایند هشینگ از طریق یک فرمول ریاضی ورودی‌ها با مقادیر مختلف تبدیل به یک خروجی با مقدار ثابت می‌شوند).

جدول هش (Hash Table) : جدول هش (Hash Table) یک ساختار داده ای است که با استفاده از تابع خاصی به نام تابع هش، کلیدهای مقادیر را نگاشت می کند. Hashداده ها را به صورت انجمنی در آرایه ای ذخیره می کند که هر مقدار داده، اندیس منحصر به فرد خود را دارد.

Hash Table


key_1
key_2
key_3
Hash function
INDEX
VALUE
0
value_1
0
value_2
0
value_3
0
value_4
Key
Hash Function
Hash Table
Components of Hashing

How does Hashing work?

فرض کنید مجموعه ای از رشته های {“ab”, “cd”, “egg”} داریم و می خواهیم آن را در یک جدول ذخیره کنیم.

هدف اصلی ما در اینجا جستجو یا به روز رسانی سریع مقادیر ذخیره شده در جدول در زمان O(1) است و ما نگران ترتیب رشته ها در جدول نیستیم. پس مجموعه داده شده از رشته ها می تواند به عنوان یک کلید عمل کند و خود رشته به عنوان مقدار رشته عمل خواهد کرد اما چگونه مقدار مربوط به کلید را ذخیره کنیم؟

eskazemi

Hash Table

مرحله 1: می دانیم که توابع هش (که برخی فرمول های ریاضی هستند)برای محاسبه مقدار هش استفاده می شوند که به عنوان شاخصی (index) از ساختار داده عمل می کند که در آن مقدار ذخیره خواهد شد.
مرحله 2: بنابراین، بیایید تعیین کنیم:

- “a” = 1,
- “b”=2, .. etc, to all alphabetical characters.

مرحله 3: بنابراین مقدار عددی با جمع همه کاراکتر های رشته

- “ab” = 1 + 2 = 3,
- “cd” = 3 + 4 = 7 ,
- “efg” = 5 + 6 + 7 = 18

مرحله 4 : حال فرض کنید که جدولی به اندازه ۷ برای ذخیره این رشته ها داریم. تابع درهم سازی که در اینجا استفاده می شود، باقی مانده مجموع کاراکترها برتعداد کلید هیا جدول هش است. ما می‌توانیم محل رشته را در آرایه با گرفتن sum(string) mod 7محاسبه کنیم

eskazemi

Hash Table

مرحله 5 : بنابراین مقادیر را به تابع هش داده برای بدست آوردن موقعیتی که باید ذخیره شوند در جدول هش

- “ab” in 3 mod 7 = 3,
- “cd” in 7 mod 7 = 0, and
- “efg” in 18 mod 7 = 4.

eskazemi

0	1	2	3	4	5	6
cd			ab	efg		

تکنیک بالا ما را قادر می سازد تا موقعیت یک رشته داده شده را با استفاده از یک تابع هش ساده محاسبه کنیم و به سرعت مقدار ذخیره شده در آن مکان را پیدا کنیم. بنابراین ایده درهم سازی(hashing) راهی عالی برای ذخیره جفت داده ها (کلید، مقدار)در یک جدول به نظر می رسد.

Hash Table

مثال پایین رو نگاه کنید در جاوا اسکریپت :

```
var myObject = {};
myObject['my_key'] = 10000;
console.log(myObject['my_key']);
```

اول یک آبجکت تعریف کردم. توی خط بعدی یک کلید به اسم my_key و مقدار 10000 رو بهش assignکردم.
اتفاقی که در backgroundرخ میده اینه که جاواسکریپت my_key رو با Hash functionخودش، Hashمیکنه و بعدش با توجه به اون Hash، یک آدرس توی بازه ای که برای اون متغییر در نظر گرفته شده تولید میشه (که در واقع آدرسی از shelvesهای Ramما هستش(مثلا توی مثال بالا شده 711) و بعدش مقدار اون که ۱۰۰۰ هستش رو ذخیره میکنه در اون shelve.
توی خط بعد اون رو logکردم. درواقع برای دسترسی به کلید my_key، همین فرایند بالا اتفاق می افته.

Hash Table

وقتی من یک متغییر رو تعریف میکنم، سیستم یک مقدار مشخصی از Ram رو به اون متغییر اختصاص میده. منطقیه دیگه نیست؟ نمیاد که کل فضای Ramرو اختصاص بده به اون متغییر. البته توی بعضی از زبان ها شما خودتون باید سایز آرایه و یا Hash-table رو مشخص کنید و توی بعضی از زبان های دیگه نیازی به تعریف سایز ندارید که هر دو نوع مزایا معایب خودشون رو دارند. بهشون میگن Static & Dynamic Array

در واقع وقتی یک آرایه Staticتعریف میکنید، سایز اون آرایه رو مشخص میکنید و اون زبان موقع تعریف اون آرایه به ازای سایزی که موقع تعریفش مشخص کردید، یک قسمت از Ramرو بهش اختصاص میده ولی توی حالت Dynamicسایز اون متغییر بسته به تعداد آیتم هایی که درون اون هستند تغییر میکنه.

فرض کنید Ramما ۱۰۰۰ تا بلاک داره. وقتی یک متغییر رو تعریف میکنیم، مثلا از بلاک ۲ تا بلاک ۱۰، به اون متغیر اختصاص داده میشه.

حالا اون Hash functionباید بر اساس ورودی که بهش داده میشه یک آدرس توی اون بازه ای که برای اون متغییر به اختصای داده شده تولید کنه. یعنی مثلا توی این مثال هر کلیدی رو به عنوان ورودی به این Hash functionبدیم باید یک آدرس بین بلاک ۲ تا ۱۰ رو تولید کنه.

به همین دلیل ممکنه بر اساس اون کلیدی که به Hash functionداده میشه آدرس تکراری تولید کنه و در واقع یک Collisionاتفاق بیافته.

Hash Function

توابع هش (درهم ساز) باید ویژگی های زیر را داشته باشند:

Efficiently computable
باید به طور یکنواخت کلیدها را توزیع کند (موقعیت هر جدول به طور مساوی برای هر کدام محتمل است.)
باید تصادم(collisions) را به حداقل برسانیم.
باید ضریب بار پایینی داشته باشد (تعداد آیتم های جدول تقسیم بر اندازه جدول).

پیچیدگی محاسبه مقدار هش با استفاده از تابع هش:
پیچیدگی زمانی: O (n)
پیچیدگی فضایی: O(1)

eskazemi

Hash Function Features

1. طول خروجی تابع هش یا مقدار هش، همیشه ثابت است.
2. تا زمانی که ورودی تغییر نکند، مقداری خروجی تابع هش قطعی و ثابت است.
3. مقدار هش معمولاً بسیار کوچک‌تر از ورودی است. به‌همین خاطر به تابع هش، فشرده‌ساز نیز می‌گویند.
4. عملکرد تابع هش با رمزنگاری متفاوت است.
5. توابع هش به‌صورت یک‌طرفه طراحی شده‌اند. به‌عبارتی دیگر در هشینگ به‌دست آوردن خروجی از ورودی امکان‌پذیر است و عملیات عکس آن یعنی به‌دست آوردن ورودی از خروجی تقریباً غیرممکن است.

Hash Function

There are many hash functions that use numeric or alphanumeric keys.

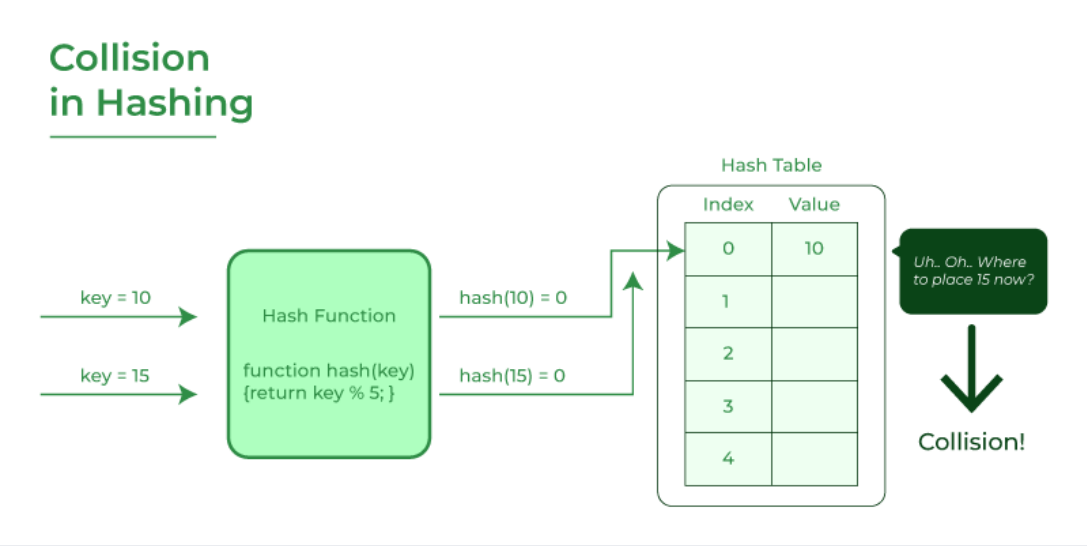
1- Division Method
2- Mid Square Method
3- Folding Method
4- Multiplication Method

collision

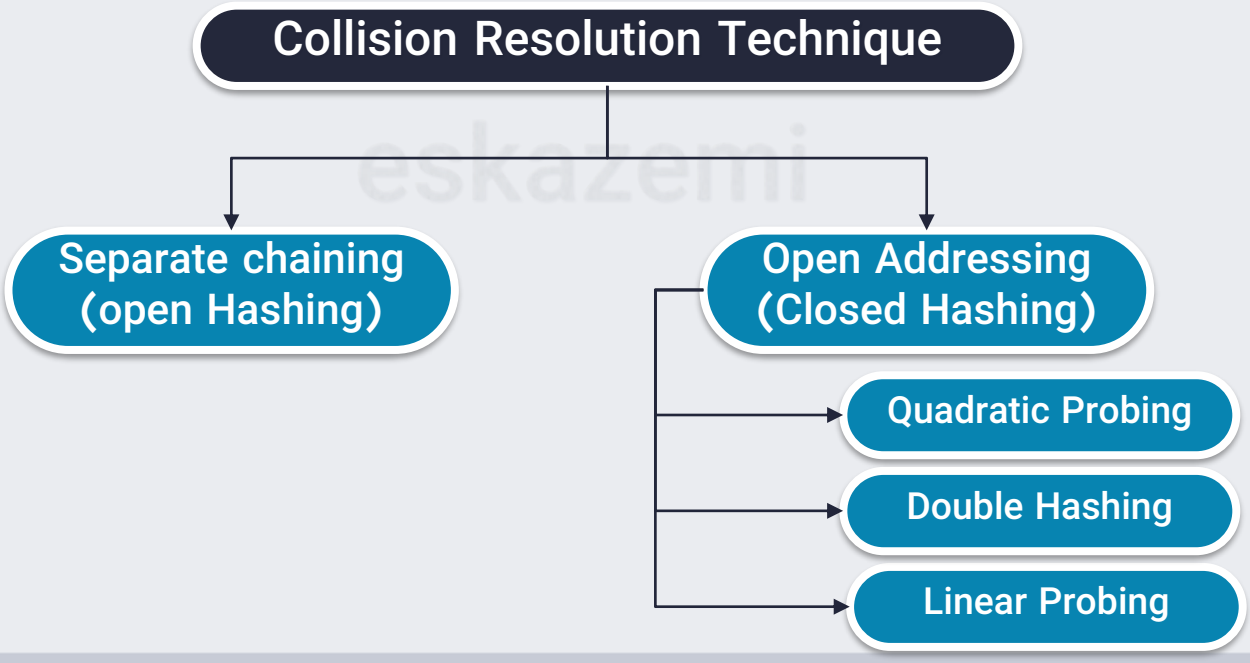
فرآیند درهم سازی یک عدد کوچک برای یک کلید بزرگ تولید می کند، بنابراین این احتمال وجود دارد که دو کلید بتوانند مقدار یکسانی تولید کنندوضعیتی که در آن کلید جدید درج شده به یک مورد قبلاً اشغال شده ارجاع می‌دهد و باید با استفاده از برخی فن‌آوری‌های مدیریت تصادم (collision) با آن برخورد کرد. ولی نکته ای که مهم هستش اینه که وقتی Collisionاتفاق میافته خواندن و نوشتن توی اون بلاک از RAM، کند تر میشه. درواقع توی این حالت، Big O عملیات خواندن و نوشتن (O(Nمیشه.

اگر مثال بالا را در نظر بگیریم، تابع درهم سازی که استفاده کردیم، مجموع حروف است، اما اگر تابع درهم سازی را از نزدیک بررسی کنیم، به راحتی می توان تصور کرد که برای رشته های مختلف، مقدار درهم سازی یکسانی توسط تابع درهم سازی تولید می شود.
به عنوان مثال {"ab"، "Ba"}هر دو مقدار هش یکسانی دارند و رشته {"cd"، "be"}نیز مقدار هش یکسانی تولید می کند و غیره. این به عنوان تصادم (collision) شناخته می شود و در جستجو، درج، حذف و به روزرسانی ارزش مشکل ایجاد می کند.

Hash Table

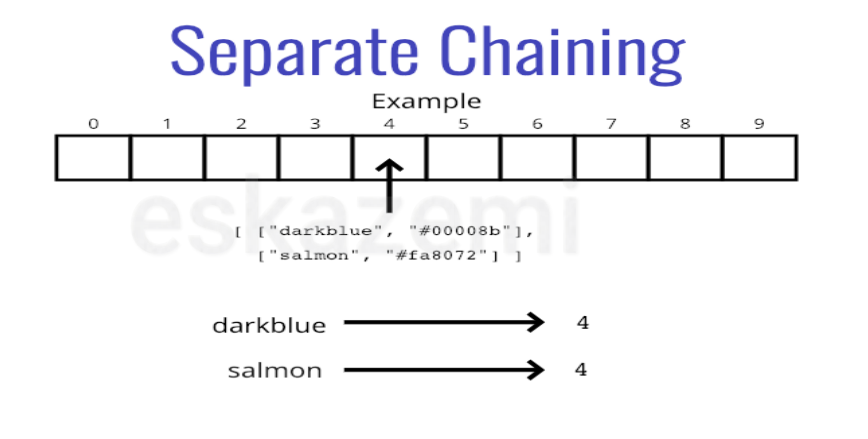

Collision
in Hashing
Hash Table
Index
Value
0
10
1
2
3
4
key = 10
key = 15
Hash Function
function hash(key)
{return key % 5; }
hash(10) = 0
hash(15) = 0
Uh.. Oh.. Where
to place 15 now?
Collision!

How to handle Collisions?



eskazemi

Separate Chaining

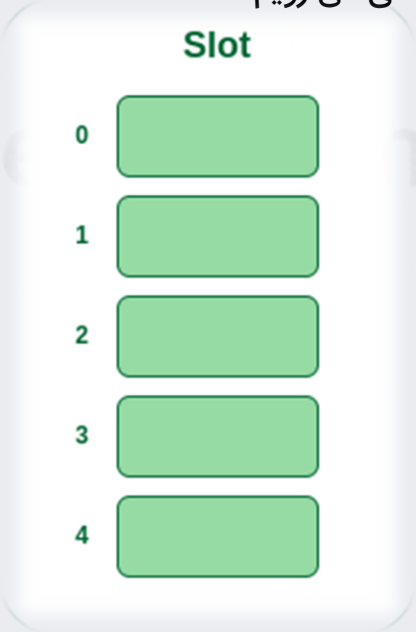


در راهبرد زنجیره‌سازی مجزا، از یک آرایه یا لیست پیوندی دیگر برای ذخیره‌سازی چند مقدار در یک اندیس واحد استفاده می‌شود. به تصویر زیر دقت کنید. هر دو کلید باعث می‌شود اندیس یکسانی ایجاد شود. ما به جای بازنویسی مقدار موجود در اندیس 4، هر دو مقدار را در آرایه نگهداری می‌کنیم.

Linear Probing

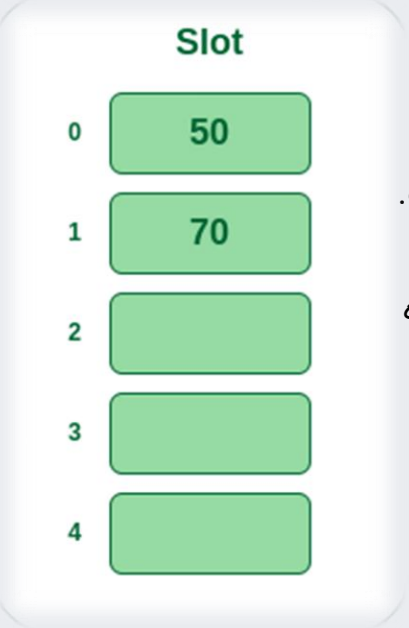
در راهبرد پراب کردن خطی جدول هش به صورت متوالی جستجو می شود که از محل اصلی هش شروع می شود. اگر مکانی که دریافت می کنیم قبلا آشغال شده باشد، به دنبال مکان بعدی می رویم.

Example: Let us consider a simple hash function as “key mod 5” and a sequence of keys that are to be inserted are 50, 70, 76, 85, 93.

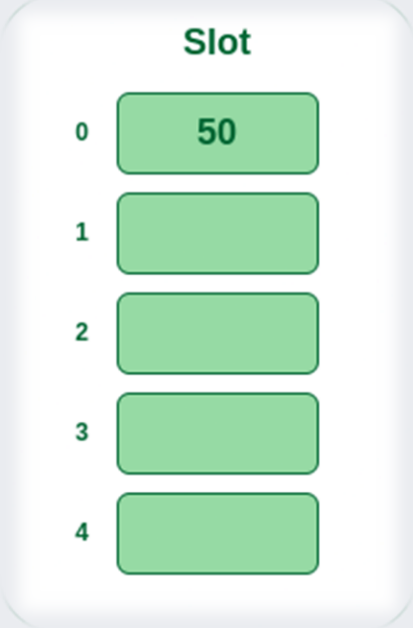


مرحله ۱: ابتدا جدول هش خالی را رسم کنید که با توجه به تابع هش ارائه شده، مقدار هش ممکن از ۰ تا ۴ خواهد داشت.

Linear Probing



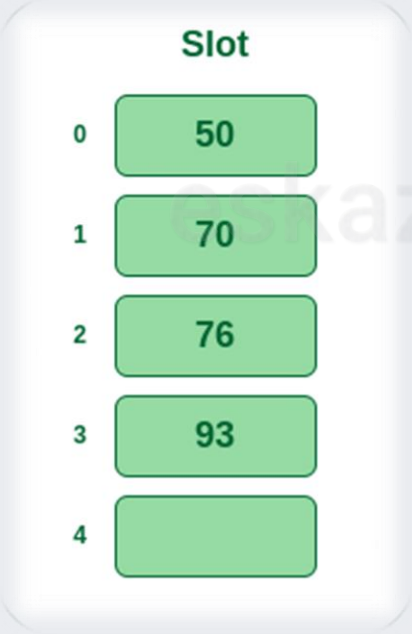
مرحله ۳: کلید بعدی ۷۰ است. برای اسلات شماره ۰ نگاشت می شود زیرا 70%5=0 اما ۵۰ در حال حاضر در اسلات شماره ۰ است بنابراین، اسلات خالی بعدی را جستجو کرده و آن را درج می کند.

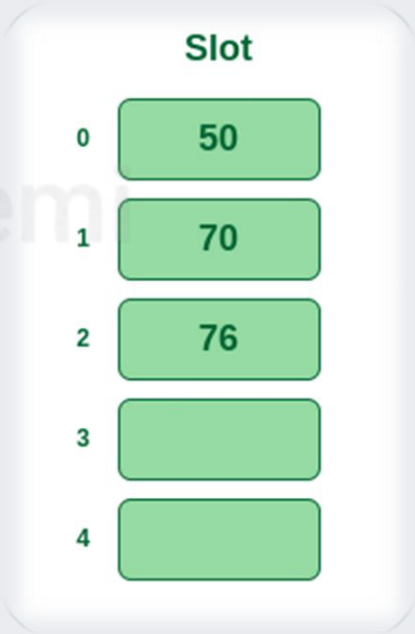


مرحله ۲: حالا تمام کلیدها را در جدول هش یک به یک وارد کنید. کلید اول ۵۰ است. به دلیل اینکه 50%5=0 پس آن را در اسلات شماره ۰ وارد کنید.

Linear Probing

مرحله ۵: کلید بعدی ۹۳ است که به دلیل 93%5=3 به اسلات شماره ۳ منتقل می شود، پس آن را در اسلات شماره ۳ وارد کنید.





کلید بعدی ۷۶ است. برای اسلات شماره ۱ نگاشت می شود زیرا 76%5=1 اما ۷۰ در حال حاضر در اسلات شماره ۱ قرار دارد بنابراین، اسلات خالی بعدی را جستجو کرده و آن را درج می کند.

eskazemi

Thanks!

Any questions?

You can find me at:

- @eskazemi
- m.esmaeilkazemi@gmail.com