جزوه جلسه پانزدهم داده ساختارها و الگوریتم

۲۳ آبان ۱۴۰۰

فهرست مطالب

۱ درهم سازی با فرض درهم سازی یکنواخت ساده

در این حالت فرض میکنیم تابع درهم سازی، هر کلید را با احتمال مساوی و مستقل از بقیه کلیدها به یکی از خانه های جدول درهم سازی می نگارد (map میکند). احتمال ذکر شده در این حالت برابر $1/\mathrm{m}$ میباشد که m اندازه جدول میباشد.
تابع درهم سازی که طراحی میکنیم، هر کلید را به صورت مجزا به یک خانه مپ میکند و عملیات هر کلید، تاثیری رو عملیات متناظر با کلید های دیگر ندارد. به همین دلیل احتمال نگاشت، مستقل است.
همچنین چنین فرضی در عمل به صورت صد در صدی عملی نمیشود و با ارائه روش هایی میتوان به فرض ارائه شده بسیار نزدیک شد. (مانند درهم سازی سراسری)

۱.۱ ضریب بارگذاری یا α

برای سادگی محاسبات، ضریب بارگذاری را تعریف میکنیم:

$\alpha = \frac{n}{m}$

در این رابطه، n تعداد اشیا و m اندازه جدول میباشد.
با درنظر گرفتن درستی فرض ما، میانگین اندازه زنجیر در هر خانه به صورت میانگین برابر α میشود.
حال میتوان مرتبه زمانی عملیات مربوط به جدول درهم سازی را بررسی کرد

۱.۱.۱ search(key)

میانگین زمان جست و جو در این حالت برابر $O(1+\alpha)$ میباشد. همچنین اگر $m = \omega(n)$ باشد، آنگاه زمان جست و جو به $O(1)$ میرسد.

۱.۱.۲ insert(key, value), delete(key)

مطلوب است که زمان این عملیات نیز برابر $O(1)$ باشد. در ابتدا، درکی از میزان بزرگی m نداریم؛ پس میتوانم:
۱. یک جدول بسیار بزرگ در نظر بگیریم که در این صورت میتواند بخش بزرگی از آن استفاده نشده باقی بماند و حافظه اضافی زیاده توسط الگوریتم اشغال شود.
۲. برخلاف حالت اول، m را کوچک در نظر گرفت که در این صورت α بزرگ میشود و مرتبه زمانی جست و جو افزایش می یابد.
بهترین رویکرد، استفاده از آرایه پویا است؛ به این صورت که هنگام پر شدن جدول، جدولی جدید به اندازه ۲ برابر قبلی و هنگام خالش شدن $3/4$ جدول، جدولی جدید با اندازه نصف جدول قبلی میگیریم و تابع درهم سازی جدیدی در نظر گرفته و عناصر را

تحت آن تابع، به خانه های جدول جدید نگاشت میکنیم.
در نتیجه، هزینه زمانی درج و حذف نیز به صورت سرشکن برابر $O(1)$ میشود.

۲ توابع درهم سازی متداول

۱.۲ روش تقسیم

$h(k) = k \, mod \, m$
این روش در عمل وقتی مفید است که m عددی اول باشد و به توان های اعداد کوچک مخصوصا ۲ و ۱۰ نزدیک نباشد.
برای مثال اگر $m = 10$ باشد، آنگاه k های زوج در خانه های زوج جدول و k های فرد در خانه های فرد جدول قرار میگیرند که به دلیل قابل پیشبینی بودن تابع Hash نامطلوب است.
همچنین از اعداد اول بزرگ به فرم $2^m - 1$ (مانند $2^{31} - 1$) نیز استفاده نمیکنیم؛ زیرا کلید هایی با مقادیر کم در تعیین $h(k)$ تاثیر بیشتری میگذارند.
از معایب این روش میتوان به زمان بر بودن پیدا کردن یک عدد اول در هربار تغییر اندازه جدول اشاره کرد. همچنین عملگر باقی مانده (mod) زمان بیشتری نسبت به ضرب یا جمع برای انجام میخواهد.

۲.۲ روش ضرب

$h(k) = [(a.k) \, mod \, 2^w] >> (w-r)$
a یک عدد تصادفی در بازه $[2^{w-1}, 2^w]$ و فرد است که ترجیحا به دو سر بازه نزدیک نیست و w برابر تعداد ارقام k و a در مبنای دو میباشد و در نهایت $m = 2^r$
a و k هردو w بیتی هستند؛ پس حاصل ضرب آنها حداکثر 2w بیتی خواهد شد. در عملیات باقی مانده گیری، تنها w بیت کم ارزش حاصل ضرب باقی می ماند و در نهایت با شیفت به راست، w-r بیت کم ارزش دیگر هم از بین رفته و تنها r بیت باقی میماند.
در این روش به دنبال m نمیگردیم و همچنین محاسبه آن نیز راحت است. محاسبه باقی مانده نیز به روش معمول انجام نمیشود و زمان بر نیست (مثلا میتوان حاصل ضرب را با یک رشته بیتی که w بیت با ارزش آن صفر و w بیت کم ارزش آن یک است and منطقی کرد).
در این روش عدد تصادفی a در ابتدا یک بار انتخاب میشود و به هنگام تغییر جدول، تغییر نمیکند. همچنین غیر قابل پیشبینی بودن a از نقاط قوت این روش است که باعث میشود ورودی بد برای تابع تولید نشود.

۳.۲ درهم سازی سراسری یا Universal Hash

$h(k) = [(a.k + b) \, mod \, P] \, mod \, m$
P یک عدد اول بزرگ (بزرگتر از اندازه مجموعه U(مجموعه جهانی کلیدها)) و a و b نیز دو عدد تصادفی صحیح در بازه $[0, P-1]$ میباشند. عدد P نیز به دلیل تعریف آن، یکبار در ابتدا محاسبه میشود و تغییر نمیکند.
به دلیل اینکه این روش به مقادیر تصادفی a و b بستگی دارد، به خانواده از توابع درهم سازی معروف است؛ گویی بسته به مقادیر a و b یک تابع درهم سازی جدید انتخاب میشود.
اثبات میشود برای دو کلید مجزای k_1 و k_2 فرض درهم سازی یکنواخت ساده برقرار است؛ به عبارت دیگر:
$\forall k_1, k_2 \text{ and } k_1 \neq k_2 : P_{a,b}[h(k_1) = h(k_2)] = 1/m$
از انجا که این احتمال برای هر کلید بیان شد، پس احتمال روی تصادفی بودن a و b است.

۳ سرفصل مطالب جلسات آینده

۱.۳ ادامه روش های مقابله با برخورد

۱. استفاده از زنجیر

۲. Open Addressing یا تلاش های متوالی با چند تابع درهم سازی برای $h(k)$ به نحوی که خانه $h(k)$ام جدول خالی باشد.
در این روش به دنبال کاهش تعداد تلاش یا برای پیدا کردن جایگاه خالی هستیم. یکی از اقداماتی که در این راستا انجام میدهیم ۲ برابر کردن اندازه جدول به هنگام پر شدن نصف خانه های آن است.
در عمل، این روش بیشتر از روش استفاده از زنجیر استفاده میشود.

۳. Perfect Hashing

۴. درهم سازی کوکو