الگوریتم
عضویت

محسن هوشمند
دانشکده علم رایانه و تکنولوژی اطلاعات
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

عضویت

مسئله عضویت برای مجموعه داده،

- تصمیم‌گیری در مورد تعلق یا عدم تعلق مقداری به آن مجموعه

برای مجموعه‌های کوچک،

- حل مسئله با جستجوی مستقیم و مقایسه مقدار داده شده با هر عضو
- وابستگی چنین رویکرد ساده‌ای به تعداد عناصر مجموعه و به طور متوسط نیاز به $O(log\ n)$ مقایسه (روی داده‌های مرتب

برای مجموعه‌های عظیم از مقادیر

- ناکارآمدی رویکرد مذکور
- نیاز به زمان و حافظه O(n)برای ذخیره عناصر

راه‌حل‌های ممکن

- مانند تقسیم چنین مجموعه‌هایی به قطعات و اجرای مقایسه‌ها به صورت موازی
- کمک کننده در کاهش زمان محاسبه
- عدم قابل اجرا بودن چنین روش‌هائی
  - ناممکنی ذخیره چنین مجموعه‌های عظیمی از عناصر
  - عدم نیاز به یافتن اصل تطابق و کفایت صرف وجود تطابق ⟸امکان ذخیرهٔ امضای عناصر به جای مقدار کامل آنها

عضویت- مثال-مسئله مرور ایمن

ایجاد مرورگر وب
- برخی از URL-ها حاوی بدافزار هستند

در پی هشدار به کاربر در صورت درخواست وی برای بازدید از آن صفحات
- یا حتی از جلوگیری بازدیدش

راه‌حل فوری با حداقل‌سازی ترافیک شبکه
- ذخیره همه این URL-ها در برنامه و پس از وارد کردن URLتوسط کاربر، صرفا بررسی بدافزاری آن

هدایت در صورت عدم بدافزاری

امکان چنین پیاده‌سازی ساده‌ای تا زمان کم بودن تعداد URL-های مخرب کم
- اما، اینصادق نبودن در برنامه‌های دنیای واقعی

نیاز به ساختار ویژه ذخیرهٔ URL-های مخرب (یا ایده‌آل، فقط برخی اطلاعات درباره آنها) بدون رشد خطی در اندازه

سایر الزامات شامل
- پشتیبانی از بررسی فهرست شدن URL و سرعت تا حد امکان اجتناب از انتشار طولانی‌مدت کاربران مدت طولانی
- کاربردهای مسئله عضویت نقش اساسی در علوم کامپیوتر و شاخه‌های مختلف

عضویت-مثال- توالی‌های دی‌ان‌اِی

طبقه‌بندی توالی‌ها به عنوان «جدید» یا متعلق به ژنوم شناخته شده
- از مسائل مهم در مطالعات متاژنومیک
- فیلتر کردن داده‌هایی دیده شده

مرحله پیش‌پردازش راه‌انداز عضویت
- درصورت اجرای کارآمد منجر به کاهش پیچیدگی داده‌ها قبل از انجام تحلیل دقیق‌تر

حل مسئله جستجوی سریع با درهم‌ساز
- ساده‌ترین راه برای انجام آن

نگاشت هر عضو از مجموعه به جدول درهم با تابع درهم‌ساز
- دارای احتمال کمی از خطا (ناشی از برخوردهای احتمالی درهم‌ساز)
- نیاز به حدود $O(\log n)$ بیت برای هر مقدار درهم‌ساز شده
- در عمل غیرقابل اجرا برای مجموعه‌های داده عظیم

مطالب جاری

معرفی راه‌حل‌های متفاوت از جدول‌های درهم با
- نیاز به فضای کمتر
- جستجوهای سریع‌تر
- احتمالات خطای کمتر

چنین داده‌ساختارهای کارآمد از نظر فضا کمک کنندهٔ به مدیریت حجم

امکان اجرای پرسش‌های عضویت با عملکرد خوب

فیلتر بلوم، بهبودها، و جانشین‌های متاخر

فیلتر بلوم

از سازوکارهای استاندارد مسئلهٔ عضویت در سامانه‌های با مجموعه‌داده‌های بسیار بزرگ

کاربرد گستردهٔ آن‌ها، به‌ویژه در شبکه‌ها و پایگاه داده‌های توزیع‌شده،

کارآمدی قابل‌توجه در شرایطی

- نیاز به نوعی کارکرد مشابه جدول درهم
- اما بدون در اختیار داشتن فضای کافی

بارتون بلوم معرفی آن در سال ۱۳۴۹

- اما مهجوری تا این اواخر
- اشتهار طی دو هه اخیر
- در پی افزایش نیاز به مهار و فشرده‌سازی مجموعه‌داده‌های عظیم
- شکوفائی! ساختار

جلب توجه جامعهٔ پژوهشی علم رایانه

- طراحی گونه‌های متعدد از آن با هدف برطرف‌کردن برخی محدودیت‌های این ساختار و سازگارکردن آن با نیازمندی‌های متفاوت

فیلتر بلوم

داده‌ساختار احتمالاتی فضا-کارآمد برای نمایش مجموعه داده $D = \{x_1, x_2, \dots, x_n\}$ از n عضو

پشتیبانی از دو عملیات افزودن مقدار به مجموعه، و آزمون عضویت در مجموعه

امکان ذخیرهٔ کارآمد مجموعه بزرگ با کنار گذاشتن ذخیره دقیق خود مقادیر

ذخیرهٔ مجموعه‌ای (تقریباً) منحصربه‌فرد از بیت‌ها

- هر بیت حاصل تعدادی تابع درهم‌ساز اعمالیِ الگوریتم روی مقدار

نمایش با آرایه‌ای بیتی با طول m و تعداد توابع درهم‌ساز مختلف $\{h_i\}_{i=1}^{k}$

طراحی و انتخاب m متناسب با تعداد عناصر مورد انتظار n و $k \ll m$

فیلتر بلوم

توابع درهم‌ساز h_i مستقل از هم و به طور یکنواخت توزیع شده

- حاصل توزیعی یکنواخت و تصادفی
- توابع درهم‌ساز به مثابهٔ نوعی مولد اعداد تصادفی
- کاهش احتمال تصادم در فیلتر

کاهش شدید فضای ذخیره‌سازی

بدون توجه به تعداد عناصر در داده‌ساختار و اندازه آنها

- با رزرو چند بیت برای هر عنصر،
- نیاز به تعداد ثابتی بیت

فیلتر بلوم

داده‌ساختار آرایه‌ای بیتی به طول m

- ست که در ابتدا همه بیت‌ها برابر با صفر
- بمعنای خالی بودن فیلتر

درج عنصر x در فیلتر،

- محاسبه مقدار درهم متناظر با هر تابع درهم‌ساز h_i یا $j = h_i(x)$
- یک شدن بیت متناظر j در فیلتر

امکان تنظیم چند بارهٔ بیتی به دلیل تصادم‌های درهم‌ساز

فیلتر بلوم

الگوریتم- افزودن عنصر به فیلتر بلوم

ورودی: عنصر $x \in D$

ورودی: فیلتر بلوم با k تابع درهم‌ساز $\{h_i\}_{\{i=۱\}}^{k}$

برای i از ۱ تا k انجام بده

$j \leftarrow h_i(x)$

$BloomFilter[j] \leftarrow ۱$

فیلتر بلوم

مثال- افزودن عناصر به فیلتر: فیلتر بلوم با طول $m = 10$ و دو تابع درهم‌ساز ۳۲ بیتی مورمور۳ و ف‌ن‌و-۱الف

- تولید مقادیر در محدوده {۰,۱,..., m-1}

$$h_1(x) = MurmurHash3(x)\%m$$
$$h_2(x) = FNV1a(x)\%m$$

فیلتر خالی

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
0	0	0	0	0	0	0	0	0	0

فیلتر بلوم

درج نام پایتخت‌ها در فیلتر

کپنهاگ

محاسبه مقادیر درهم برای یافتن بیت‌های متناظر در فیلتر

$$h_1(\text{Copenhagen}) = MurmurHash3(\text{Copenhagen})\%۱۰ = ۷$$
$$h_2(\text{Copenhagen}) = FNV1a(\text{Copenhagen})\%۱۰ = ۳$$

تنظیم بیت‌های ۳ و ۷ در فیلتر

0	۱	۲	۳	۴	۵	۶	۷	۸	۹
0	0	0	۱	0	0	0	۱	0	0

فیلتر بلوم

امکان بیت‌های متناظر مشترک برای ورودی‌های مختلف

مثال: دوبلین

$$h_1(\text{Dublin}) = MurmurHash3(\text{Dublin})\%۱۰ = ۱$$
$$h_2(\text{Dublin}) = FNV1a(\text{Dublin})\%۱۰ = ۳$$

موقعیت‌های بیت متناظر آن در فیلتر ۱ و ۳

- تنظیم نبودن بیت ۱ از قبل
- به معنای نبودن دوبلین در فیلتر
- مدخل ۳ به عنوان یکی از بیت‌های متناظر از قبل پر

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
0	۱	0	۱	0	0	0	۱	0	0

فیلتر بلوم

آزمون وجود مقدار x در فیلتر،

محاسبهٔ تمامی k تابع درهم‌ساز $h_i = \{h_i(x)\}_{i=1}^{k}$

بررسی بیت‌ها در موقعیت‌های متناظر

اگر همه بیت‌ها روی یک تنظیم شده باشند، امکان عضویت مقدار x در فیلتر

- در غیر این صورت، معدم وجود قطعی مقدار x در فیلتر

عدم قطعیت در مورد وجود عنصر

ناشی از احتمال موقعیت‌هایی از تنظیم برخی از بیت‌های حاصل از مقادیر قبلی اضافه شده

یا به دلیل تصادم‌های شدید، در مواقع تصادم توابع درهم‌ساز با هم

فیلتر بلوم

الگوریتم-۲- آزمون عنصر در فیلتر بلوم

ورودی: عنصر $x \in D$

ورودی: فیلتر بلوم با k تابع درهم $\{h_i(x)\}_{i=۱}^{k}$

خروجی: غلط در صورت نبودن عنصر و درست در صورت احتمال وجود عنصر

برای j از ۱ تا k انجام بده

$j \leftarrow h_i(x)$

اگر $BloomFilter[j] \neq ۱$

برگرداندن غلط

برگرداندن درست

فیلتر بلوم

مثال- آزمون عضویت در فیلتر

فیلتر بلوم با دو مقدار نمایه شده، کپنهاگ و دوبلین

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
0	۱	0	۱	0	0	0	۱	0	0

فیلتر بلوم

آزمون وجود عنصر کپنهاگ در فیلتر

محاسبه مقادیر درهم آن

$h_1(Copenhagen) = 7$
$h_۲(Copenhagen) = ۳$

بررسی بیت‌های متناظر در فیلتر بررسی و برابر یک بودن هر دو

- ادعا بر احتمال وجود کپنهاگ در فیلتر

فیلتر بلوم

رُم

محاسبهٔ مقادیر درهم‌ساز آن جهت یافتن بیت‌های متناظر در فیلتر

$$h_1(\text{Rome}) = MurmurHash3(\text{Rome})\%۱۰ = ۵$$
$$h_2(\text{Rome}) = FNV1a(\text{Rome})\%۱۰ = ۶$$

بررسی بیت‌های ۵ و ۶

- تنظیم نبودن بیت ۵
- عدم عضویت مقدار رم در فیلتر
- عدم نیاز به بررسی بیت ۶

فیلتر بلوم

مقادیر درهم متناظر شهر برلین

$$h_1(\text{Berlin}) = MurmurHash3(\text{Berlin})\%۱۰ = ۱$$
$$h_2(\text{Berlin}) = FNV1a(\text{Berlin})\%۱۰ = ۷$$

تنظیم بیت‌های متناظر ۱ و ۷ هر دو در فیلتر

پیروزی نتیجه تابع آزمون و امکان وجود مقدار در فیلتر

اما طبق مثال اطلاع از عدم وارد کردن چنین مقداری

مثبت کاذب

نمونه‌ای از تصادم درهم‌ساز است.

در این مورد خاص تنظیم بیت ۱ با $h_1(\text{Dublin})$ و بیت ۷ با $h_۲(\text{Copenhagen})$

فیلتر بلوم

در صورت زمان ثبات محاسبهٔ هر تابع درهم‌ساز $\{h_i(x)\}_{i=1}^{k}$
- صادق در توابع درهم‌ساز پرکاربرد

زمان افزودن مقداری جدید یا آزمون مقداری
- ثابت $O(1)$
- مستقل از طول فیلتر m و تعداد مقایدر افزوده شده به فیلتر

وابستگی عملکرد فیلتر بلوم به توابع درهم‌ساز انتخاب شده

کاهش میزان مثبت کاذب مشاهده شده عملی با تابع درهم‌ساز با یکنواختی خوب

سریع‌تر بودن محاسبه هر تابع درهم‌ساز منجر به کاهش زمان کل هر عملیات
- توصیه به اجتناب از توابع درهم‌ساز رمزنگاری

فیلتر بلوم

مثال- جلوگیری از رمزهای عبور به خطر افتاده

صفحه ثبت‌نام سرویس وبی

در پی اجتناب از انتخاب رمزهای عبور ضعیف و به خطر افتادهٔ کاربران

امکان یافتن صدها میلیون رمز عبور هک شده در وب تاریک

امکان استفاده در حمله لغت‌نامه‌ای

- حملهٔ جستجو کامل که با امتحان همه مقادیر از فهرستی از پیش آماده
- تلاش‌های تکراری برای شکست احراز هویت

اطمینان از نبود در فهرست با رمز عبور جدیدی

- عدم امنیت ذخیره رمزهای عبور خام
- عدم ذخیره مجموعه داده عظیم دارای رشد خطی با افزودن هر رمز جدید و کندسازی جستجوها

استفاده از فیلتر بلوم فضا-کارآمد

رویداد مثبت کاذب

- تشخیص اشتباه رمز عبور وارد شده نامناسب
- در چنین موارد نادری، درخواست از کاربر برای رمز عبور دیگری
- معمولاً بدون ضرری

فیلتر بلوم

مثال-کاربرد در اپلیکیشن موبایل بیت‌کوین:

- استفاده از فیلتر بلوم در تبادل داده در شبکه‌های همتابه‌همتا
  - سامانهٔ بیت‌کوین
- از ویژگی‌های مهم بیت‌کوین، تضمین شفافیت بین کاربران
- تحقق از طریق آگاه‌بودن هر گره از تراکنش‌های سایر گره‌ها
- گره‌هایی روی تلفن‌های هوشمند یا دستگاه‌های مشابه با حافظه و پهنای باند محدود اجرا
  - نامناسبی و پرهزینگی نگه‌داری نسخهٔ کامل تمام تراکنش‌ها
- معرفی امکان «تأیید سادهٔ پرداخت» (Simplified Payment Verification – SPV)
- گره: اعلام فهرستی از تراکنش‌های مرتبط با خود،
  - به‌عنوان یک «گره سبک»
- امری که در تقابل با «گره‌های کامل» نگهدارنده کل داده‌های زنجیره

فیلتر بلوم

گره‌های سبک،

- فیلتر بلومی دارای فهرست تراکنش‌های مرتبط با آنها ساخته شده است
- ارسال به گره‌های کامل

گره کامل پیش از ارسال اطلاعات مربوط به یک تراکنش برای گره سبک،

بررسی فیلتر بلوم آن در ابتدا فیلتر بلوم

- کشف ارتباط یا بی‌ارتباطی

اگر «مثبت کاذب» نادیده گرفتن آن در گرهٔ سبک پس از دریافت پیام

فیلتر بلوم-ویژگی‌ها

امکان مثبت کاذب

عدم ذخیرهٔ مقادیر اصلی در فیلتر بلوم

اتکا به درهم‌سازهای محاسبه شده

ذخیره آنها در آرایه‌ای بیتی

با وجود فضا-کارآمدی چنین نمایشی

- منجر به موقعیت‌هایی تایید وجود برخی عناصر که غیر عضو

به دلیل تصادم‌های درهم

عدم وجود دانش قبلی در عملیات آزمون جهت مقایسه مقدار آزمون

فیلتر بلوم-ویژگی‌ها

اصل فیلتر بلوم: هنگام استفاده از فهرست یا مجموعه، و با ارزشی فضا ، در صورت امکان تا زمانی که اثر مثبت کاذب چندان بالا نرفته یا آستانه‌ای را رد نکرده است از فیلتر بلوم بهره گرفته شود.

فیلتر بلوم-ویژگی‌ها

نادر بودن رخداد مثبت کاذب

امکان تخمین احتمال آن

مقدار کران پائین احتمال

$$\mathrm{pr}_{fp} \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$$

فیلتر بلوم-ویژگی‌ها

اگر t کسری از بیت‌های فیلتر بلوم پس از انجام n درج هنوز برابر صفر

k تعداد توابع درهم‌سازی

احتمال رخداد مثبت کاذب (f) برابر با احتمال مشاهدهٔ k بیتِ یک
- به دلیل تایید حضور نَیاز به برقراری k بیت با مقدار ۱

دستیابی به k مقدار ۱ می‌تواند نتیجهٔ جست‌وجوی موفق برای عضو واقعاً درج شده

در صورت انتخاب تصادفی پرس‌وجوها از دامنه‌ای بسیار بزرگ‌تر از مجموعه‌داده
- آنگاه احتمال مثبتِ واقعی، کسری ناچیز از این مقدار
- ناشناختگی مقدار t پیش از آن‌که تمام عملیات درج
- وابستگی به خروجی درهم‌ساز

امکان کار با احتمالِ p
- احتمال صفر بودن یک بیت ثابت پس از انجام n درج
- مناسب جهت تقریب احتمالاتی درصد بیت‌های صفر موجود در فیلتر (t)

$$p = \left(1 - \frac{1}{m}\right)^{nk}$$

فیلتر بلوم-ویژگی‌ها

فهم دلیل:

- آغاز از فیلتر بلوم خالی
- پس از آن‌که نخستین تابع درهم‌سازی $h1$ یکی از بیت‌ها را برابر ۱ قرار می‌دهد،
  - احتمال این‌که یک بیت دلخواه در فیلتر بلوم مقدار ۱ داشته باشد برابر $1/m$ است و احتمال صفر بودن آن برابر $1 - \frac{1}{m}$

پس از آن‌که همهٔ k تابعِ درهم در درج نخست بیت‌های مربوطه را ۱ کردند، احتمال این‌که آن بیتِ مفروض همچنان ۰

$$\left(1 - \frac{1}{m}\right)^k$$

پس از درج کل n مقدار، احتمال صفر بودن بیت مفروض

$$\left(1 - \frac{1}{m}\right)^{nk}$$

- $\left(1 + \frac{1}{x}\right)^x \approx e$

$$p \approx e^{-\frac{nk}{m}}$$

میزان انحراف درصد واقعی صفرها انحراف از این مقدار؟

فیلتر بلوم-ویژگی‌ها

با استفاده از کران‌های چِرنوف

- قضیه‌ای محدودساز احتمال انحراف شدید متغیر تصادفی از میانگینش

امکان نمایش کسر تمرکز مقدار کسر صفرها در فیلتر بلوم حول مقدار میانگین

کران چرنوف برای متغیرهای تصادفی X برابر جمع متغیرهای دنباله‌ایِ مستقلِ از مقادیر دوتائی

$$X = \sum_{i=1}^{n} X_i$$

X تعداد کل بیت‌های صفر موجود در فیلتر بلوم

- اگر $X_i = 0$ بیت i-ام مقدار ۱ و $Xi = 1$ در غیر این‌صورت

فیلتر بلوم-ویژگی‌ها

استفاده از کران چرنوف جهت نمایش عدم انحراف قابل توجه مقدار X از میانگینش

- عدم استقلال متغیرهای Xi در مسئله و وجود اندکی همبستگی منفی
- البته مطلوب‌تر
- در صورت برابر شدن بیتی برابر یک کاهش احتمال ۱ شدن سایر بیت‌ها

بیان کلی کران بالای چرنوف

- با μ به‌عنوان میانگین متغیر تصادفی X

$$Pr[X \geq (1 + \delta)\,\mu] \leq e^{-\mu\frac{\delta^2}{3}}$$

فیلتر بلوم-ویژگی‌ها

اعمال آن در مسئله

$$\mu = E[X] = mp = me^{-\frac{nk}{m}}$$

$$\delta = 1$$

احتمال این‌که X بیش از دو برابر مقدار میانگین خود انحراف پیدا کند

$$Pr[X \geq 2\mu] \leq e^{-\frac{\mu}{3}}$$

ادعا $nk = \theta(m)$

- به معنای مقدار نمایی کوچک بودن احتمال انحراف X از میانگین به‌اندازهٔ بیش از ضریب ۲، $e^{-\theta(m)}$

بنابراین، p تقریبِ بسیار خوبی برای t یا درصد صفرهای فیلتر بلوم

فیلتر بلوم-ویژگی‌ها

با مشاهدهٔ کران، با تعداد مفروض و ثابت عناصر مورد انتظار n،

وابستگی احتمال مثبت کاذب به انتخابk و m

نیاز به بررسی و سبک-سنگینی واضح بین طول فیلتر، تعداد توابع درهم‌ساز، و احتمال چنین رویدادهایی

بدترین حالت هنگام پر بودن فیلتر
- به معنای یک شدن همه بیت‌ها

هر جستجو پاسخ مثبت (کاذب)
- انتخاب m وابسته به به تعداد (تخمینی) موردانتظار n
- ضرورت بزرگ بودن m نسبت به n

نحوهٔ محاسبهٔ طول فیلتر m
- با توجه به احتمال مثبت کاذب pr_{fp} و تعداد مورد انتظار n

$$m = -\frac{n \ln \mathrm{pr}_{fp}}{(\ln 2)^2}$$

فیلتر بلوم-ویژگی‌ها

نتیجه: در صورت اصرار بر حفظ میزان احتمال مثبت کاذب

- نیاز به رشد خطی فیلتر با نسبتی خطی از تعداد عناصر

امکان تنظیم احتمال مثبت کاذب با داشتن نسبت $\frac{m}{n}$

- به معنای تعداد بیت‌های تخصیص داده شده به ازای هر مقدار
- تنظیم با انتخاب تعداد توابع درهم‌ساز k

انتخاب بهینه k با هدف کمینه کردن احتمال مثبت کاذب

$$k = \frac{m}{n} \ln 2$$

تعداد بهینه توابع درهم‌ساز k حدود ۰,۷ تعداد بیت‌ها به ازای هر عنصر

ترجیح مقادیر زیربهینه کوچکتر به دلیل نیاز به عدد صحیح بودن k

فیلتر بلوم-ویژگی‌ها

برخی از راه‌حل‌های نزدیک به بهینه که با استفاده فراوان

pr_fp	m/n	k
0.0561	6	4
0.0215	8	6
0.00314	12	8
0.000458	16	11

فیلتر بلوم-ویژگی‌ها

مثال- تخمین پارامترها

احتمال مثبت کاذب $\mathrm{pr}_{fp} = ۱\%$

- نیاز است فیلتر ده برابر طولانی‌تر از تعداد عناصر مورد انتظار n
- استفاده از شش تابع درهم‌ساز
- عدم وابستگی طول فیلتر به اندازه خود مقادیر ورودی و ثابت ماندن برای مقادیر با ماهیت‌های مختلف

فیلتر بلوم-ویژگی‌ها

فیلترهای بلوم به مثابهٔ تعمیمی از جدول‌های درهم

فیلتر با یک تابع درهم‌ساز معادل جدول درهم‌ساز

امکان ثابت نگه‌داشتن احتمال مثبت کاذب با استفاده از توابع درهم‌ساز متعدد در فیلترهای بلوم حتی برای تعداد ثابتی از بیت‌ها به ازای هر عنصر

- عدم امکان در جدول‌های درهم

فیلتر بلوم-ویژگی‌ها

ب- منفی کاذب امکان‌ناپذیر

- اگر خروجی فیلتر بلوم عدم عضویت عنصری باشد، عدم وجود قطعی مقدار مذکور

$$\mathrm{pr}_{fn} = ۰$$

فیلتر بلوم-ویژگی‌ها

ج- فیلتر بلوم مناسب برای استفاده در حافظه داخلی

کاهش احتمال مثبت کاذب با تخصیص حافظه بیشتر

تمایل به ایجاد فیلترهای بزرگتر (با m بزرگتر)

به محض بزرگ شدن بیش از حد، لاجرم و به اجبار انتقال به دیسک

چنین انتقالی با مشکل ناشی از طراحی

نیاز به دسترسی تصادفی با تولید شاخص‌های تصادفی در توابع درهم‌ساز با توزیع یکنواخت

ناکارآمدی نیاز مذکور برای دیسک‌های با صفحه‌های گردان و نوک‌های خواندن نوشتن متحرک

- همین‌طور برای دستگاه‌های ذخیره‌سازی حالت جامد

فیلتر بلوم-ویژگی‌ها

مثال- حافظه مورد نیاز: با توجه به معادله تخمین m، جهت مدیریت یک میلیارد عنصر و حفظ احتمال رویدادهای مثبت کاذب حدود دو درصد و استفاده از تعداد بهینه‌ای از توابع درهم‌ساز،

فیلتر بلوم-ویژگی‌ها

مثال- حافظه مورد نیاز: با توجه به معادله تخمین m، جهت مدیریت یک میلیارد عنصر و حفظ احتمال رویدادهای مثبت کاذب حدود دو درصد و استفاده از تعداد بهینه‌ای از توابع درهم‌ساز،

نیاز به انتخاب فیلتری به طول $m = -10^9 \times \frac{\ln 0.02}{(ln\ 2)^2} \approx 8.14 \times 10^9$ بیت انتخاب کنیم

فیلتر بلوم-ویژگی‌ها

مثال- حافظه مورد نیاز: با توجه به معادله تخمین m، جهت مدیریت یک میلیارد عنصر و حفظ احتمال رویدادهای مثبت کاذب حدود دو درصد و استفاده از تعداد بهینه‌ای از توابع درهم‌ساز،

نیاز به انتخاب فیلتری به طول $m = -10^9 \times \frac{\ln 0.02}{(ln\ 2)^2} \approx 8.14 \times 10^9$ بیت انتخاب کنیم

تقریباً برابر با یک گیگابایت حافظه

فیلتر بلوم-ویژگی‌ها

ادغام دو فیلتر بلوم متفاوت با طول یکسان

- تنها در صورتی‌که دارای توابع درهم‌ساز یکسان
- تحویل ادغامْ به عملیات یای بیتی
- حاصل آن با فیلتر بلوم حاصل از اتحاد آن دو مجموعه از عناصر کاملاً متناظر

اشتراک دو فیلتر بلوم نیز ممکن است و با and بیتی صورت می‌پذیرد (چرا؟)

- با این حال، نتیجه امکان احتمال تصادم بیشتر

در صورت اتمام فضای فیلتر بلوم و نیاز به افزایش فضای آن

- ناممکن بودن بدون محاسبه مجدد تمام درهم‌هایی که قبلا در فیلتر قرار گرفتند
- ناممکن در کابردهای داده بزرگ

فیلتر بلوم-ویژگی‌ها

مثال- اشتراک کش

فهرستی از پروکسی‌های کش توزیع شده $(P_1,P_2, \dots ,P_n)$در شبکه‌ای که کش‌های خود را به اشتراک می‌گذارند

در صورت ذخیرهٔ محتوای URL درخواستی روی پروکسی P_i

- برگرداندن آن در پروکسی بدون تماس واقعی با سرور راه دور
- در غیر این صورت، بازیابی محتوا، ذخیرهٔ محلی، و ارسال به مشتری

با هدف به کمینه‌سازی ترافیک شبکه و توزیع ذخیره‌سازی، می‌توان مسیریابی را در شبکه پروکسی تنظیم و سعی کنیم به سمت پروکسی‌ای هدایت کنیم که قبلاً محتوا را ذخیره کرده است. در غیر این صورت، با سرور راه دور تماس بگیریم. از آنجایی که درخواست‌های مشتری می‌توانند به هر پروکسی بیایند، مشکل اشتراک‌گذاری مسیریابی در هر پروکسی وجود دارد و زمانی که تغییر می‌کند، تبادل آن در شبکه یا ادغام، در صورت لزوم، وجود دارد.

فیلتر بلوم

- انتخابی طبیعی برای ذخیرهٔ چنین فهرست‌های مسیریابی و انجام پرسش‌های عضویت سریع
- به دلیل اندازهٔ کوچک خود به راحتی در شبکه قابل انتقال

فیلتر بلوم-ویژگی‌ها

رویداد مثبت کاذب، در این مورد، این است که فرضا یکی از پروکسی‌ها مثلا P_i فرض کند که پروکسی دیگری P_j ممکن است محتوای URL درخواستی را داشته باشد، اما در واقع ندارد. P_i ترافیک را به P_j هدایت می‌کند و از آن می‌خواهد محتوا را برگرداند، بنابراین P_j باید با سرور راه دور تماس بگیرد. در نتیجه، مقداری ترافیک شبکه اضافی تولید می‌کند و کپی‌های محلی اضافی را برای چنین محتوایی ذخیره می‌کند، که کاملاً قابل قبول است.

فیلتر بلوم-ویژگی‌ها

د- حذف امکان‌پذیر نیست.

برای حذف عنصری خاص از فیلتر بلوم نیاز به پاکسازی بیت‌های k متناظر آن در آرایه بیتی

امکان مطابقت یک بیت واحد با چند مقدار

- به دلیل تصادم‌های درهم و بیت‌های مشترک بین مقادیر

معرفی چند بهبود جهت پشتیبانی حذف

- لنگیدن یک پای ماجرا
- تحمیل هزینه‌هایی از نظر فضا و سرعت

به همین دلیل است که فیلتر بلوم کلاسیک بسیار سریع و کارآمد از نظر فضا است. نکته مثبت این است که عدم پشتیبانی از حذف در بسیاری از برنامه‌های کاربردی دنیای واقعی مشکلی ایجاد نمی‌کند، اما اگر واقعاً به آن نیاز باشد، چاره‌ای جز به بهبود فیلتر بلوم مانند «فیلتر بلوم شمارنده» نیست.

شمارش عناصر منحصربه‌فرد در فیلتر

روشی برای تخمین تعداد عناصر منحصربه‌فرد نمایه شده در فیلتر

اس. جاشوا سامیاس و پیر بالدی

- در واقع، بهبودی بر «الگوریتم شمارش خطی» است که در بخش‌های بعد بحث

با استفاده از اطلاعات مربوط به تعداد بیت‌های تنظیم شده در فیلتر و احتمال تنظیم هر بیت

- عرضهٔ معادله ساده‌ای برای تقریب تعداد عناصر موجود در فیلتر
- عدم تغییر بیت‌های تنظیم شده در ازای افزودن دو مقدار برابر به فیلتر
- ارائهٔ چنین تقریبی تخمینی برای تعداد عناصر منحصربه‌فرد (که به عنوان تعداد اصلی شناخته می‌شود)

شمارش عناصر منحصربه‌فرد در فیلتر

الگوریتم- شمارش عناصر منحصربه‌فرد در فیلتر بلوم

ورودی: فیلتر بلوم به طول m با k تابع درهم‌ساز

خروجی: تعداد عناصر منحصربه‌فرد در فیلتر

N برابر شمارش Bloomfilter[j] برای $j = ۱, \dots, m$

اگر $N < k$

برگرداندن صفر

اگر N==k

برگرداندن یک

اگر N==m

برگرداندن m/k

برگرداندن $-\frac{m}{k}.\ln\left(۱ - \frac{N}{m}\right)$

شمارش عناصر منحصربه‌فرد در فیلتر

احتمال صفر بودن بیتی پس از درج n عضو جدید برابر است با

$$\text{pr}(\text{bit} = 0) = \left(1 - \frac{۱}{m}\right)^{kn} \approx e^{-kn/m}$$

در نتیجه، میانگین نسبت یک‌ها برابر است با

$$\frac{E[N]}{m} = 1 - e^{-kn/m}$$

و N تعدد بیت‌های متناظر یک در فیلتر

جهت تقریب n از N می‌توان از $N/m = 1 - e^{-kn/m}$ مقدار زیر را بدست آورد:

$$e^{-kn/m} = 1 - \frac{N}{m}$$

شمارش عناصر منحصربه‌فرد در فیلتر

$$e^{-kn/m} = 1 - \frac{N}{m}$$
$$-\frac{kn}{m} = \ln\left(1 - \frac{N}{m}\right)$$
$$n = -\frac{m}{k}\ \ln\left(1 - \frac{N}{m}\right)$$

که دقیقا برابر مقداری است که الگوریتم وقتی $0 < N < m$ برمی‌گرداند. حالت‌های خاص آن عبارت است از الف- $N < k$ که غیرممکن است مگر $n = 0$. ب- $N = k$ که نشان می‌دهد احتمالا یک عضو افزوده شدن. ج- $N = m$ یا تمامی بیت‌ها برابر ۱ است. در نتیجه ظرفیت بیشینه یا m/k عضو دارد.

معادلهٔ $-\frac{m}{k}\ln(1 - N/m)$ تقریبی مناسب از n در مواقعی است که فیلتر اشباع نشده است.

فیلتر بلوم شمارنده

معروف‌ترین بهبود بر فیلتر بلوم کلاسیک
- پشتیبانی از حذف
- لی فن، پی کائو، جوسارا آلمیدا، و آندری برودر در سال ۱۳۷۹
- با تکیه بر الگوریتم فیلتر بلوم کلاسیک،

آرایه‌ای از m شمارنده $\{C_j\}_{j=1}^{m}$
- متناظر با هر بیت در آرایه فیلتر

با افزوده شدن هر عنصر افزایش یک واحدی مدخل‌های متناظر
- در نتیجه، فراهم کردن تقریبی از تعداد دفعاتی مشاهدهٔ هر عنصر در فیلتر

داده‌ساختار مرتبط
- شامل آرایه‌ای بیتی و آرایه‌ای از شمارنده‌ها هر دو به طول m
- همگی مقداردهی اولیه به صفر

فیلتر بلوم شمارنده

هنگام درج مقداری جدید

- محاسبهٔ موقعیت‌های بیت متناظر آن
- افزایش شمارنده متناظر برای هر موقعیت
- تغییر آرایهٔ بیتی از صفر به یک با تغییر شمارنده از صفر به یک

فیلتر بلوم شمارنده

الگوریتم- افزودن عنصر به فیلتر بلوم شمارنده

ورودی: عنصر $x \in D$

ورودی: فیلتر بلوم شمارنده با m شمارنده $\{C_j\}_{j=۱}^{m}$ و k تابع درهم‌ساز $\{h_i\}_{i=۱}^{k}$

برای i از ۱ تا k انجام بده

$j \leftarrow h_i(x)$

$C_j \leftarrow C_j + ۱$

اگر $C_j == ۱$

$CountingBloomFilter[j] \leftarrow ۱$

فیلتر بلوم شمارنده

عملیات آزمون دقیقاً همانند فیلتر بلوم

- عدم نیاز به به بررسی شمارنده‌ها

به دلیل آرایه‌های بیتی یکسان فیلترها

- دارای زمان برابر برای آزمون عنصری با الگوریتم کلاسیک

فیلتر بلوم شمارنده

الگوریتم- آزمایش عنصر در فیلتر بلوم شمارنده

ورودی: عنصر $x \in D$

ورودی: فیلتر بلوم شمارنده با k تابع درهم‌ساز $\{h_i\}_{i=۱}^{k}$

خروجی: غلط درصورت نیافتن عنصر و درست در صورت وجود عنصر

برای i از ۱ تا k انجام بده

$j \leftarrow h_i(x)$

اگر $CountingBloomFilter[j] \neq ۱$

برگرداندن غلط

برگرداندن درست

فیلتر بلوم شمارنده

حذف برعکس درج

حذف عنصر x،

- محاسبهٔ تمامی k مقدار درهم $\{h_i\}_{i=1}^{k}$
- کاهش شمارنده‌های متناظر

درصورت تغییر مقدار شمارنده از یک به صفر

- حذف بیت متناظر در آرایه بیتی

فیلتر بلوم شمارنده

الگوریتم- حذف عنصر از فیلتر بلوم شمارنده

ورودی: عنصر $x \in D$

ورودی: فیلتر بلوم شمارنده با m شمارنده $\{C_j\}_{j=۱}^{m}$ و k تابع درهم‌ساز $\{h_i\}_{i=۱}^{k}$

برای i از ۱ تا k انجام بده

$j \leftarrow h_i(x)$

$C_j \leftarrow C_j - ۱$

اگر $C_j == ۰$

$CountingBloomFilter[j] \leftarrow ۰$

فیلتر بلوم شمارنده

فرض الگوریتم حذف عنصر
- وجود مقدار x در فیلتر
- یا امکان وجود

نیاز به الگوریتم آزمون عضویت پیش از کاهش شمارنده‌های متناظر

وارث برحق!
- به ارث بردن تمام ویژگی‌های فیلتر بلوم معمول از جمله تخمین خطای مثبت کاذب و معادلات انتخاب بهینه m و k

فیلترهای بلوم شمارنده بسیار بزرگتر از فیلترهای بلوم کلاسیک
- به دلیل نیاز به حافظه اضافی برای شمارنده‌ها ولو اینکه بیشتر مقادیر برابر صفر
- اهمیت تخمین میزان بزرگ شدن چنین شمارنده‌هائی
- وابستگی اندازهٔ آنها به طول فیلتر m و تعداد توابع درهم‌ساز k

با فرض هر شمارنده C دارای سطح ظرفیت N
- احتمال فراتر رفتن مقدار از N (به معنای سرریزی) با طول m با انتخاب بهینه k برابر مقدار

$$pr(\max(C) \geq N) \leq m \times \left(\frac{eln2}{N}\right)^N$$

فیلتر بلوم شمارنده

مثال- شمارنده‌های ۴ بیتی $N = 16$، احتمال سرریز با معادله بالا برابر است با

فیلتر بلوم شمارنده

مثال- شمارنده‌های ۴ بیتی $N = 16$، احتمال سرریز با معادله بالا برابر است با

$$pr(\max(C) \geq 16) \leq m \times 1.37 \times 10^{-15}$$

فیلتر بلوم شمارنده

مثال- شمارنده‌های ۴ بیتی $N = 16$، احتمال سرریز با معادله بالا برابر است با

$$pr(\max(C) \geq 16) \leq m \times 1.37 \times 10^{-15}$$

به عبارت دیگر، اگر ۴ بیت به ازای هر شمارنده تخصیص دهیم، احتمال سرریز برای مقادیر عملی m (به عنوان مثال، چند میلیارد موقعیت بیتی) در طول درج اولیه در فیلتر بسیار کم است.

فیلتر بلوم شمارنده

مثال- شمارنده‌های ۴ بیتی $N = 16$، احتمال سرریز با معادله بالا برابر است با

$$pr(\max(C) \geq 16) \leq m \times 1.37 \times 10^{-15}$$

به عبارت دیگر، اگر ۴ بیت به ازای هر شمارنده تخصیص دهیم، احتمال سرریز برای مقادیر عملی m (به عنوان مثال، چند میلیارد موقعیت بیتی) در طول درج اولیه در فیلتر بسیار کم است.

پس از حذف و درج‌های متعدد، احتمال می‌تواند کمی بیشتر شود، اما هنوز برای استفاده عملی به اندازه کافی کم است.

فیلتر بلوم شمارنده

جلوگیری از سرریز حسابی

- یا افزایش شمارنده‌ای که قبلاً حداکثر مقدار ممکن را دارد

هر شمارنده در آرایه به اندازه کافی بزرگ باشد تا ویژگی‌های فیلترهای بلوم حفظ شود.

- در عمل، شمارنده از ۴ یا چند بیت تشکیل شده است و فیلتر بلوم شمارنده
- نیاز به چهار برابر فضای بیشتر نسبت به فیلتر بلوم کلاسیک

فرارفتن مقدار شمارنده چهار بیتی از مقدار ۱۵

- تحویل آن «به حد بالا» و باقی ماندن مقدار در ۱۵

امکان تولید پاسخ منفی پس از حذف‌های زیاد

- صفر شدن شمارنده زمانی که نباید صفر شود
- اما احتمال چنین زنجیره‌ای از رویدادها آنقدر کم است که راه‌اندازی مجدد برنامه و ایجاد دوباره فیلتر بسیار محتمل‌تر

فیلتر بلوم شمارنده

با این حال، امکان طراحی نسخه پیچیده‌تری از فیلتر بلوم شمارنده با شمارنده‌های کوچکتر

- مثلا دو بیتی
- در عوض با اتخاذ رویکردی مشابه آدرس‌دهی بسته در جدول‌های درهم و معرفی جدول درهم ثانویه برای مدیریت شمارنده‌های سرریز

بنابراین، فیلتر بلوم شمارنده به دلیل احتمال مثبت خطا به دلیل سرریز شدن شمارنده صرفا از حذف درست به صورت احتمالی پشتیبانی می‌کند.

با وجود ایرادهای مورد اشاره، فیلترهای بلوم شمارنده در Apache Hadoop و Apache Spark در برنامه‌های MapReduce برای تسریع پردازش مجموعه‌های داده عظیم روی خوشه‌های بزرگ با کمک به کاهش حجم سخت مورد استفاده‌اند.

فیلتر خارج‌قسمت

مایکل بندر و همکاران در سال ۱۳۹۰ معرفی فیلتر خارج قسمت

- پشتیبانی از عملیات اصلی فیلترهای بلوم
- بهبود محلی‌سازی داده
- نیاز به تعداد کمی از دسترسی‌های دیسک پیوسته

دارای عملکرد فضا و زمان مشابه

پشتیبانی از حذف و امکان تغییر اندازه یا ادغام پویا

نام این داده‌ساختار ریشه از خارج‌قسمت نتیجه عملیات تقسیم

فیلتر خارج‌قسمت

فیلتر خارج‌قسمت در ساده‌ترین شکل خود

- جدول درهم با استفاده از کاوش خطی

به جای ذخیره‌سازی مقدارهای کلید در خانه‌ها، مانند جدول درهم خطی کلاسیک،

- ذخیره مقدارهای درهم در فیلتر خارج‌قسمت
  - اصطلاح اثرانگشت به جای درهم

- ذخیرهٔ تکه‌ای از هر درهم در فیلتر خارج‌قسمت
- با امکان بازسازی کامل مقدار درهم به طور قابل اعتماد

فیلتر خارج‌قسمت

در «طیف دقت»

- فیلتر خارج‌قسمت جایی بین جدول درهم و فیلتر بلوم

درهم‌شدن دو مقدار کلید متمایز به یک اثرانگشت یکسان

- فیلتر خارج‌قسمت قادر به تشخیص تفاوت بین آن‌ها به شیوه‌ای که جدول درهم‌سازی می‌تواند، نخواهد بود.
- اما اگر دو مقدار کلید به اثرانگشت‌های متفاوتی درهم شوند، فیلتر خارج‌قسمت قادر خواهد بود آن‌ها را از یکدیگر تشخیص دهد.
- این وضعیت در مورد فیلترهای بلوم صادق نیست،
  - جایی که پرس و جو بر روی یک مقدار کلید با مجموعه‌ای منحصربه‌فرد از k درهم ممکن است یک مثبت کاذب ایجاد نماید.
- فیلتر خارج‌قسمت کارکردی مشابه فیلتر بلوم با طراحی بسیار متفاوت
- با استفاده از اثرانگشت‌های طویل‌تر، فیلترهای خارج‌قسمت می‌توانند میزان مثبت کاذب را کاهش دهند
- اما اثرانگشت‌های طویل‌تر ممکن است فضای زیادی نیز مصرف کنند.

فیلتر خارج‌قسمت

صناعات و ترفندهای مختلفی مورد استفاده فیلتر خارج‌قسمت برای ذخیره‌سازی فشرده اثرانگشت‌ها

قابلیت بازسازی اثرانگشت کامل زمانی مفید واقع می‌شود که بخواهیم مقدارها را حذف کنیم.

- فیلترهای خارج‌قسمت می‌توانند به طور کارآمد حذف را انجام دهند.

فیلتر خارج‌قسمت قادر به تغییر اندازه

- ادغام دو فیلتر خارج‌قسمت در یک فیلتر خارج‌قسمت بزرگ‌تر

ادغام کارآمد، تغییر اندازه کارآمد، و حذف کارآمد

مفید بودن این ویژگی‌ها به ویژه در سیستم‌های توزیع شده پویا

درک و پیاده‌سازی فیلترهای خارج‌قسمت نسبت به فیلترهای بلوم اندکی پیچیده‌تر

- از طرف دیگر، به دلیل نیاز به دسترسی تصادفی چندباره به هر عملیات، کارائی فیلتر بلوم در صورت عدم گنجایش در حافظه اصلی و نیاز به حافظه جانبی سخت افت می‌کند و نامناسب می‌شود.

فیلتر خارج‌قسمت

طراحی فیلتر خارج‌قسمت

- توضیح خارج‌قسمت‌گیری
- توصیف چگونگی فیلتر خارج‌قسمت از بیت‌های متاداده همراه با خارج‌قسمت‌گیری برای صرفه‌جویی در فضا

از راه‌های به نظر آوردن فیلتر خارج‌قسمت

- ، مشاهده آن به عنوان بازی برای صرفه‌جویی یک بیت اینجا و آنجا
- به کارگیری ترفندهای مختلف در این راستا

فیلتر خارج‌قسمت تنها ساختمان داده از این نوع نیست،

- اما برخی از ترفندهایی که در آن استفاده می‌شود می‌توانند به طور کلی در درک ساختمان‌های داده مشابه مبتنی بر جدول‌های درهم فشرده مفید واقع شوند.

خارج‌قسمت‌گیری

خارج‌قسمت‌گیری

- مقدار درهم ورودی
- تقسیم به خارج‌قسمت و باقیمانده

استفاده از خارج‌قسمت برای نمایه‌گذاری در سطل متناظر جدول درهم در فیلتر خارج‌قسمت

ذخیره باقیمانده در خانه متناظر

برای مثال با فرض درهم به طول h بیت و اندازه جدول m معادل ۲ به توان q

- خارج‌قسمت مقداری که با q بیت چپ درهم تعیین می‌شود و باقیمانده نشان‌دهنده r بیت باقیمانده به تعداد h منهای q بیت

مثال- با فرض m=32 و h=11 باشد پس q=5 و r=6

- صرفه‌جویی به دلیل خارج‌قسمت‌گیری به ازای هر خانه معادل $q = 5$ بیت در مثال قبلی

فیلتر خارج‌قسمت

جهت بهبود محلی‌سازی فضایی،

- داده‌ساختار فیلتر خارج قسمت
  - نمایش با جدول درهم‌ساز فشرده با آدرس‌دهی باز با $m = 2^q$ سطل
- ذخیرهٔ باقیمانده f_r در سطل نمایه‌گذاری شده با خارج‌قسمت f_q
- حل تصادم‌های احتمالی با کاوش خطی

فیلتر خارج‌قسمت

الگوریتم- تکنیک خارج‌قسمت‌گیری

ورودی: اثر انگشت f

خروجی: خارج‌قسمت f_q و باقیمانده f_r

$f_r \leftarrow f \% ۲^r$

$f_q \leftarrow \left\lfloor \frac{f}{۲^r} \right\rfloor$

برگرداندن f_q و f_r

فیلتر خارج‌قسمت

درج در فیلتر خارج‌قسمت چگونه

h = len(f) ← تعداد بیت‌های در درهم یا اثر انگشت

q = log2(m) ← فرض بر اندازه درهم توانی از دو

r = h – q

f_q = math.floor(f / 2 ** r) ← q بیت سمت چپ اثرانگشت

f_r = f % 2 ** r) ← r بیت سمت راست اثرانگشت

filter(f_q) = rem ← ذخیرهٔ باقی‌مانده در سطل متناظر خارج قسمت

فیلتر خارج‌قسمت

اگر هیچ دو اثرانگشتی دارای خارج‌قسمت یکسان نباشند

- به معنای عدم تصادم
- در این صورت، اشغال هر باقیمانده در سطل مخصوص به خود
- بازسازی درهم کامل از طریق الحاق بازنمایی دودویی شماره سطل b با بازنمایی دودویی باقیمانده ذخیره شده در سطل b

با توجه به باقیمانده f_r در سطل f_q، بازسازی منحصر به فرد اثر انگشت کامل

$$f = f_q \times 2^r + f_r$$

کمک این ویژگی به تغییر اندازه و ادغام

- عدم امکان بازسازی مقدارهای اصلی
- یادآوری جایی بین جدول‌های درهم نگهدارنده مقدارهای کلید واقعی و فیلترهای بلوم بدون امکان بازسازی مقدارمتناظر درهم‌ها

فیلتر خارج‌قسمت

با این حال

- رایج بودن تصادم در جدول‌های درهم
- حل تصادم در فیلترهای خارج‌قسمت با استفاده از گونه‌ای از کاوش خطی

امکان رخداد پیچیدگی‌هایی

- در نتیجهٔ انتقال برخی از باقیمانده‌ها از سطل اصلی خود به سمت خانه‌های بعدی بر اثر کاوش خطی
  - از دست رفتن ارتباط بین خارج‌قسمت و باقیمانده

برای بازسازی درهم کامل،

- نیاز فیلتر خارج‌قسمت به ازای هر خانه از سه بیت متاداده
- سه بیت در مقابل صرفه‌جویی حدود ۲۰ تا ۳۰ بیت به ازای هر خانه بابت خارج‌قسمت در جدول‌های درهم بزرگ‌تر
  - بهائی اندکی

فیلتر خارج‌قسمت

چند اصطلاح

مفهوم اجرا یا Run:

- هر اجرا توالی پیوسته‌ای از خانه‌های فیلتر خارج‌قسمت
- ذخیرهٔ باقی‌مانده‌ها (یا اثرانگشت‌ها) با خارج‌قسمت یکسان
- ذخیرهٔ همهٔ اثرانگشت‌های دارای تصادمی روی یک سطل مشخص، به‌صورت پشت‌سرهم و به ترتیب صعودیِ باقی‌مانده‌ها

به‌علت وقوع تصادم‌ها و عمل هل دادن یا انتقال هنگام درج

- امکان آغاز اجرا از فاصله‌ای بسیار دورتر از مدخل سطل متناظر خود

مفهوم خوشه

- خوشه رشته‌ای پیوسته از یک یا چند اجرا
- رشته شامل خانه‌ها اشغال شده با باقیمانده
- قرارگیری اولین باقیمانده در خانهٔ اصلیِ درهم‌شدهٔ خود
  - به نام
  - به نام مدخل لنگر و تکیه
  - پایان خوشه یا با مدخلی خالی، یا با شروع خوشه‌ای دیگر

فیلتر خارج‌قسمت

انجام عملیات جست‌وجو در فیلتر باقیمانده و بازیابی اثرانگشت کامل

- نیاز به تفسیر باقیمانده‌ها با بیت‌های متاداده
- آغاز تفسیر از آغاز خوشه (موقعیت لنگر)
- نقش سه بیت متادادهٔ موجود در هر خانه
- ۱. **bucket_occupied**: درهم‌شدن کلیدی روی سطل متناظر
  - مقدار ۱: درهم‌شدگی حداقل یک کلید به این مدخل
  - مقدار ۰: عدم درهم‌شدگی کلیدی به این سطل
- سخن کوتاه، بیت روشنگر تعیین خارج‌قسمت‌های حاضر در خوشه
- ۲. **run_continued**: باقیمانده‌های فعلی ادامهٔ اجرای قبلی یا آغاز اجرای جدید
  - مقدار ۰: باقیمانده فعلی اولین عضو اجرا
  - مقدار ۱: تعلق باقیماندهٔ فعلی به اجرای قبلی
- سخن کوتاه، مرزبندی دقیق اجراها با این بیت
- ۳. **is_shifted**: تشخیص و تبیین قرار گرفتن باقیمانده در خانهٔ اصلی خود یا دچار جابجا شدگی به‌دلیل تصادم
  - مقدار ۰: درهم‌شدن باقیمانده در خانهٔ اصلی
  - مقدار ۱: انتقال باقیمانده به‌سمت خانه‌های بعدی
- یافتن آغاز خوشه با این بیت

فیلتر خارج‌قسمت

- هر سطل شامل سه بیت متادادهٔ bucket _occupied، run_continued، و is_shifted
  - همگی در ابتدا بدون مقدار
  - نقشی مهم در مدیریت داده‌ساختار

فیلتر خارج‌قسمت

شکل- سطل در فیلتر خارج‌قسمت

سطل f_q

Is_shifted منتقل شده یا سرجای خود	Run_continued ادامه اجرا	Bucket_occupiedاشغال سطل
f_r		

درج در فیلتر خارج‌قسمت

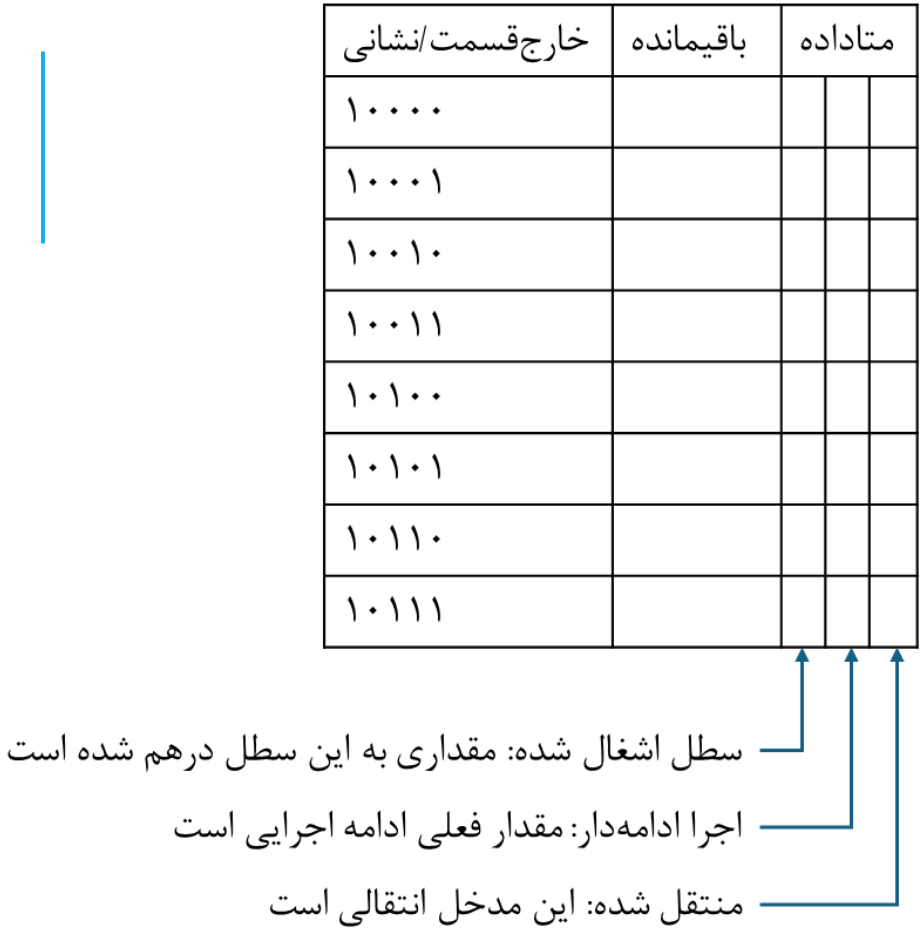
مثال جهت فهم بهتر

بررسی گام به گام فرایند درج مقدارهای ۷، w و x

آغازبا فیلتر خارج‌قسمت در ابتدا خالی با ۳۲ خانه برابر با ۲ به توان ۵

- تنظیم هر خانه و همچنین هر سه بیت متاداده ابتدا به مقدار ۰

درج در فیلتر خارج‌قسمت



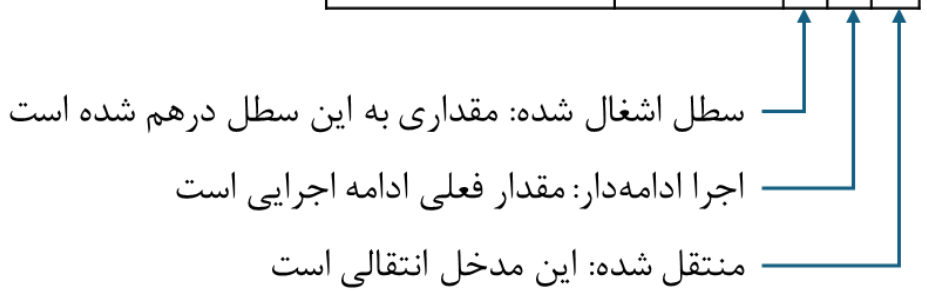
درج در فیلتر خارج‌قسمت

۱ درج v:

- مقدار درهم h(v) برابر با ۱۰۰۰۱۰۰۱۰۱۰
- سطلی که با خارج‌قسمت ۱۰۰۰۱ تعیین می‌شود قبلاً اشغال نشده است.
- بیت bucket_occupied را روی مقدار ۱ تنظیم و ذخیره باقیمانده را در خانه‌ای که با خارج‌قسمت آن تعیین می‌شود
- نیازی به هیچ اقدام اضافی روی سایر بیت‌ها نیست
  - زیرا این مورد در حال حاضر هم آغاز یک اجرا و هم آغاز یک خوشه محسوب می‌شود

درج در فیلتر خارج‌قسمت

خارج‌قسمت/نشانی	باقیمانده	متاداده		
۱۰۰۰۰				
۱۰۰۰۱	۰۰۱۰۱۰	۱	۰	۰
۱۰۰۱۰				
۱۰۰۱۱				
۱۰۱۰۰				
۱۰۱۰۱				
۱۰۱۱۰				
۱۰۱۱۱				



درج در فیلتر خارج‌قسمت

۲ درج w:

مقدار درهم h(w) برابر با ۱۰۰۱۱۱۱۰۰۱۱۱

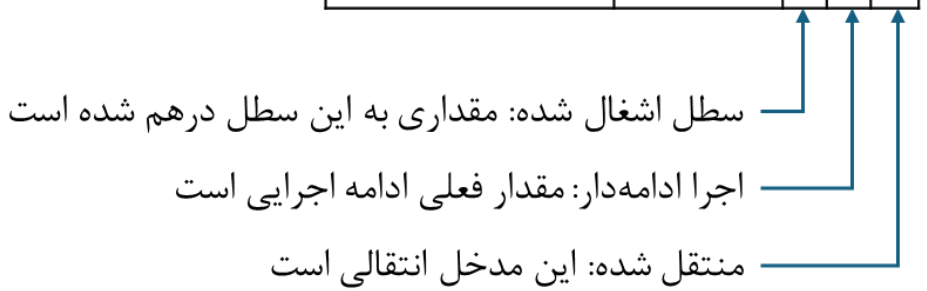
مجدداً بیت bucket_occupied را در خارج‌قسمت ۱۰۰۱۱ روی مقدار ۱ تنظیم

ذخیره باقیمانده در خانه متناظر در دسترس

عدم تغییر بیت‌های دیگر

درج در فیلتر خارج‌قسمت

خارج‌قسمت/نشانی	باقیمانده	متاداده		
۱۰۰۰۰				
۱۰۰۰۱	۰۰۱۰۱۰	۱	۰	۰
۱۰۰۱۰				
۱۰۰۱۱	۱۰۰۱۱۱	۱	۰	۰
۱۰۱۰۰				
۱۰۱۰۱				
۱۰۱۱۰				
۱۰۱۱۱				



درج در فیلتر خارج‌قسمت

۳ درج x:

- مقدار درهم h(x) برابر با ۱۰۰۱۱۱۱۱۱۱۱
- سطل متناظر با خارج‌قسمت ۱۰۰۱۱ هنگامی که سعی می‌کنیم بیت را روی مقدار ۱ تنظیم کنیم از پیش اشغال شده است.
- بنابراین هر جا که باقیمانده را ذخیره کنیم باید بیت run_continued آن خانه را روی مقدار ۱ تنظیم کنیم.
- خانه در سطل درهم‌شده اشغال می‌باشد.
- از این رو هر جا که باقیمانده را ذخیره کنیم باید بیت is_shifted آن خانه را نیز روی مقدار ۱ تنظیم نماییم.
- با توجه به این که در ابتدای خوشه‌ای قرار داریم که فقط یک اجرا دارد (که خارج‌قسمت آن با خارج‌قسمت x برابر است)،
  - پیمایش به سمت پایین تا یافتن نخستین خانه در دسترس درون اجرا در سطل ۱۰۱۰۰
- ذخیره باقیمانده
- تنظیم مقدار ۱ بیت‌های run_continued و is_shifted

درج در فیلتر خارج‌قسمت

خارج‌قسمت/نشانی	باقیمانده	متاداده		
۱۰۰۰۰				
۱۰۰۰۱	۰۰۱۰۱۰	۱	۰	۰
۱۰۰۱۰				
۱۰۰۱۱	۱۰۰۱۱۱	۱	۰	۰
۱۰۱۰۰	۱۱۱۱۱۱	۰	۱	۱
۱۰۱۰۱				
۱۰۱۱۰				
۱۰۱۱۱				

سطل اشغال شده: مقداری به این سطل درهم شده است

اجرا ادامه‌دار: مقدار فعلی ادامه اجرایی است

منتقل شده: این مدخل انتقالی است

درج در فیلتر خارج‌قسمت

در حال حاضر فیلتر خارج‌قسمت

- دارای دو خوشه است که هر خوشه دارای یک اجرا
- در ادامه، درج چند مقدار دیگر

درج در فیلتر خارج‌قسمت

۴ درج y:

- مقدار درهم h(y) برابر با ۱۰۱۰۰۱۰۱۱۰۱
- بیت bucket_occupied در خارج‌قسمت ۱۰۱۰۰ مقدار صفر داشت
- تغییر آن به مقدار ۱
  - بیت run_continued در خانه باقیمانده نهایی روی مقدار ۰ تنظیم خواهد شد
- خانه در سطل درهم‌شده اشغال
  - بیت is_shifted در خانه باقیمانده نهایی روی مقدار ۱ تنظیم خواهد شد
- با شروع از ابتدای خوشه به دنبال نخستین مکان برای ذخیره اجرای جدید خود می‌گردیم.
- نخستین خانه در دسترس در خارج‌قسمت ۱۰۱۰۱ قرار دارد.
- از این رو باقیمانده را ذخیره کرده و بیت‌های متاداده را متناسب با آن تنظیم می‌کنیم.

درج در فیلتر خارج‌قسمت

خارج‌قسمت/نشانی	باقیمانده	متاداده		
۱۰۰۰۰				
۱۰۰۰۱	۰۰۱۰۱۰	۱	۰	۰
۱۰۰۱۰				
۱۰۰۱۱	۱۰۰۱۱۱	۱	۰	۰
۱۰۱۰۰	۱۱۱۱۱۱	۱	۱	۱
۱۰۱۰۱	۱۰۱۱۰۱	۰	۰	۱
۱۰۱۱۰				
۱۰۱۱۱				

سطل اشغال شده: مقداری به این سطل درهم شده است

اجرا ادامه‌دار: مقدار فعلی ادامه اجرایی است

منتقل شده: این مدخل انتقالی است

درج در فیلتر خارج‌قسمت

۵ درج z:

- مقدار درهم h(z) برابر با ۱۰۱۰۰۱۱۱۱۱۰
- بیت bucket_occupied سطل مربوط به z از پیش روی مقدار ۱ تنظیم شده است
  - بیت run_continued در خانه نهایی روی مقدار ۱ خواهد بود
- خانه اصلی اشغال است
  - بیت is_shifted در خانه نهایی روی مقدار ۱ خواهد بود
- با شروع از ابتدای خوشه محل اجرای مناسب خود را پیدا می‌کنیم.
- در امتداد اجرا به سمت پایین تا خارج‌قسمت ۱۰۱۱۰ پویش می‌کنیم تا باقیمانده z را ذخیره کنیم و بیت‌ها را متناسب با آن تنظیم می‌نماییم.

درج در فیلتر خارج‌قسمت

خارج‌قسمت/نشانی	باقیمانده	متاداده		
۱۰۰۰۰				
۱۰۰۰۱	۰۰۱۰۱۰	۱	۰	۰
۱۰۰۱۰				
۱۰۰۱۱	۱۰۰۱۱۱	۱	۰	۰
۱۰۱۰۰	۱۱۱۱۱۱	۱	۱	۱
۱۰۱۰۱	۱۰۱۱۰۱	۰	۰	۱
۱۰۱۱۰	۱۱۱۱۱۰	۰	۱	۱
۱۰۱۱۱				

اجرا

اجرا

خوشه

سطل اشغال شده: مقداری به این سطل درهم شده است

اجرا ادامه‌دار: مقدار فعلی ادامه اجرایی است

منتقل شده: این مدخل انتقالی است

درج در فیلتر خارج‌قسمت

به دلیل ترتیب صعودی مقادیر درج شده، عمل درج نسبتاً ساده

امکان ترتیب مرتب شده درج‌ها در فیلتر خارج‌قسمت پیرنگی

- اما غالب نیست

همچنین، باید توانایی مدیریت پیرنگ‌هائی ممکن باشد که در آن اثرانگشت‌های درج شده به ترتیب دلخواه و نامرتب می‌رسند.

درج در فیلتر خارج‌قسمت

درج خارج از ترتیب مرتب شده اثرانگشت‌ها

پس از یافتن اجرای صحیح برای باقیمانده‌ای که قرار است در آن درج شود

امکان جابجایی چندین مورد از آن اجرا و سایر مقدارهای موجود در آن خوشه به خانه‌های بعدی

نمونه درج یک مقدار با نام a با مقدار درهم h(a) برابر با ۱۰۱۰۰۰۰۰۰۰۰

این مقدار متعلق به دومین اجرای دومین خوشه

- اشغال خانه‌های تعیین شده با خارج‌قسمت‌های ۱۰۱۰۱ و ۱۰۱۱۰
- جابجایی یک خانه کل اجرا با مقدار a به سمت پایین
- ذخیره در خانه ۱۰۱۰۱
  - زیرا باقیمانده آن نخستین مقدار در ترتیب صعودی مرتب شده در آن اجرا محسوب می‌شود
  - در نتیجه همچنین باعث ایجاد تغییرات در بیت‌های متاداده می‌گردد.
- باقیمانده‌ها باید در درون یک اجرا مرتب شده باشند و همچنین اجراها در میان خوشه‌ها مرتب باشند
  - چرا؟

درج در فیلتر خارج‌قسمت

متاداده			باقیمانده	خارج‌قسمت/نشانی
				۱۰۰۰۰
۰	۰	۱	۰۰۱۰۱۰	۱۰۰۰۱
				۱۰۰۱۰
۰	۰	۱	۱۰۰۱۱۱	۱۰۰۱۱
۱	۱	۱	۱۱۱۱۱۱	۱۰۱۰۰
۱	۰	۰	۰۰۰۰۰۰	۱۰۱۰۱
۱	۱	۰	۱۰۱۱۰۱	۱۰۱۱۰
۱	۱	۰	۱۱۱۱۱۰	۱۰۱۱۱

اجرا

اجرا

خوشه

سطل اشغال شده: مقداری به این سطل درهم شده است

اجرا ادامه‌دار: مقدار فعلی ادامه اجرایی است

منتقل شده: این مدخل انتقالی است

درج در فیلتر خارج‌قسمت

اهمیت کوچک بودن اندازه خوشه‌ها

- نیاز بالقوه به جابجایی کل یک خوشه از مقادیر در حین درج یا حذف
- همچنین رمزگشایی کل یک خوشه در حین انجام عملیات جستجو

هر چه فضای خالی بیشتری

- کاهش احتمال تشکیل خوشه‌های بزرگ که عملیات درج و جستجو نیاز به پویش و رمزگشایی در آن‌ها داشته باشد
- درست مانند جدول‌های درهم معمولی با کاوش خطی،
  - فیلترهای خارج‌قسمت زمانی سریع‌تر کار می‌کنند که ضریب بار در محدوده ۷۵ تا ۹۰ درصد نگه داشته شود

فیلتر خارج قسمت

فیلتر خارج‌قسمت مجموعه داده $D = \{x_1, x_2, \dots, x_n\}$

با ذخیره اثر انگشتی p-بیتی برای هر یک از اعضاء مجموعه داده

وابستگی به تابع درهم‌ساز برای تولید چنین اثر انگشت‌هایی

جهت پشتیبانی از تصادفی‌سازی مناسب

- نیاز به تابع درهم‌ساز با تولید اثر انگشت‌هایی با توزیع یکنواخت و مستقل
- هر اثر انگشت f در الگوریتم تقسیم به دو بخش
  - استفاده از تکنیک خارج‌قسمت‌گیری
  - q بیت با بیشترین اهمیت (خارج‌قسمت)
  - r بیت با کمترین اهمیت (باقیمانده)
- پیشنهاد دونالد کنوث

فیلتر خارج قسمت

بهبود محلی‌سازی فضایی
- داده‌ساختار فیلتر خارج قسمت
- نمایش با جدول درهم‌ساز فشرده با آدرس‌دهی باز با $m = 2^q$ سطل
  - ذخیرهٔ باقیمانده f_r در سطل نمایه شده با خارج‌قسمت f_q
  - حل تصادم‌های احتمالی با کاوش خطی

وجود باقیمانده f_r در سطل f_q،
- بازسازی منحصربه‌فرد اثر انگشت کامل

$$f = f_q \times 2^r + f_r$$

دو اثر انگشت متفاوت f و f' با خارج‌قسمت یکسان $(f_q = f'_q)$
- تصادم نرم
  - حل با کاوش خطی
- پیاده‌سازی با ذخیره پشت سر هم تمام باقیمانده‌های اثر انگشت‌ها با خارج‌قسمت یکسان در اجرا
- امکان انتقال باقیمانده از مکان اصلی خود به خانه‌های بعدی و ذخیره در سطل بعدی
  - حلقه در انتهای آرایه

فیلتر خارج

الگوریتم ۸- استفاده از جابجایی به راست جهت تخلیهٔ سطل‌ها

۱	ورودی: نمایه سطل i	
۲	ورودی: فیلتر خارج‌قسمت به طول m	
۳	$qbl \leftarrow QF[i]$	
۴	$k \leftarrow i + ۱$	
۵	تازمانی که درست (True) انجام بده	
۶	اگر $k \geq m$ آنگاه	
۷	$k \leftarrow ۰$	
۸	اگر $QF[k] = \phi$ آنگاه	
۹	$QF[k] \leftarrow qbl$	
۱۰	$QF[k].\text{run\_continued} \leftarrow ۱$	هر مقدار منتقل شده ادامه اجراست
۱۱	$QF[i] \leftarrow \phi$	در موقعیت اصلی خود نیست
۱۲	$QF[i].\text{bucket\_occupied} \leftarrow ۰$	
۱۳	برگرداندن QF	
۱۴	ورنه	
۱۵	$knoni \leftarrow QF[k]$	
۱۶	$QF[k] \leftarrow qbl$	
۱۷	$QF[k].\text{run\_continued} \leftarrow ۱$	تبدیل به ادامهٔ اجرا
۱۸	$QF[k].\text{is\_shifted} \leftarrow ۱$	منتقل شده
۱۹	$qbl \leftarrow knoni$	
۲۰	$k \leftarrow k + ۱$	

فیلتر خارج قسمت

خوشه

- دنباله‌ای از یک یا چند اجرای متوالی بدون سطل خالی در بین آنها

قرارگیری همه خوشه‌ها بلافاصله پیش از سطل خالی

- عدم مقداردهی بیت **is_shifted** اولین مقدار آن

فیلتر خارج قسمت

ذخیره جدول درهم داخلی به صورت فشرده در آرایه‌ای

- کاهش حافظه موردنیاز کاهش
- بهبود محلی‌سازی داده

منجر به پیچیده‌تر شدن مدیریت

تابع اسکن یا پیمایش برای یافتن اجرا

- گام‌برداری عقب‌رو از سطل متعارفِ f برای یافتن ابتدای خوشه
- به محض پیدا شدن شروع خوشه
  - گام‌برداری پیش‌رو تا یافتن مکان اولین باقیمانده برای سطل f_q
  - شروع واقعی اجرا آغاز r

فیلتر خ

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت	
۲	$j \leftarrow f_q$.	یافتن آغاز خوشه دارای f_q
۳	تازمانی که $QF[j].is\ shifted == ۱$ انجام بده	
۴	$j--$	
۵	$r_{\text{آغاز}} \leftarrow j$	
۶	تازمانی که $j \neq f_q$	
۷	تازمانی که $QF\left[r_{\text{آغاز}}+۱\right].runContinued == ۱$	
۸	$r_{\text{آغاز}}++$	
۹	$r_{\text{آغاز}}++$	
۱۰	تازمانی که $QF[j+۱].bucktOccupied \neq ۱$	
۱۱	$j++$	
۱۲	$j++$	
۱۳	$r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$	یافتن پایان اجرای دارای f_q
۱۴	تازمانی که $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued == ۱$	
۱۵	$r_{\text{پایان}}++$	
۱۶	برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$	

فیلتر خارج قسمت

درج مقداری جدید

- محاسبهٔ خارج‌قسمت و باقیمانده آن
- در صورت اشغال نبودن سطل متناظر
  - استفاده از روش درج
- در غیر این صورت، قبل از درج،
  - نیاز به یافتن سطلی مناسب با تابع اسکن از الگوریتم قبلی
  - با یافتن سطل مناسب
    - درج واقعی همچنان نیاز به ادغام مناسب f_r در دنباله عناصر ذخیره شده قبلی
    - امکان نیاز به جابجایی به راست مقادیر بعدی و به‌روزرسانی بیت‌های متاداده متناظر
- الگوریتم کامل درج با استراتژی انتخاب برای سطل مناسب و تابع جابجایی به راست در الگوریتم ۸

فیلتر خارج قسمت

الگوریتم ۱۰- درج مقدار در فیلتر خارج‌قسمت

۱	ورودی: عضو $x \in D$	
۲	ورودی: فیلتر خارج‌قسمت با تابع درهم‌ساز h	
۳	$f \leftarrow h(x)$	
۴	$f_q, f_r \leftarrow f$	
۵	اگر $QF[f_q].\text{bucket}Occupied == ۰$ و خالی بودن $QF[f_q]$	
۶	$QF[f_q] \leftarrow f_r$	خالی و اشغال نبودن سطل
۷	$QF[f_q].\text{bucket}Occupied \leftarrow ۱$	
۸	برگرداندن True	
۹	$QF[f_q].\text{bucket}Occupied \leftarrow ۱$	علامت اشغال شدن سطل متناظر
۱۰	$r_{\text{انفاز}}, r_{\text{پایان}} \leftarrow Scan(QF, f_q)$	
۱۱	برای i از انفاز r تا پایان r	
۱۲	اگر $QF[i] == f_r$	به معنای وجود f_r از قبل و لزوم خروج
۱۳	برگرداندن True	
۱۴	وگر $QF[i] > f_r$ آنگاه	درج f_r در سطل i و جابجا کردن بقیه
۱۵	$QF \leftarrow ShiftRight(QF, i)$	
۱۶	$QF[i] \leftarrow f_r$	
	اگر $i \neq r_{\text{انفاز}}$	
۱۷	$QF[i].run\ continued \leftarrow ۱$	اولین نبودن در اجرا
۱۸	اگر $i \neq f_q$	
۱۹	$QF[i].is\_shifted \leftarrow ۱$	
۲۰	برگرداندن True	
۲۱	$QF \leftarrow ShiftRight\left(QF, r_{\text{پایان}} + ۱\right)$	باقیمانده جدید بزرگتر از تمامی مقادیر فعلی در اجرا، درج در انتهای اجرا و بسط همان اجرا
۲۲	$QF\left[r_{\text{پایان}} + ۱\right] \leftarrow f_r$	
۲۳	$QF[i].run\ continued \leftarrow ۱$	
۲۴	اگر $r_{\text{پایان}} + ۱ \neq f_q$	
۲۵	$QF[r_{\text{پایان}} + ۱].is\_shifted \leftarrow ۱$	
۲۶	برگرداندن True	

فیلتر خارج قسمت

$f_q = 12$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱ ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت

۲ $j \leftarrow f_q$

۳ تازمانی که $QF[j].is\ shifted == ۱$ انجام بده

۴ $j--$

۵ $r_{\text{آغاز}} \leftarrow j$

۶ تازمانی که $j \neq f_q$

۷ تازمانی که $QF\left[r_{\text{آغاز}} + ۱\right].runContinued == ۱$

۸ $r_{\text{آغاز}}++$

۹ $r_{\text{آغاز}}++$

۱۰ تازمانی که $QF[j + ۱].bucktOccupied \neq ۱$

۱۱ $j++$

۱۲ $j++$

۱۳ $r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$

۱۴ تازمانی که $r_{\text{پایان}} + ۱ \leq m\ \&\ QF\left[r_{\text{پایان}} + ۱\right].runContinued == ۱$

۱۵ $r_{\text{پایان}}++$

۱۶ برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

فیلتر خارج قسمت

گام ۱
- پیدا کردن شروع خوشه
- گذر به چپ تا زمان وجود انتقال

$$j = f_q = 12$$

اندیس ۱۲: is_shifted = 1 پس $j = 11$

اندیس ۱۱: is_shifted = 1 پس $j = 10$

اندیس ۱۰: is_shifted = 0 پس توقف

$$r_{start} = j = 10$$

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت
۲	$j \leftarrow f_q$
۳	تازمانی که $QF[j].is\ shifted == ۱$ انجام بده
۴	$j--$
۵	$r_{آغاز} \leftarrow j$
۶	تازمانی که $j \neq f_q$
۷	تازمانی که $QF\left[r_{آغاز}+۱\right].runContinued == ۱$
۸	$r_{آغاز}++$
۹	$r_{آغاز}++$
۱۰	تازمانی که $QF[j+۱].bucktOccupied \neq ۱$
۱۱	$j++$
۱۲	$j++$
۱۳	$r_{پایان} \leftarrow r_{آغاز}$
۱۴	تازمانی که $r_{پایان}+۱ \leq m\ \&\ QF\left[r_{پایان}+۱\right].runContinued == ۱$
۱۵	$r_{پایان}++$
۱۶	برگرداندن $r_{آغاز}$ و $r_{پایان}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

گام ۲

- گذر از اجراها تا رسیدن j به $f_q = ۱۲$
- تکرار اول ($j = 10$، $r_{start} = 10$، $12 \neq 10$)
- زیرگام ۲ الف
- یافتن r_{start}
- $r_{start} = 10$
- $r_{start}+1 = 11$ بررسی runContinued=1
  - پس r_start=11
- $r_{start}+1 = 12$ بررسی runContinued =1
  - پس r_start=12
- $r_{start}+1 = 13$ بررسی runContinued =0
- پس توقف
- r_start = 13

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت
۲	$j \leftarrow f_q$
۳	تازمانی که $QF[j].is\ shifted == ۱$ انجام بده
۴	$j--$
۵	$r_{\text{آغاز}} \leftarrow j$
۶	تازمانی که $j \neq f_q$
۷	تازمانی که $QF\left[r_{\text{آغاز}}+۱\right].runContinued == ۱$
۸	$r_{\text{آغاز}}++$
۹	$r_{\text{آغاز}}++$
۱۰	تازمانی که $QF[j+۱].bucktOccupied \neq ۱$
۱۱	$j++$
۱۲	$j++$
۱۳	$r_{\text{آغاز}} \leftarrow r_{\text{پایان}}$
۱۴	تازمانی که $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued == ۱$
۱۵	$r_{\text{پایان}}++$
۱۶	برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

زیرگام ۲ ب:

انتقال j به سطل اشغال شدهٔ بعدی

- $j = 10$
- $j+1 = 11$ بررسی bucket_occupied=1
- پس توقف

$$j = 11$$

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱ ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت

۲ $j \leftarrow f_q$

۳ تازمانی که ۱ == $QF[j].is\ shifted$ انجام بده

۴ $j--$

۵ $r_{\text{آغاز}} \leftarrow j$

۶ تازمانی که $j \neq f_q$

۷ تازمانی که ۱ == $QF\left[r_{\text{آغاز}}+۱\right].runContinued$

۸ $r_{\text{آغاز}}++$

۹ $r_{\text{آغاز}}++$

۱۰ تازمانی که ۱ ≠ $QF[j+۱].bucktOccupied$

۱۱ $j++$

۱۲ $j++$

۱۳ $r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$

۱۴ تازمانی که ۱ == $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued$

۱۵ $r_{\text{پایان}}++$

۱۶ برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

پایان تکرار اول

$j = 11$

$r_{start} = 13$

12≠11 پس ادامه

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت
۲	$j \leftarrow f_q$
۳	تازمانی که ۱ == $QF[j].is\ shifted$ انجام بده
۴	$j--$
۵	$r_{\text{آغاز}} \leftarrow j$
۶	تازمانی که $j \neq f_q$
۷	تازمانی که ۱ == $QF\left[r_{\text{آغاز}}+۱\right].runContinued$
۸	$r_{\text{آغاز}}++$
۹	$r_{\text{آغاز}}++$
۱۰	تازمانی که ۱ ≠ $QF[j+۱].bucktOccupied$
۱۱	$j++$
۱۲	$j++$
۱۳	$r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$
۱۴	تازمانی که ۱ == $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued$
۱۵	$r_{\text{پایان}}++$
۱۶	برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

- تکرار دوم ($j = 11$ ،$r_{start} = 13$)
- زیرگام ۲ الف
- $r_{start} = 13$
- r_start+1 = 14 بررسی runContinued=1
  - پس r_start=14
- r_start+1 = 15 بررسی runContinued =1
  - پس توقف
- r_start = 15

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت
۲	$j \leftarrow f_q$
۳	تازمانی که ۱ == $QF[j].is\ shifted$ انجام بده
۴	$j--$
۵	$r_{\text{آغاز}} \leftarrow j$
۶	تازمانی که $j \neq f_q$
۷	تازمانی که ۱ == $QF\left[r_{\text{آغاز}}+۱\right].runContinued$
۸	$r_{\text{آغاز}}++$
۹	$r_{\text{آغاز}}++$
۱۰	تازمانی که ۱ ≠ $QF[j+۱].bucktOccupied$
۱۱	$j++$
۱۲	$j++$
۱۳	$r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$
۱۴	تازمانی که ۱ == $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued$
۱۵	$r_{\text{پایان}}++$
۱۶	برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

زیرگام ۲ ب:

- $j = 11$
- $j+1 = 12$ بررسی bucket_occupied=1
- پس توقف

$$j = 12$$

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱	ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت
۲	$j \leftarrow f_q$
۳	تازمانی که ۱ == $QF[j].is\ shifted$ انجام بده
۴	$j--$
۵	$r_{\text{آغاز}} \leftarrow j$
۶	تازمانی که $j \neq f_q$
۷	تازمانی که ۱ == $QF\left[r_{\text{آغاز}}+۱\right].runContinued$
۸	$r_{\text{آغاز}}++$
۹	$r_{\text{آغاز}}++$
۱۰	تازمانی که ۱ ≠ $QF[j+۱].bucktOccupied$
۱۱	$j++$
۱۲	$j++$
۱۳	$r_{\text{آغاز}} \leftarrow r_{\text{پایان}}$
۱۴	تازمانی که ۱ == $r_{\text{پایان}}+۱ \leq m\ \&\ QF\left[r_{\text{پایان}}+۱\right].runContinued$
۱۵	$r_{\text{پایان}}++$
۱۶	برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

فیلتر خارج قسمت

پایان تکرار دوم

$j = 12$

$r_{start} = 15$

$f_q = 12$

- خارج شدن از حلقه

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱ ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت

۲ $j \leftarrow f_q$

۳ تازمانی که $QF[j].is\ shifted == ۱$ انجام بده

۴ $j--$

۵ $r_{آغاز} \leftarrow j$

۶ تازمانی که $j \neq f_q$

۷ تازمانی که $QF\left[r_{آغاز}+۱\right].runContinued == ۱$

۸ $r_{آغاز}++$

۹ $r_{آغاز}++$

۱۰ تازمانی که $QF[j+۱].bucktOccupied \neq ۱$

۱۱ $j++$

۱۲ $j++$

۱۳ $r_{پایان} \leftarrow r_{آغاز}$

۱۴ تازمانی که $r_{پایان}+۱ \leq m\ \&\ QF\left[r_{پایان}+۱\right].runContinued == ۱$

۱۵ $r_{پایان}++$

۱۶ برگرداندن $r_{آغاز}$ و $r_{پایان}$

f_q	اشغال	ادامه	انتقال
10	1	0	0
11	1	1	1
12	1	1	1
13	1	0	1
14	1	1	1
15	0	0	1
16	0	1	1
17	0	0	1
18	0	0	1
19	0	0	0

الگوریتم ۹- پیمایش فیلتر خارج‌قسمت برای یافتن اجرا

۱ ورودی: اندیس سطل متعارف f_q، فیلتر خارج‌قسمت

۲ $j \leftarrow f_q$

۳ تازمانی که $QF[j].is\ shifted == ۱$ انجام بده

۴ $j--$

۵ $r_{\text{آغاز}} \leftarrow j$

۶ تازمانی که $j \neq f_q$

۷ تازمانی که $QF\left[r_{\text{آغاز}} + ۱\right].runContinued == ۱$

۸ $r_{\text{آغاز}}++$

۹ $r_{\text{آغاز}}++$

۱۰ تازمانی که $QF[j+۱].bucktOccupied \neq ۱$

۱۱ $j++$

۱۲ $j++$

۱۳ $r_{\text{پایان}} \leftarrow r_{\text{آغاز}}$

۱۴ تازمانی که $r_{\text{پایان}} + ۱ \leq m\ \&\ QF\left[r_{\text{پایان}} + ۱\right].runContinued == ۱$

۱۵ $r_{\text{پایان}}++$

۱۶ برگرداندن $r_{\text{آغاز}}$ و $r_{\text{پایان}}$

فیلتر خارج قسمت

گام ۳

- یافتن پایان اجرای شروع شده از ۱۵
- $r_{\text{end}} = r_{start} = 15$
- r_end+1 = 16 بررسی runContinued=1
  - پس r_end=16
- r_end+1 = 17 بررسی runContinued =0
- پس توقف

خروجی: (r_end, r_start) = (۱۶,۱۵)

فیلتر خارج قسمت

با توجه به طرح کاوش خطی

- زمان اکثر اجراها از مرتبهٔ $O(1)$
- اشاره نویسندگان فیلتر به احتمال زیاد همه اجراها دارای طول O(log m)

فیلتر خارج قسمت

مثال- افزودن مقادیر به فیلتر

فیلتر خارج‌قسمت با اثر انگشت‌های ۱۶ بیتی حاصل از نسخهٔ ۳۲ بیتی تابع درهم‌ساز مورمور۳

$$h(x) = MurmurHash3(x)\%\ 16$$

سطل‌بندی، رزرو $q = 3$ بیت با بیشترین اهمیت

- تبعا اندازهٔ فیلتر باقیمانده برابر $m = 2^3 = 8$
- ذخیرهٔ $p = 13$ بیت باقیمانده در سطل‌های انتخاب شده

فیلتر خارج قسمت

همانند مثال قبلی نمایه‌سازی نام پایتخت‌ها

کپنهاگ اولین مقدار افزودنی به فیلتر

محاسبهٔ اثر انگشت آن با استفاده از تابع درهم‌ساز **h**

$$f = h(Copenhagen) = 4248224207$$

- خارج‌قسمت و باقیمانده

$$f_q = \left\lfloor \frac{f}{2^{13}} \right\rfloor = 7, f_r = f \% 2^{13} = 490127823$$

فیلتر خارج قسمت

همانند مثال قبلی نمایه‌سازی نام پایتخت‌ها

کپنهاگ اولین مقدار افزودنی به فیلتر

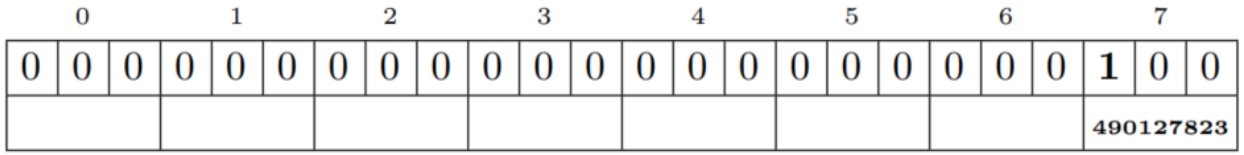
محاسبهٔ اثر انگشت آن با استفاده از تابع درهم‌ساز h

$$f = h(Copenhagen) = 4248224207$$

- خارج‌قسمت و باقیمانده

$$f_q = \left\lfloor \frac{f}{2^{13}} \right\rfloor = 7, f_r = f \% 2^{13} = 490127823$$

سطل متناظر کپنهاگ $j = f_q = 7$



- پر کردن باقیمانده f_r در آن
- درج در این نقطه ساده به دلیل پر نبودن سطل‌ها
  - درج $f_r = 490127823$ در سطل با نمایه ۷ و قرار دادن مقدار یک در بیت bucket_occupied

فیلتر خارج قسمت

مقدار لیسبون

- اثر انگشت f = 629555247 و سطل متناظر ۱،

مقدار پاریس

- اثر انگشت f = 2673248856 و سطل متناظر ۴

به دلیل آزاد بودن سطل‌های متناظر، درج به ترتیب باقیمانده‌ها و تنظیم بیت‌های bucket_occupied

0			1			2			3			4			5			6			7		
0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	1	0	0
			92684335									**525765208**									490127823		

فیلتر خارج قسمت

درج استکهلم

- اثر انگشت $f = 775943400$ و سطل متناظر $j = f_q = 1$ و باقیماندهٔ $f_r = 239072488$
- مقداردهی بیت bucket_occupied سطل متناظر ۱
  - به معنای اشغال با باقیمانده مقداری دیگری (در این مورد لیسبون)

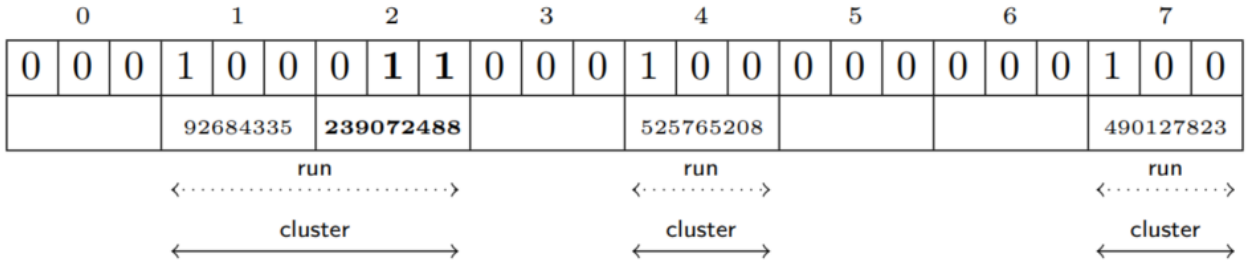
فیلتر خارج قسمت

درج استکهلم

- اثر انگشت f = 775943400 و سطل متناظر $j = f_q = 1$ و باقیماندهٔ $f_r = 239072488$
- مقداردهی بیت bucket_occupied سطل متناظر ۱
  - به معنای اشغال با باقیمانده مقداری دیگری (در این مورد لیسبون)

به دلیل عدم تنظیم بیت‌های is_shifted و run_continued

- در ابتدای خوشه و در عین حال شروع اجرا
- باقیماندهٔ f_r بزرگتر از مقدار نمایه شده قبلی ۹۲۶۸۴۳۳۵
  - پس ذخیره در سطل بعدی موجود، یعنی سطل ۲
  - تنظیم بیت‌های is_shifted و run_shifted آن
  - بدون تغییر ماندن با این حال، بیت bucket_occupied سطل ۲
  - به دلیل عدم تناظر باقیمانده ذخیره‌ای با سطل



فیلتر خارج قسمت

درج زاگرب

اثر انگشت f = 1474643542، سطل متناظر $j = 2$ و باقیمانده $f_r = 400901718$

بیت مقداردهی شده is_shifted نمایشگر اشغال سطل ۲ با مقداری انتقالی

فیلتر خارج قسمت

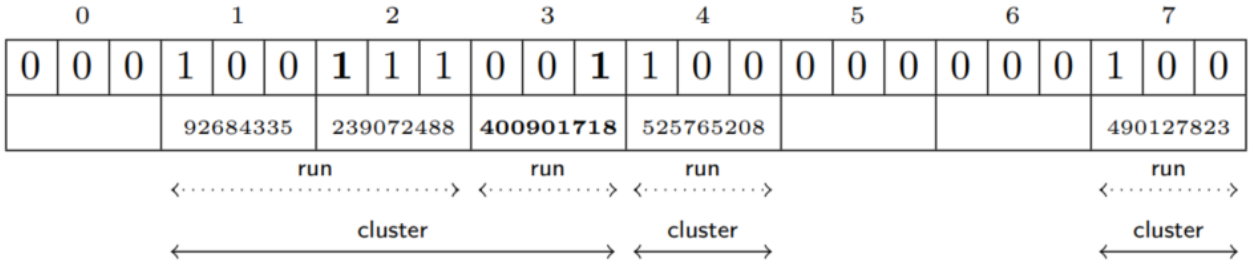
درج زاگرب

اثر انگشت $f = 1474643542$، سطل متناظر $j = 2$ و باقیمانده $f_r = 400901718$

بیت مقداردهی شده is_shifted نمایشگر اشغال سطل ۲ با مقداری انتقالی

عدم تنظیم بیت bucket_occupied

- انتقال مقدار f_r به سمت راست، به سطل بعدی موجود که در این مورد سطل ۳



فیلتر خارج قسمت

مقداردهی بیت run_shifted

- نمایش سطل حاوی مقداری منتقل شده از موقعیت متناظر خود
- عدم تغییر بیت run_shifted به دلیل متناظر بودن اولین عضو آن با سطل مرتبط
- مضافا، مقداردهی بیت bucket_occupied سطل ۲ جهت به یاد داشتن ذخیره شدن حداقل یک باقیمانده مربوط سطل متناظر

درج ورشو

- با اثر انگشت f = 567538184 با خارج قسمت و باقیماندهٔ

$$f_q = \left\lfloor \frac{f}{2^{13}} \right\rfloor = 1$$

$$f_r = f\%2^{13} = 30667272$$

فیلتر خارج قسمت

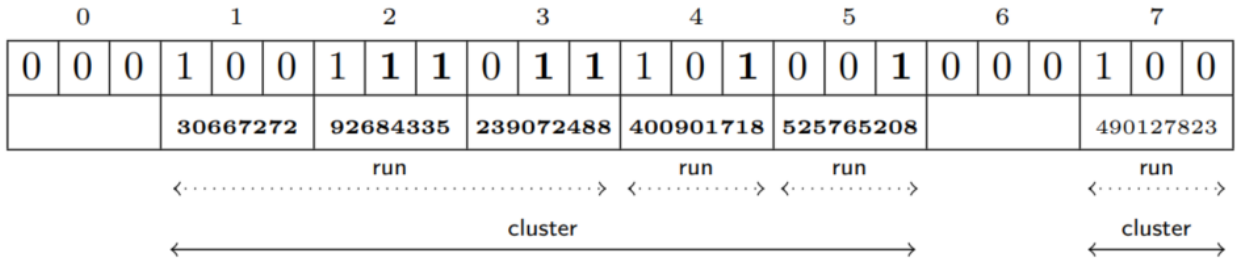
اشغال از قبل سطل متناظر $j = f_q = 1$ بر اساس بیت bucket_occupied

- عدم مقداردهی بیت‌های دیگر
  - به معنای در ابتدای خوشه بودن
- کوچکتر بودن مقدار باقیمانده f_q از مقدار نمایه شدهٔ ۹۲۶۸۴۳۳۵
- در نتیجه نیاز به نمایه شدن در سطل متناظر و انتقال بقیهٔ باقیمانده‌ها و مقداردهی به عنوان ادامه

فیلتر خارج قسمت

اشغال از قبل سطل متناظر $j = f_q = 1$ بر اساس بیت bucket_occupied

- عدم مقداردهی بیت‌های دیگر
  - به معنای در ابتدای خوشه بودن
- کوچکتر بودن مقدار باقیمانده f_q از مقدار نمایه شدهٔ ۹۲۶۸۴۳۳۵
- در نتیجه نیاز به نمایه شدن در سطل متناظر و انتقال بقیهٔ باقیمانده‌ها و مقداردهی به عنوان ادامه



فیلتر خارج قسمت

آزمون عضویت

- همانند درج انجام
- بررسی سطل متناظر
  - مشاهدهٔ وجود باقیمانده متناظر در فیلتر با مشاهده بیت bucket_occupied
  - در صورت یک نبودن مقدار بیت مذکور،
    - عدم وجود مقدار در فیلتر
  - وگرنه، بررسی فیلتر با رویه اسکن جهت یافتن اجرای مناسب سطل
  - مقایسه باقیمانده‌های حاضر در اجرا با باقیمانده مقدار مورد پرسش
    - در صورت یافت شدن برگرداندن احتمال وجود آن در فیلتر

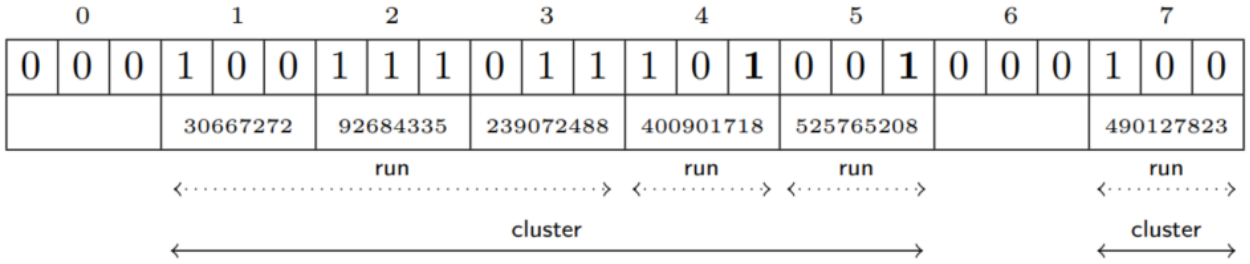
فیلتر

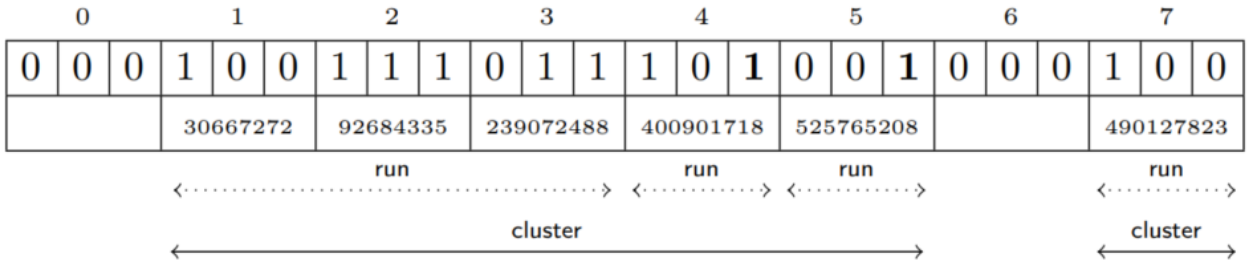
الگوریتم ۱۱- آزمون عنصر در فیلتر خارج‌قسمت		
۱	ورودی: عنصر $x \in D$	
۲	ورودی: فیلتر خارج‌قسمت با تابع درهم‌ساز h	
۳	خروجی: False اگر عنصر یافت نشد و True اگر احتمال حضور وجود داشته باشد	
۴	$f \leftarrow h(x)$	
۵	$f_q, f_r \leftarrow f$	
۶	اگر $QF[f_q].bucket\_occupied \neq ۱$ آنگاه	
۷	برگرداندن False	
۸	در غیر این صورت	
۹	$r_{آغاز}, r_{پایان} \leftarrow Scan(QF, f_q)$	
۱۰	برای i از $r_{آغاز}$ تا $r_{پایان}$ انجام بده	جستجوی f_r در داخل اجرا
۱۱	اگر $QF[i] == f_r$ آنگاه	
۱۲	برگرداندن True	
۱۳	برگرداندن False	

فیلتر خارج قسمت

مثال- آزمون عناصر در فیلتر

داده‌ساختار حاصل مثال قبل





فیلتر خارج قسمت

آزمون عضویت پاریس با خارج‌قسمت $f_q = 4$ و باقیمانده $f_r = 525765208$

- اشغال سطل ۴ از قبل
  - به معنای وجود حداقل یک باقیمانده در جایی از فیلتر با سطل متناظر ۴
  - عدم امکان مقایسه مقدار سطل را با f_r به دلیل تنظیم بیت bucket_shifted
  - در نتیجه در پی اجرایی دارای تطابق با سطل متناظر ۴ در خوشه فعلی
  - اسکن از سطل ۴ به سمت چپ
  - شمارش سطل‌ها را با مقدار بیت‌های bucket_occupied جهت رسیدن به آغاز خوشه
  - آغاز خوشه از سطل ۱ و دو سطل اشغال شده (سطل‌های ۱ و ۲) در سمت چپ سطل ۴ قرار دارند. بنابراین، اجرای مدنظر سومین در خوشه
  - درنتیجه نیاز به اسکن از ابتدای خوشه (سطل ۱) به سمت راست و شمارش سطل‌هایی که بیت‌های run_shifted برابر صفر تا رسیدن به اجرای مدنظر
  - آغاز اجرا از سطل ۵
  - مقایسهٔ مقدار با باقیمانده‌های ذخیره
  - با در نظر گرفتن ترتیب صعودی آنها

مطابقت مقدار موجود در سطل ۵ با باقیمانده $f_r = 525765208$

- امکان وجود پاریس در فیلتر

فیلتر خارج قسمت

حذف از فیلتر خارج‌قسمت

به روشی سخت شبیه به درج مقدار جدید

ذخیره با ترتیب افزایشی باقیمانده‌های اثر انگشت‌ها با خارج‌قسمت یکسان

حذف باقیمانده‌ای از خوشه نیاز به جابجایی تمام اثر انگشت‌ها و اصلاح بیت‌های به ترتیب متاداده

فیلتر خارج قسمت

الگوریتم۱۲- استفاده از انتقال به چپ برای پر کردن سطل‌های خالی

```
۱   ورودی: اندیس سطل i
۲   ورودی: فیلتر خارج‌قسمت به طول m
۳   k ← i + ۱
۴   تازمانی که QF[k] ≠ ∅ انجام بده
۵       QF[k − ۱] ← QF[k]
۶       QF[k − ۱].is_shifted ← QF[k].is_shifted
۷       QF[k] ← ϕ
۸       QF[k].run_continued ← ۰
۹       QF[k].is_shifted ← ۰
۱۰      k ← k + ۱
۱۱      اگر k > m آنگاه
۱۲          k ← ۰
```

فیلتر خارج قسمت

نیاز به بررسی اشغال سطل متناظر

- در صورت عدم مقداردهی عدم وجود قطعی در فیلتر و توقف

در صورت مقداردهی

- اسکن برای یافتن سطل مناسب و حذف عنصر درخواستی (در صورت وجود) و جابجائی مقادیر عناصر بعدی و به‌روزرسانی بیت‌های متاداده متناظر

در صورت حذف آخرین باقیماندهٔ سطل متناظر پاکسازی بیت bucket_occupied

فیلتر خارج

الگوریتم ۱۳- حذف مقدار از فیلتر خارج‌قسمت		
۱	ورودی: عنصر $x \in D$	
۲	ورودی: فیلتر خارج‌قسمت با تابع درهم‌ساز h	
۳	خروجی: False اگر عنصر یافت نشد و True در غیر این صورت	
۴	$f \leftarrow h(x)$	
۵	$f_q, f_r \leftarrow f$	
۶	اگر $QF[f_q].bucket\_occupied \neq ۱$ آنگاه برگرداندن True	
۷	$r_{\text{آغاز}}, r_{\text{پایان}} \leftarrow Scan(QF, f_q)$	
۸	برای i از $r_{\text{آغاز}}$ تا $r_{\text{پایان}}$ انجام بده	
۹	اگر $QF[i] == f_r$ آنگاه	یافتن مقدار و حذف آن
۱۰	$QF[i] \leftarrow \phi$	
۱۱	$QF[i].run\_continued \leftarrow ۰$	
۱۲	$QF[i].isShifted \leftarrow ۰$	
۱۳	اگر $i < r_{\text{پایان}}$ آنگاه	انتقال به چپ
۱۴	$QF \leftarrow ShiftLeft(QF, i)$	
۱۵	اگر $r_{\text{آغاز}} == r_{\text{پایان}}$ آنگاه	تهی شدن اجرا
۱۶	$QF[i].bucket\_occupied \leftarrow ۰$	صفر شدن مقدار اشغال
۱۷	برگرداندن True	
۱۸	برگرداندن False	نبودن عضو در اجرا

فیلتر خارج قسمت-ویژگی‌ها

مثبت کاذب امکان‌پذیر

- داده‌ساختار فیلتر خارج‌قسمت نمایش فشرده‌ای از مجموعه چندگانه از اثرانگشت‌ها
- نسبت مثبت کاذب آن تابعی از تابع درهم‌ساز h و تعداد مقادیر عناصر n اضافه شده به فیلتر
- مضافا، تصادم سخت
  - دو مقدار متفاوت دارای مقادیر یکسان باقیمانده و خارج‌قسمت
  - به دلیل چنین رویدادهای بسیار نادری، امکان وقوع پاسخ‌های مثبت کاذب و محدود بودن احتمال آنها pr_{fp} با کران بالای زیر

$$pr_{fp} \approx 1 - e^{-\frac{n}{2^p}} \leq \frac{n}{2^p}$$

پس، با تعداد ثابت عناصر مورد انتظار n، امکان سبک-سنگینی بین احتمال مثبت کاذب pr_{fp} و طول اثر انگشت p

استفاده پیاده‌سازی‌های عملی فیلترهای خارج‌قسمت از اثر انگشت‌های ۳۲ و ۶۴ بیتی

فیلتر خارج قسمت

اهمیت ضریب بار در فیلتر خارج‌قسمت همانند سایر جدول‌های درهم

در پی تخصیص حداقل اندازه به تعداد مقادیر مورد انتظار سطل

- به بیان دیگر، انتخاب تعداد سطل‌ها m به صورت زیر

$$m = 2^q > n$$

محاسبهٔ طول باقیمانده r

$$r = \left\lceil \log\left(-\frac{n}{2^q} \times \frac{1}{\ln(1-pr_{fp})} \right) \right\rceil$$

فیلتر خارج قسمت

منفی کاذب امکان‌ناپذیر

- در صورت پاسخ منفی الگوریتم عضویت فیلتر خارج‌قسمت، قطعاً عدم عضویت در مجموعه

$$pr_{fn} = 0$$

سرعت

- فیلتر خارج‌قسمت حدود بیست درصد بزرگتر از فیلتر بلوم
  - در عین حال سریع‌تر
  - چرا؟
- سرعت بالاتر به دلیل نیاز به ارزیابی یک تابع درهم‌ساز در هر دسترسی
- ذخیرهٔ داده‌ها در پیمانه‌های پیوسته
- آزمون‌ها در فیلتر خارج‌قسمت منجر به یک missed cache
  - در مقابل حداقل دو missed cache در انتظار برای الگوریتم فیلتر بلوم

فیلتر خارج قسمت

میزان مثبت کاذب و ملاحظات فضایی

- وقوع مثبت کاذب در فیلتر خارج‌قسمت با تولید اثرانگشت یکسان برای دو کلید متفاوت
- جدول شامل 2^q خانه و طول اثرانگشت برابر $p = q + r$
- آن‌گاه احتمال وقوع مثبت کاذب تقریباً برابر با 2^{-r}
  - وابستگی این احتمال به تعداد بیت‌های باقی‌مانده
  - فضای موردنیاز برای جدول فیلتر خارج‌قسمت برابر با $2^q(r + 3)$ بیت
  - تعداد عناصری که در فیلتر درج می‌شوند برابر با $n = \alpha\, 2^q$
    - α ضریب بارگذاری
    - تاثیر چشمگیر ضریب تأثیر بر کارایی عملیات درج، جست‌وجو، و حذف

معرفی گونهٔ کم‌حجم‌تر از فیلتر خارج‌قسمت

- استفاده از تنها از دو بیت متاداده
- نیاز به فضای $2^q(r + 2)$ بیت
- با وجود کاهش یک بیت، عدم افزایش میزان مثبتِ کاذب
- پیچیدگی سخت بیشتر مرحلهٔ رمزگشایی
- به‌ویژه در مواقع وجود خوشه‌های بلند،
  - عملیات متداول (درج، جست‌وجو و حذف) وابستگی بیشتر به پردازنده و پرهزینه‌تر

فیلتر خارج قسمت

مقایسهٔ با فیلترهای بلوم

- به‌سببِ فضای اضافی لازم برای جدولِ جست‌وجوی خطی
- برای میزان رایجِ مثبتِ کاذب فیلترهای خارج‌قسمت
  - معمولاً مصرف فضا اندکی بیشتر از فیلترهای بلوم فضا
  - کارایی فضایی بیشتر فیلترهای خارج‌قسمت نسبت به فیلترهای بلوم برای میزان بسیار پایینِ مثبتِ کاذب
- درج
  - فیلترهای خارج‌قسمت می‌توانند ۲٫۴ میلیون درج در ثانیه در مقابل فیلترهای بلوم محدود به حدود ۰٫۶۹ میلیون
- آزمون عضویت
  - هر دو در همان سطح حدود ۲ میلیون در ثانیه

فیلتر خارج قسمت

مثال- حافظه مورد نیاز-

- برای مدیریت یک میلیارد مقدار، فیلتر خارج‌قسمت باید حداقل $2^{۳۰}$ سطل
- عدم امکان استفاده از اثر انگشت‌های کوتاه‌تر از ۳۱ بیت
- در پی احتمال رویدادهای مثبت کاذب حدود دو درصد
  - تعداد بیت‌ها برای باقیمانده

$$r = \left\lceil \log\left(-\frac{10^9}{2^{30}} \times \frac{1}{\ln(1-0.02)} \right) \right\rceil = 6$$

- پس، طول مورد نیاز اثر انگشت $p = q + r = 36$ بیت
- استفاده از ۳۰ بیت اول برای سطل‌بندی و ذخیره ۶ بیت باقی‌مانده در سطل مناسب
- هر سطل علاوه بر این شامل سه بیت متاداده
  - اندازه کل فیلتر خارج‌قسمت $2^{30} \times 9$ بیت معادل تقریباً ۱٫۲ گیگابایت حافظه

فیلتر خارج قسمت

فیلتر خارج‌قسمت

امکان بازیابی اثر انگشت‌ها از داده‌های ذخیره شده

- در نتیجه پشتیبانی از حذف، ادغام، و تغییر اندازه

عدم تاثیر ادغام بر نسبت مثبت کاذب فیلترها

همچنین،

- پشتیبانی فیلتر بلوم شمارنده از حذف‌های احتمالا درست اما درستی همیشگی حذف در فیلتر خارج‌قسمت

فیلتر خارج قسمت

تغییر اندازه فیلتر خارج‌قسمت

- هم کوچک‌سازی و هم بزرگ‌سازی
- با حرکت روی فیلتر
  - کپی کردن هر اثر انگشت در داده‌ساختاری تازه تخصیص یافته
- بدون نیاز به درهم‌سازی مجدد

متاداده			باقیمانده	خق
				۰۰۰۰
				۰۰۰۱
				۰۰۱۰
				۰۰۱۱
۰	۰	۱	۱۰۱۰	۰۱۰۰
				۰۱۰۱
				۰۱۱۰
				۰۱۱۱
				۱۰۰۰
				۱۰۰۱
				۱۰۱۰
				۱۰۱۱
				۱۱۰۰
				۱۱۰۱
۰	۱	۱	۰۱۱۱	۱۱۱۰
				۱۱۱۱

متاداده			باقیمانده	خق
				۰۰۰
				۰۰۱
۰	۰	۱	۰۱۰۱۰	۰۱۰
				۰۱۱
				۱۰۰
				۱۰۱
				۱۱۰
۰	۰	۱	۰۰۱۱۱	۱۱۱

متاداده			باقیمانده	خق
				۰۰
۰	۰	۱	۰۰۱۰۱۰	۰۱
				۱۰
۰	۰	۱	۱۰۰۱۱۱	۱۱

فیلتر خارج قسمت

ادغام دو یا چند فیلتر خارج‌قسمت

- با استفاده از الگوریتمی شبیه به مرتب‌سازی ادغامی، الگوریتم مرتب‌سازی تقسیم و غلبه ابداعیِ جان فون نویمان

امکان اسکن موازی همه فیلترهای ورودی و نوشتن نتیجه ادغام شده در فیلتر خروجی

زمان مورد نیاز برای انجام آزمون، افزودن یا حذف در فیلتر خارج‌قسمت

- وابسته به زمان اسکن به عقب و جلو

طراحی فیلتر خارج‌قسمت با تمرکز بر کلان‌داده (به عنوان مثال، یک میلیارد مقدار برای تابع درهم‌ساز ۶۴ بیتی)

کاهش مزایا در مجموعه‌های داده کوچک یا متوسط، به دلیل پیچیدگی آن

مرتب‌سازی ادغامی

اگر $n = ۱$، آن‌گاه آرایه مرتب

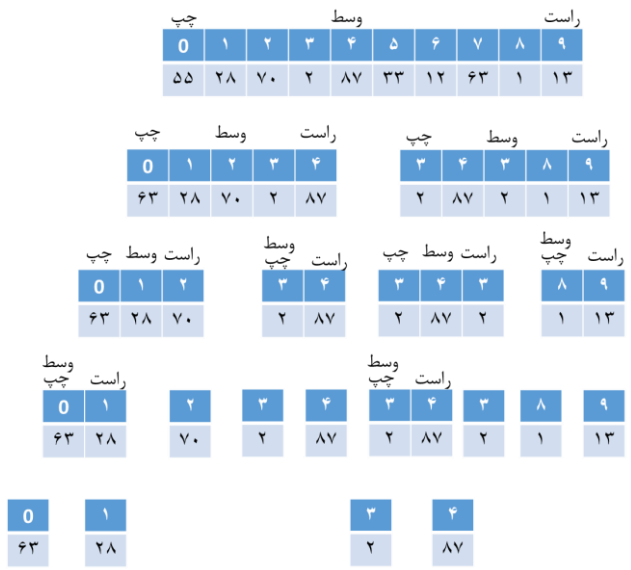
در صورت مرتب بودن جداگانهٔ دو زیرفهرست با اندیس‌های $[0: n/۲]$ و $[n/۲ + ۱: n - ۱]$

- ادغام روشمند موجب مرتب‌سازی ترکیب دو زیرفهرست
- دو عمل تقسیم و ترکیب
- عدم جابجایی در هنگام تقسیم

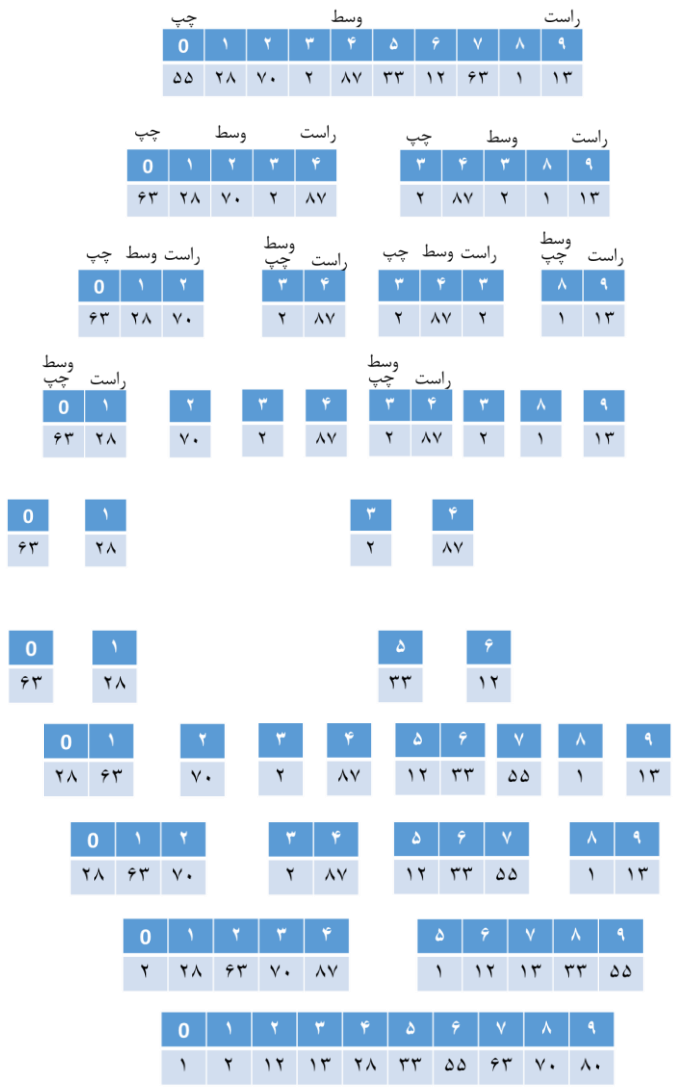
مرتب‌سازی ادغامی

چپ			وسط					راست
0	۱	۲	۳	۴	۵	۶	۷	۸	۹
۵۵	۲۸	۷۰	۲	۸۷	۳۳	۱۲	۶۳	۱	۱۳

مرتب‌سازی ادغامی



مرتب‌سازی ادغامی



مرتب‌سازی ادغامی

شبه‌کد بخش ادغام آن به صورت زیر است.

```
Alg. Edqam(arr, c, v, r)
    arr_c = arr[c:v ];    arr_r[v+١:r]
    i = j = ٠
    k = c
    while i <= v & j <= r – v
        if arr_c[i] <= arr_r[j]
            arr[k] = arr_c[i++]
        else
            arr[k] = arr_r[j++]
        k++
    while i <=v
        arr[k] = arr_c[i++]
        i++; k++
    while j <=r - v
        arr[k] = arr_r[j++]
        j++; k++
```

الگوریتم تقسیم و غلبه بر اساس تابع ادغام به صورت زیر تعریف می‌شود.

```
Alg. Morattabsazi_edqami(arr, c, r)
    If c < r
        v = c + (r – c) / ٢
    Morattabsazi_edqami(arr, c, v)
    Morattabsazi_edqami(arr, v + ١ r)
    Edqam(arr, c, v, r)
```

فیلتر فاخته

اکثر بهبودهای فیلتر بلوم کلاسیک با پشتیبانی از حذف

دچار افت از نظر فضا یا عملکرد

بین فن، دیوید اندرسون، مایکل کامینسکی و مایکل میتزن ماخر در سال ۱۳۹۳ پیشنهاد فیلتر فاخته

نوعی فشرده از جدول درهم فاخته

به جای ذخیرهٔ جفت‌های کلید-مقدار، تطبیق اثر انگشت‌هایی با طول p برای هر مقدار درج شده

پیاده‌سازی آسان‌تر، پشتیبان درج و حذف پویا

استفاده از فضای کمتری نسبت به سایر اصلاحات فیلتر بلوم در بسیاری از کاربردهای عملی

نمایش عملکردی بالاتر

فیلتر فاخته

داده‌ساختار با جدول درهم فاخته چندگانه با m سطل

ذخیره b مقدار در هر یک

درج مقدار جدید در درهم فاخته استاندارد

- نیاز به دسترسی به مقادیر موجود اصلی
  - جهت تعیین محل جدید برای ذخیرهٔ مقادیر جدید در صورت نیاز به فضا
  - ذخیرهٔ اثر انگشت‌ها در فیلتر فاخته
  - عدم وجود راهی برای بازیابی عناصر اصلی و درهم‌سازی مجدد آنها برای یافتن سطل جدیدشان در جدول درهم

فیلتر فاخته

هدف غلبه بر این محدودیت و همچنان استفاده از درهم‌ساز فاخته

- استفاده الگوریتم فیلتر فاخته از «درهم‌ساز فاخته با کلید جزئی»
- استخراج مقدار موجود از سطل بدون دانستن خود مقدار اصلی و صرفا از اثر انگشت آن
- درج هر مقدار **x**، محاسبهٔ اثر انگشت **p**-بیتی **f** و نمایه‌های دو سطل نامزد:

$$i = h(x)\%m$$
$$j = (i \oplus (h(f)\%m)\%m$$

جهت داشتن توزیعی یکنواخت از مقادیر

فیلتر فاخته

کوچکتر بودن طول اثر انگشت p در مقایسه با طول فیلتر m

- تغییر تعداد کمی از بیت‌های پایین‌تر با عملیات یای انحصاری
  - ثابت ماندن بیشتر بیت‌های مرتبه بالاتر
- نمایشگر تمایل مقادیر انتقال یافته از سطل‌های اصلی خود به قرارگیری در نزدیکی یکدیگر در سطل‌های جانشین
- در نتیجه توزیع در جدول درهم دچار انحراف
  - تاثیرگذار بر کارایی فیلتر
  - درهم‌ساز اثر انگشت‌ها ضامن انتقال این مقادیر به سطل‌هایی در بخش‌های کاملاً متفاوت جدول درهم
    - بنابراین کاهش تصادم‌های درهم و بهبود استفاده از جدول

فیلتر فاخته

عملیات یا انحصاری $\oplus$

- با دانستن سطل‌های فعلی مقدار k امکان محاسبهٔ سطل جانشین آن k^* بدون بازیابی مقدار اصلی

$$k^* = \big(k \oplus h(f)\big)\%m$$

فیلتر فاخته

درج مقدار جدید x

- محاسبهٔ نمایه‌های دو سطل نامزد با معادلات
- در صورت خالی بودن یکی از دو سطل حاصل، درج مقدار در آن
- در صورت پر بودن هر دو
  - انتخاب تصادفی یکی از آن سطل‌ها و ذخیرهٔ مقدار x در آن‌جا
  - انتقال مقدار از آن سطل به سطل نامزد جانشین
  - تکرار روش تا زمان یافتن سطل خالی یا تا زمان رسیدن به آستانهٔ به حداکثر تعداد جابجایی‌ها
    - در صورت خالی نبودن سطل خالی، پر در نظر گرفتن فیلتر

فیلتر فاخته

الگوریتم ۱۴- درج مقدار در فیلتر فاخته

ورودی: مقدار $x \in D$

ورودی: فیلتر فاخته با اثر انگشت‌گذاری و تابع درهم‌ساز h

خروجی: True اگر مقدار اضافه شده باشد و False در غیر این صورت

$f \leftarrow$ انگشت اثر (x)

$i \leftarrow h(x)$

$j \leftarrow i \oplus h(f)$

اگر CUCKOOFILTER [i] فضای خالی دارد آنگاه

CUCKOOFILTER [i].add (f)

برگرداندن True

در غیر این صورت اگر CUCKOOFILTER [j] فضای خالی دارد آنگاه

CUCKOOFILTER [j] .add (f)

برگرداندن True

$k \leftarrow$ نمونه $(\{i, j\})$

برای n از صفر تا بیش-تکرار انجام بده

$x \leftarrow$ نمونه$(CUCKOOFILTER\ [k])$

f و اثر انگشت ذخیره شده در ورودی x را با هم عوض کن

$k = k \oplus h(f)$

اگر CUCKOOFILTER [k] فضای خالی دارد آنگاه

$CUCKOOFILTER\ [k]\ .add\ (f)$

برگرداندن True

برگرداندن False

فیلتر فاخته

مثال- افزودن مقدار به فیلتر

داده‌ساختار به طول $m = 10$

جهت سادگی، ذخیرهٔ صرفا یک اثر انگشت $p = 16$ بیتی در هر سطل

- استفاده از تابع درهم‌ساز مورمور۳ با اندازه ۳۲ بیتی برای محاسبهٔ اثر انگشت‌ها و نمایه‌های سطل
- درج نام پایتخت‌ها درج

فیلتر فاخته

کپنهاگ

$$f = MurmurHash3(Copenhagen)\%\, 2^p = 49615$$

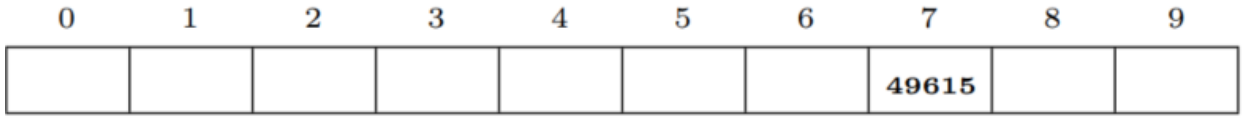
سطل اصلی i

$$i = MurmurHash3(Copenhagen)\%\, m = 7$$

استخراج سطل جایگزین j از i و اثر انگشت f

$$j = \left(i \oplus MurmurHash3(f)\right)\%\, m = (1 \oplus 34475545)\%\, 10 = 0$$

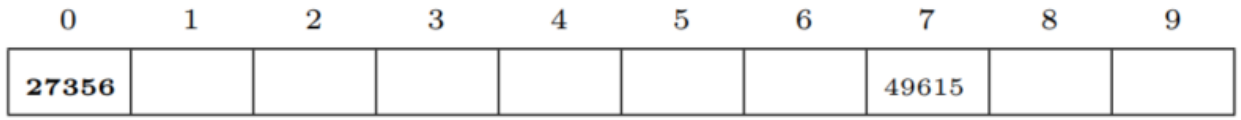
- امکان نمایهٔ اثر انگشت f در سطل ۷ یا ۰
  - استفاده از سطل اصلی به دلیل خالی بودن آن



فیلتر فاخته

آتن

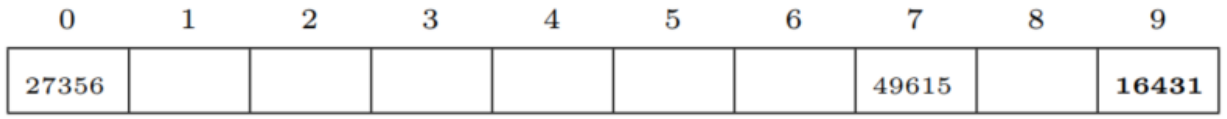
- اثر انگشت f = 27356 و سطل‌های نامزد ۰ و ۷
  - سطل اصلی ۰ آزاد



فیلتر فاخته

لیسبون

- اثر انگشت آن 16431 $= f$ و سطل‌های نامزد ۷ و ۹
- شروع با سطل اصلی ۷
  - اما اشغال شده
  - در حداکثر ظرفیت یک
  - بررسی سطل جانشین ۹
    - خالی
  - ذخیره اثر انگشت در آنجا

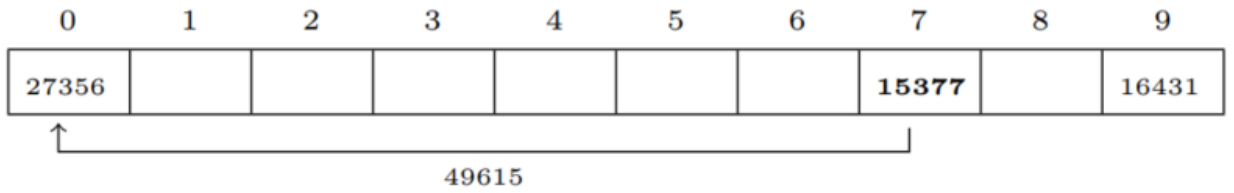


فیلتر فاخته

هلسینکی

- اثر انگشت $f = 15377$ و هر دو نمایه سطل برابر با ۷
  - چنین برخورد نمایه‌ای برای فیلترهای کوچک، محتمل‌تر از فیلترهای واقعی
  - اشغال شدگی سطل ۷ اشغال و عدم امکان پذیرش بیش از یک مقدار
- نیاز به آغاز روش جابجایی در فیلتر
- سطل k
- تبادل مقدار ۴۹۶۱۵ از سطل ۷ با مقدار f
- انتقال آن مقدار به سطل جدید k

$$k = \left(7 \oplus MurmurHash3(49615)\right)\%\ 10 = 0$$

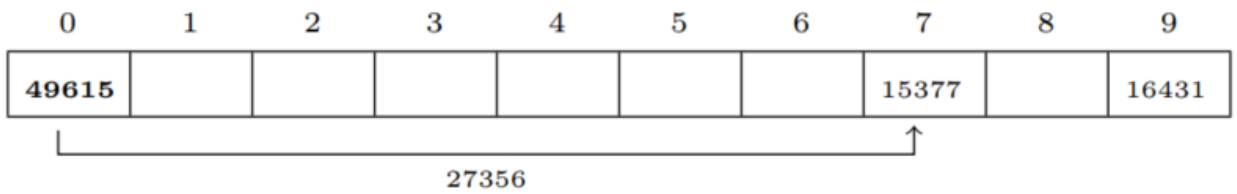


فیلتر فاخته

سطل ۰ قبلاً حاوی مقدار ۲۷۳۵۶

- تبادل آن با ۴۹۶۱۵
- محاسبه نمایه سطل جدید

$$k = \left(0 \oplus MurmurHash3(27356)\right)\% 10 = 7$$



فیلتر فاخته

برگشت به سطل ۷

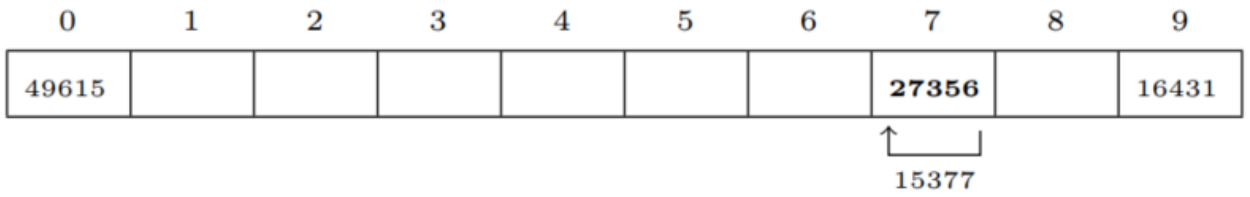
- پر بودن آن

نیاز به تکرار روش جابجایی

تبادل مقدار ۲۷۳۵۶ و ذخیره آن در سطل فعلی

محاسبه سطل جدید برای مقدار ۱۵۳۷۷

$$k = (7 \oplus \text{MurmurHash3}(15377)) \% 10 = 7$$



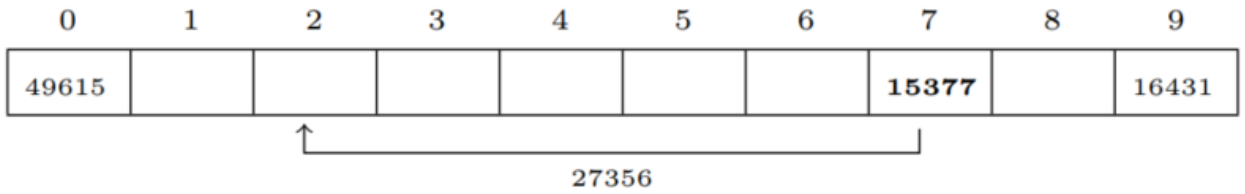
فیلتر فاخته

برگشت به نمایه به دلیل تصادم

ذخیرهٔ مقدار ۱۵۳۷۷ در آن

انتقال مقدار ۲۷۳۵۶ به یک سطل جدید k

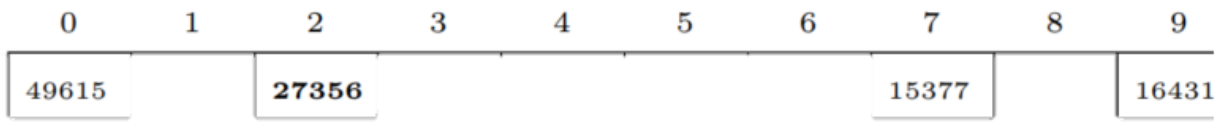
$$k = \left(7 \oplus MurmurHash3(27356)\right)\%\ 10 = 2$$



فیلتر فاخته

سطل ۲ خالی

- ذخیرّ مقدار ۲۷۳۵۶ در آن
  - ختم روش درج



فیلتر فاخته

آزمون وجود مقدار در فیلتر

- ساده
- ابتدا، برای عنصر مورد آزمون، محاسبهٔ اثر انگشت و سطل‌های نامزد آن
- اگر اثر انگشت در هر یک از سطل‌ها وجود داشته باشد، نتیجه می‌گیریم که عنصر ممکن است وجود داشته باشد.
- در غیر این صورت، قطعاً در فیلتر نیست.

فیلتر فاخته

الگوریتم ۱۵- آزمون مقدار در فیلتر فاخته

ورودی: مقدار $x \in D$

ورودی: فیلتر فاخته با اثر انگشت‌گذاری و تابع درهم‌ساز h

خروجی: False اگر مقدار یافت نشد و True اگر ممکن است وجود داشته باشد

$f \leftarrow$ انگشت اثر (x)

$i \leftarrow h(x)$

$j \leftarrow i \oplus h(f)$

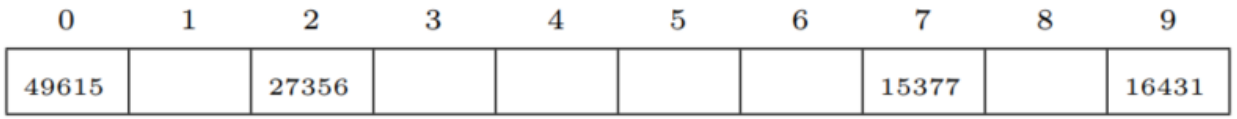
اگر $f \in CuckooFilter[i]$ یا $f \in CuckooFilter[j]$ آنگاه

برگرداندن True

برگرداندن False

فیلتر فاخته

مثال- آزمون عناصر در فیلتر



فیلتر فاخته

لیسبون

با سطل‌های نامزد آن ۷ و ۹ و اثر انگشت آن f = 16431

همانطور که در بالا محاسبه کردیم، آزمایش کنیم.

می‌توانیم مقدار ۱۶۴۳۱ را در سطل ۹ پیدا کنیم و نتیجه بگیریم که عنصر لیسبون ممکن است در فیلتر وجود داشته باشد.

در مقابل، عنصر اسلو را در نظر می‌گیریم که دارای اثر انگشت f = 53104 و سطل‌های نامزد ۰ و ۶ است.

همانطور که می‌بینیم، چنین مقداری در این سطل‌ها وجود ندارد، بنابراین عنصر اسلو قطعاً در فیلتر نیست.

فیلتر فاخته

حذف عنصر،

ایجاد اثر انگشت

محاسبهٔ نمایه‌های سطل‌های نامزد

بررسی اثر انگشت در نمایه‌ها

در صورت مطابقت با هر یک از مقادیر موجود در هر یک از سطل‌ها

- حذف یک نسخه از اثر انگشت از آن سطل

فیلتر فاخته

الگوریتم ۱۶- حذف مقدار از فیلتر فاخته

ورودی: مقدار $x \in D$

ورودی: فیلتر فاخته با اثر انگشت‌گذاری و تابع درهم‌ساز h

خروجی: True اگر مقدار حذف شده باشد و False در غیر این صورت

$f \leftarrow$ انگشت اثر (x)

$i \leftarrow h(x)$

$j \leftarrow i \oplus h(f)$

اگر $f \in CUCKOOFILTER[i]$ آنگاه

CUCKOOFILTER[i].drop(f)

برگرداندن True

در غیر این صورت اگر $f \in CUCKOOFILTER[j]$ آنگاه

CUCKOOFILTER[j].drop(f)

برگرداندن True

برگرداندن False

فیلتر فاخته-ویژگی‌ها

دو راه حل پشتیبانی از حذف فیلتر فاخته نیاز

- ذخیره چندین نسخه از مقدار
- ایجاد شمارنده برای هر مقدار ذخیره شده

هر دو رویکرد پشتیبانی «حذف‌های احتمالا درست»

- دلایل:
- به دلیل ظرفیت محدود سطل
  - عدم امکان ذخیره بیش از $2b$ مقدار یکسان
- به دلیل سرریز شمارنده‌ها، مانند فیلتر بلوم شمارنده

نداشتن مشکل در فیلتر فاخته غیرقابل حذف

- فضاکارآمد
- عدم نیاز به به خاطر سپردن مقادیر یکسانی که چندین بار اضافه شده‌اند

فیلتر فاخته

مثبت کاذب امکان‌پذیر

مقادیر مختلف دارای اثر انگشت یکسان

- در بیشتر موارد سطل‌های نامزد متفاوتی
- در نتیجه امکان تمایز آنها

با این حال، سطل‌های نامزد یکسان برای آن مقادیر

- رخ دادن تصادم سخت

امکان پاسخ مثبت کاذب به دلیل چنین رویدادهای بسیار نادری

- احتمال مثبت کاذب

$$pr_{fp} = 1 - \left(1 - \frac{1}{2^p}\right)^{2b} \approx \frac{2b}{2^p}$$

فیلتر فاخته

با تعداد ثابت عناصر مورد انتظار n،

- سبک سنگینی بین احتمال مثبت کاذب pr_{fp} و اندازه سطلb
  - جبران با طول اثر انگشت‌ها p
  - اثر انگشت‌ها به اندازه کافی طولانی باشند، درهم‌ساز فاخته با کلید جزئی تقریب خوبی از درهم‌ساز فاخته استاندارد
  - تاثیر اثر انگشت‌های طولانی‌تر بر فضای مورد نیاز
- برآورد طول اثر انگشت توصیه شده p

$$p \geq \left\lceil \log \frac{2b}{pr_{fp}} \right\rceil$$

فیلتر فاخته

کران پائین ذخیره مقادیر به اندازه تعداد مقادیر ورودی در m سطل با اندازه b:

$$m \geq \left\lceil \frac{n}{b} \right\rceil$$

فیلتر فاخته

منفی کاذب امکان‌ناپذیر

مانند سایر موارد،

- در صورت برگشت عدم عضویت
- قطعاً عدم عضویت

$$pr_{fn} = 0$$

تضمین اشغال فضای بالا

- اصلاح تصمیم‌گیری‌های قبلی قرارگیری مقادیر با درج مقدار جدید
- با این حال، دارای حداکثر ظرفیتی بیان شده با ضریب بار α
- پس از رسیدن به حداکثر ضریب بار قابل اجرا،
  - شکست فزایندهٔ درج‌ها
  - نیاز به بزرگ کردن جدول درهم جهت ذخیره مقادیر بیشتر

فیلتر فاخته

میانگین تعداد بیت‌ها به ازای هر عنصر β
- به عنوان نسبت بین طول اثر انگشت‌ها و ضریب بار α
- تخمین آن با احتمال مثبت کاذب ثابت pr_{fp}

$$\beta \leq \frac{1}{\alpha} \times \left\lceil \log \frac{2b}{pr_{fp}} \right\rceil$$

استفاده درهم‌ساز فاخته از دو تابع درهم‌ساز
- ضریب بار با سطل‌هایی با اندازه $b = 1$ برابر با پنجاه درصد
- افزایش اندازه سطل امکان بهبود اشغال جدول
- مثال، برای $b = 2$ و $b = 4$ ضریب بار به ترتیب ۸۴ درصد و ۹۵ درصد است.

مطالعه تجربی
- کفایت سطل‌هایی با اندازه $b \in \{1,2,3,4\}$

فیلتر فاخته

مثال- حافظه مورد نیاز

- مدیریت یک میلیارد مقدار با فیلتر فاخته
- احتمال رویدادهای مثبت کاذب در حدود دو درصد
- با اشغال ۸۴ درصدی جدول را ۸۴

نیازمندی‌های پشتیبانی از چنین ضریب باری

- انتخاب اندازه سطل‌ها $b = 2$
- پس طول فیلتر $m = 2^{۲۹}$
- حداقل طول اثر انگشت

$$p = \left\lceil \log \frac{4}{0.02} \right\rceil = 8$$

فیلتر فاخته

بنابراین، طول مورد نیاز اثر انگشت ۸ بیت است

اندازه کل فیلتر فاخته $2 \times 8 \times 2^{29}$ بیت

- تقریباً معادل ۱,۰۷ گیگابایت حافظه

برای مقایسه نیازهای فضایی با سایر فیلترهای مطالعه شده،

- امکان استفاده از $b = 1$
  - دستیابی به ۵۰ درصد اشغال جدول
- نیاز به فیلتری به طول $m = 2^{۳۰}$
- استفاده از اثر انگشت‌های ۹ بیتی استفاده کنیم

منجر به حدود ۰,۹۴ گیگابایت حافظه

فیلتر فاخته

استفاده فیلترهای فاخته از رویکردی مشابه فیلتر بلوم شمارنده

- با کارایی فضایی بهتری و پیاده‌سازی بسیار ساده‌تر

مناسب برنامه‌های با مقادیر ذخیره شده زیاد و میزان مثبت کاذب نسبتاً پایینی (کمتر از ۳ درصد)

فضای کمتر نسبت به فیلترهای بلوم

زمانی که فیلتر فاخته در حداکثر ظرفیت خود

- نیاتز به بزرگ کردن جدول درهم
- عدم امکان درج عضو جدید تا آن زمان امکان
- در مقابل، درج در فیلتر بلوم به قیمت افزایش نسبت مثبت کاذب