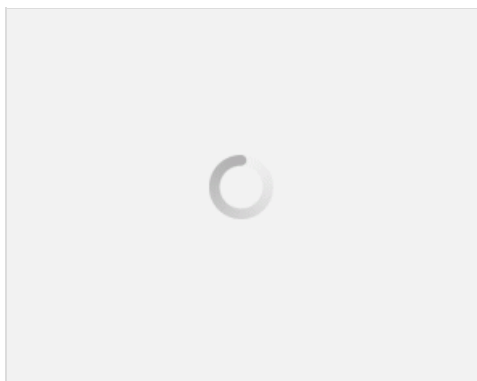


Hashing چیست؟ چرا از هشینگ استفاده می کنیم؟ قسمت ۱ (نسخه PDF)

Hashing چیست؟ چرا از هشینگ استفاده می کنیم؟ قسمت ۱ (نسخه چاپی)

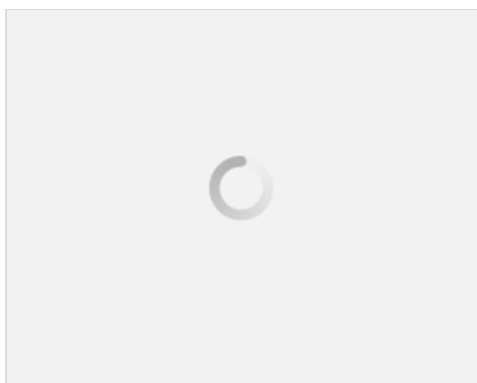
Hashing چیست؟ بی شک واژه هش و هشینگ را ممکن است روزانه بارها و بارها بشنوید و یا در صحبت های خود بکار برید. درباره ان نظر میدهید و نظر میشنوید اما خیلی کمتر از آن رغبت کرده اید خود را در دریای مفهوم آن غرق کنید. در اینجا میخواهیم به عنوان توفیقی اجباری شما را با مفهوم هش و هشینگ بطور ساده آشنا کنیم. برای این موضوع فقط کافیه کمی و فقط کمی از اطلاعات ریاضی خود را به ذهنتان فراخوانی کنید. با ما همراه باشید:



چرا هشینگ؟

اینترنت تشکیل شده از میلیون ها انسان که روزانه چندین ترابایت محتوا تولید میکنند. بر طبق سرویس های جستجوی داده های اینترنت، مقدار محتوای تولید شده در اینترنت در هر شش ماه، دو برابر میشوند. با وجود این رشد سریع، جستجوی هرچیز در اینترنت غیر ممکن به نظر میرسد، مگر آن که ساختار و الگوریتم های جدیدی از داده را بر طبق مدل موجود، توسعه دهیم تا بتوانیم داده ها را در اینترنت ذخیره سازی و به آن ها دسترسی پیدا کنیم. اما چرا ساختارهای سنتی و مرسوم ذخیره سازی داده مانند استفاده از آرایه ها و لیست های در هم لینک شده نمیتوانند جوابگوی نیاز امروزه باشند؟

فرض کنید داده زیادی را در اختیار داریم که آن ها را بر روی یک آرایه ذخیره کردیم. زمان تخمینی برای آن که پارامتری را در آرایه جستجو و پیدا کنیم میتواند $O(\log)$ و یا $O(n)$ باشد. این زمان بستگی به این دارد که آرایه ما مرتب شده است یا خیر. اگر آرایه مرتب شده باشد، استفاده از روش هایی چون جستجوی باینری میتواند در جستجوی آرایه مورد استفاده قرار گیرد. در غیر این صورت آرایه باید بصورت خطی مورد جستجو قرار گیرد. تازه این در حالی است که ممکن است نتیجه حاصله از جستجوی حجم زیادی از داده مورد نظر ما نباشد.



بنابراین با توجه به موارد بالا، میخواهیم درباره تکنیک جدیدی به اسم هشینگ (Hashing) صحبت کنیم که این امکان را بما میدهد تا هر داده ورودی را زمان ثابت $O(1)$ بروزرسانی و بازیابی کنیم. زمان ثابت یا $O(1)$ به این مفهوم است که مقدار زمان لازم برای بازیابی داده بستگی به سایز داده (n) ندارد.

سختی ساختار داده

در دنیای ریاضیات، یک نگاشت به ارتباط بین دو مجموعه اطلاق میشود. ما میتوانیم نگاشت M را به عنوان مجموعه ای از یک زوج در نظر بگیریم، جایی که هر جفت بشکلی از (Key, Value) است، در اینصورت برای کلید داده شده، میتوانیم مقداری را با استفاده از نوعی عملکرد که کلیدها را به مقادیر متناظر میکند، پیدا کنیم. این کلید میتواند با استفاده از تابعی به نام تابع هش محاسبه شود. در ساده ترین حالت، میتوانیم تصور کنیم که آرایه ای بشکل نگاشت خواهد بود اگر کلید بصورت یک شاخص قرار گیرد و مقدار بصورت ارزشی برای آن شاخص. برای مثال آرایه A با داده شده است.

اگر i کلید آن باشد، در اینصورت میتوانیم مقدار را با استفاده از جستجوی $A[i]$ ، پیدا کنیم. ایده استفاده از جدول هش میتواند تعمیم یافته موارد بالا باشد که در ادامه آن را توضیح خواهیم داد. مفهوم جدول هش (Hash Table) ایده ای کلی برای یک آرایه است، زمانی که کلید الزاماً یک مقدار صحیح نیست. ما میتوانیم بجای کلید، یک نام و یا هر عنوانی در رابطه با شیء مورد نظر بگذاریم. هدف پیدا کردن تابع هش برای محاسبه یک شاخص است؛ در این صورت یک شیء میتواند در جای خاصی از جدول ذخیره شود مثلاً جایی که بتواند راحتی پیدا شود.

- مثال : فرض کنید مجموعه از رشته های {"abc", "def", "ghi"} را در اختیار داریم که میخواهیم آن ها را در جدول ذخیره کنیم. هدف ما در اینجا پیدا کردن و یا برورسانی این رشته ها از جدول و با استفاده از $O(1)$ است. نگرانی از بابت ترتیب آن ها یا هرگونه وجود نظم در آن ها به هیچ وجه وجود ندارد. فرض میکنیم $a=1$ ، $b=2$ و همین ترتیب را برای تمام حروف الفبا در نظر میگیریم. سپس میتوانیم براحتی با این مکانیزم برای هرکدام از رشته ها بوسیله جمع مقدار عددی کاراکترها، یک عدد را محاسبه کنیم.

```
"abc"= 1+2+3=6,"def"=4+5+6=15,"ghi"=7+8+9=24
```

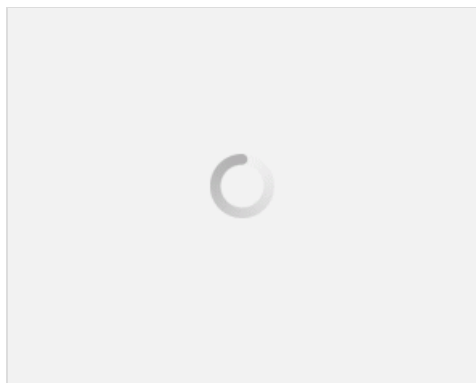
اگر ما فرض کنیم که جدولی با سایز ۵ برای ذخیره این رشته ها داریم، میتوانیم مکان رشته را با استفاده از فرمول "sum mod ۵" محاسبه کنیم. بنابراین میتوانیم

• "abc" را در $6 \bmod 5 = 1$

• "def" را در $15 \bmod 5 = 0$

• و "ghi" را در $24 \bmod 5 = 4$

یعنی مکان های ۱، ۰ و ۴ ذخیره کنیم.



در نهایت میخواهیم اگر به ما رشته ای داده شد، بتوانیم بسرعت مکان آن را با استفاده از یک تابع هش ساده که باقی مانده تقسیم مجموع کاراکترها بر سایز جدول است را محاسبه کنیم. با استفاده از این مقدار هش میتوانیم رشته را جستجو کنیم. به نظر میرسد این روش راه خوبی برای ذخیره یک دیکشنری باشد. همچنین راه خوبی برای ذخیره زوج های (key,value) در جدول نیز هست. در قسمت بعد قصد داریم تا درباره مسائل و مشکلات موجود پیرامون هشینگ، پیدا کردن یک تابع هش خوب و اجرای ساده ای از یک جدول هش صحبت کنیم. در قسمت بعد با ما همراه باشید. سربلند و ITPro ای باشید.

پایان قسمت اول

نویسنده: احسان امجدی

هرگونه نشر و کپی برداری بدون ذکر منبع و نام نویسنده دارای اشکال اخلاقی می باشد.

مطلب اصلی