جزوه جلسه شانزدهم داده ساختارها و الگوریتم

۲۵ آبان ۱۴۰۰

فهرست مطالب

۱ روش آدرس دهی باز (Open Addressing) برای مقابله با برخورد

ایده حل مشکل این است که به جای استفاده از زنجیر در خانه های جدول، کل اشیا را بدون استفاده از زنجیر در خود جدول ذخیره کرد و در صورت برخورد دو شی، تلاشی مجدد برای درج شی میکنیم. در این شیوه اندازه جدول (m) را همواره طوری نگه میداریم که: $m \geq 2n$

۱.۱ دنباله وارسی

دنباله وارسی شی x (Probing Sequence) به صورت زیر تعریف میشود:
$h(x,0),\ h(x,1),\ h(x,2),\ \ldots\ ,h(x,m-1)$
در این دنباله، $h(x,i)$ نشانگر تلاش $i+1$ام برای درج عنصر x در جدول تحت تابع درهم سازی h میباشد.
در یک دنباله وارسی، مطلوب این است که طول دنباله تا حد امکان کاهش یابد و به مقادیری مانند n یا $m/2$ نرسد.
حال عملیات سه گانه جدول درهم سازی را با این روش بررسی میکنیم:

۱.۱.۱ insert

برای درج کافی است از ابتدای دنباله شروع به بررسی خانه های جدول بکنیم و هرگاه به اولین خانه خالی رسیدیم، عنصر را درج کنیم.

۲.۱.۱ search

برای جست و جو نیز خانه های جدول را به ترتیب دنباله وارسی بررسی میکنیم و این کار را تا رسیدن به شی مدنظر یا یک خانه ادامه میدهیم. این بدین معنی است که هر شی x تا یک مرحله خاص از دنباله جلو رفته است و اگر به یک $h(x,i)$ خالی برسیم یعنی x در جدول درج نشده است.

۳.۱.۱ delete

برای حذف عنصر x نیاز به دقت بیشتری داریم؛ اگر پس از حذف یک کلید خانه مربوط به آنرا خالی کنیم، جست و جو های بعدی ممکن است درست کار نکند. به همین جهت پس از حذف شئ، در خانه مربوط به آن مینویسم "حذف شده". به این تکنیک، پرچم (flag) یا سنگ قبر (tomb stone) میگویند.

هنگام جست و جو، این خانه ها به مانند یک خانه پر در نظر گرفته میشوند و در هنگام درج نیز مشابه خانه های خالی هستند. در هنگام درج نیز برای جلوگیری از مشکلات احتمالی، با جست و جو در جدول، تکراری بودن کلید را بررسی میکنیم؛ اگر کلید تکراری نبود به صورت عادی درج انجام میگیرد و درغیر اینصورت در خانه اول جست و جو شده درج انجام میگیرد.
حذف زیاد باعث افزایش پرچم ها در جدول و افزایش زمان اجرای الگوریتم میشود. برای حل این مشکل، هنگامی که تعداد خانه های با پرچم به ۱۰ درصد کل جدول رسید، جدولی جدید با تایع درهم سازی و اندازه قبلی میسازیم. اثبات میشود که هزینه سرشکن ایجاد جدول جدید برابر $O(1)$ میباشد.

۲.۱ روش های وارسی

۱.۲.۱ وارسی خطی یا **Linear Probing**

$h(x,i) = (h'(x) + i)\ mod\ m$
به مانند پارک کردن ماشین در یک قسمت از پارکینگ عمل میکنیم؛ یعنی از یه جای خالی شروع کرده و به جلو میرویم تا به اولین جای خالی برسیم.
تابع $h'(x)$ نیز یک تابع درهم سازی از توابع معرفی شده در جلسه قبل است.
از معایب این روش میتوان به پر شدن یک قسمت از جدول (اینجاد خوشه) اشاره کرد. هرگاه که $h'(x)$ در یک قسمت پر جدول بیوفتد، زمان درج و جست و جو نیز افزایش می یابد. هرچه خوشه ها بزرگتر باشند، احتمال رشد آنها نیز بیشتر میشود و احتمال وجود خوشه به اندازه $\Theta(logn)$ نیز افزایش می یابد که زمان های درج و جست و جو در این حالت به $O(logn)$ میرسد.

۲.۲.۱ وارسی درجه دو

$h(x,i) = ((h'(x) + ax^2 + bx)\ mod\ m$
تابع $h'(x)$ نیز مانند حالت قبل یه تابع معمول درهم سازی است. همچنین ضرایب a و b باید به خوبی انتخاب شوند و لزومی به انتخاب تصادفی نیست.
در روش قبلی، اگر یک خانه پر بود به سراغ خانه بعدی اش میرفتیم که این کار از معایب وارسی خطی بود؛ اما در این روش خانه بعدی که میخواهیم آنرا بررسی کنیم در یک ناحیه دیگر در جدول قرار دارد و به اصطلاح پخش میشویم و مشکل خوشه تا حدودی حل میشود.
ثابت شده وارسی درجه ۵ (با انتخاب مناسب ضرایب) خوب کار میکند و وارسی های درجه ۴ نیز بر خلاف درجه ۵، عملکرد بدی دارند.

۳.۲.۱ درهم سازی دوگانه

$h(x,i) = (h_1(x) + ih_2(x))\ mod\ m$
توابع $h_1(x)$ و $h_2(x)$ جز توابع معمول درهم سازی هستند که باید $h_2(x)$ و m نسبت به هم اول باشند. بدین منظور:
۱. m اول باشد و $h_2(x)$ بین ۰ تا m-1 باشد.
۲. m توانی از ۲ و $h_2(x)$ فرد باشد.
راه حل دوم برای درهم سازی دوگانه بهتر است.

۳.۱ فرض درهم سازی یکنواخت یا Uniform Hashing Assumption

این فرض بیان میکند که دنباله وارسی هر کلید، یکی از $m!$ جایگشت ممکن را به صورت تصادفی و مستقل تولید میکند.
هیچ کدام از روش های مطرح شده بصورت کامل این فرض را پیاده سازی نمیکنند اما درهم سازی دوگانه و درجه ۵ تا حد خوبی به این فرض نزدیک هستند.
حال با فرض درستی این فرض، زمان اجرای عملیات سه گانه را در روش آدرس دهی باز بررسی میکنیم:

۱.۳.۱ insert

در مرحله اول n خانه پر و $m-n$ خانه خالی هستند. پس احتمال موفقیت برابر $\frac{m-n}{m}$ میباشد.
$P = \frac{m-n}{m}$
در مرحله دوم یک خانه حذف می شود و احتمال به $\frac{m-n}{m-1}$ میرسد.
.
.
.
در مرحله iام احتمال موفقیت برابر با $\frac{m-n}{m-(i-1)}$ میباشد.
از مرحله دوم، احتمال موفقیت بزرگتر از P میباشد یا به دیگر بیان، حداقل برابر با P میباشد. پس میانگین تعداد مراحل مورد نیاز برای درج برابر میشود با:
$E[\#trials] \leq \frac{1}{P} = \frac{1}{1-n/m} = \frac{1}{1-\alpha}\ ,\ \alpha = n/m$
در نتیجه، زمان درج برابر با $O(\frac{1}{1-\alpha})$ میباشد.

۲.۳.۱ delete و search

به طریق مشابه و مانند درج، برای حذف و جست و جو نیز ثابت میشود که زمان مورد نیاز برابر $O(\frac{1}{1-\alpha})$ می باشد.

۲ فیلتر بلوم یا Bloom Filter

ابتدا با مفاهیم مربوط آشنا میشویم.
سرور های CDN سرورهایی هستند که برای ذخیره سازی داده های ثابت به کار میروند (مانند تصاویر و متن های ثابت یک سایت یا برنامه). از اولین شرکت هایی که چنین سرور هایی در اختیار شرکت هایی مانند یاهو قرار داد، آکامای نام داشت.
این شرکت دید که یک سوم حجم سرور هایش پر از فایل های one-hit-wonders (فایل هایی که یک بار و توسط یک فرد استفاده شده است) میباشد. راه حل این مشکل فیاتر بلوم بود.
اگر یک فایل توسط حداقل دو نفر استفاده میشد، روی سرور ها باقی میماند و در غیر اینصورت حذف میشد.
فیلتر بلوم یک روش بسیار پرکاربرد برای جست و جو (وجود یا عدم وجود) احتمالی یک عضو در مجموعه میباشد؛ بدین صورت که اگر این الگوریتم پاسخ یک کوئری جست و جو را مثبت بدهد (عنصر وجود دارد)، به احتمال $1-P$ این عنصر وجود دارد و اگر پاسخ منفی باشد، قطعا عنصر مدنظر وجود ندارد.
در فیلتر بلوم عمل درج داریم ولی حذف نداریم!
فیلتر بلوم در مرورگر کروم برای بررسی مخرب بودن یا نبودن یک سایت، در بیت کوین برای همگام سازی کیف پول، در اتریوم و در پایگاه های داده نیز استفاده میشود.

۱.۲ پیاده سازی فیلتر بلوم

در این فیلتر، یک جدول صفر و یک و به تعداد k تابع درهم سازی داریم. خروجی توابع درهم سازی، مکان یک ها را در جدول نشان میدهد و با یک کردن آن خانه ها درج صورت میگیرد.
برای جست و جو نیز k خانه که خروجی توابع درهم سازی هستند را بررسی میکنیم؛ اگر حداقل یکی صفر بود با قطعیت میگوییم عنصر در جدول وجود ندارد و در غیر اینصورت با یک احتمال خطا میگوییم عنصر در جدول وجود دارد. احتمال خطا نیز به صورت زیر تعریف میشود:

$$P = (1 - e^{-kn/m})^k$$

که با فیکس کردن احتمال خطا روی یک مقدار خاص، سایز جدول به دست میاید:
$m = \frac{-nlog_e p}{(ln2)^2}$, $k = (\frac{m}{n})ln2$