الگوریتم
تعداد متمایز

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

تخمین تعداد منحصر به فرد

مسئله تخمین تعداد منحصربه‌فرد

- در پی یافتن تعداد مقادیر متمایز در مجموعه داده‌ای با وجود اعضای تکراری
- کاربرد در حوزه‌های گوناگونی از تولید نرم‌افزار اعم پایگاه داده‌ها و ترافیک شبکه

روش سنتی و دم دستی تعیین تعداد دقیق مجموعه

- ایجاد فهرستی از همه مقادیر
- اجتناب از شمارش عضو تکراری با مرتب‌سازی و جستجو
  - احصاء تعداد مقادیر متمایز
- شمارش دقیق مقادیر منحصربه‌فرد با پیچیدگی زمانی $O(N \log N)$
  - N تعداد کل مقادیر از جمله تکراری‌ها،
- نیاز به حافظه خطی کمکی

تخمین تعداد منحصر به فرد

گسترش خدمات اینترنتی،

- با میلیاردها کلیک، جستجو، و سفارش
- روزانه با تعداد بسیار کمتری کاربر
- اهمیت بازطراحی شمارش تعداد متمایز

تعیین تعداد منحصر به فرد

- با پیمایش یکبارهٔ داده و در فضای اشغالی کمتر
- از حوزه‌های مهم تحقیقاتی

تخمین تعداد منحصر به فرد

مثال‌های تخمین کاردینالیتی

- تعیین تعداد بازدیدکنندهٔ یکتا از محصولی خاص
- تعداد کاربران متمایز استفاده کننده از قابلیتی مشخص در برنامه‌های وبی
- تشخیص تغییرات ناگهانی در تعداد آدرس‌های IP مبدأ–مقصدِ یکتا عبوری از روتر
  - نشان‌دهندهٔ امکان حملهٔ انکارِ خدمت

تخمین تعداد منحصر به فرد

نحوهٔ تکثیر مکرر اطلاعات در وب،

- اندازه‌گیری کاردینالیتی کمک کننده در تشخیص مواجهه با تعداد محتوای متمایز
  - مثال تعداد مقاله‌های خبری یکتا یا تعداد نسخه‌های متفاوت از محتوای وب‌سایت

داده‌مجموعه‌های عظیم امروز

- علاقهٔ فزاینده‌ای به طراحی الگوریتم‌هایی جهت تقریب کاردینالیتی مجموعه
- با دقتی مناسب و در فضایی بسیار کوچک‌تر از اندازهٔ خود مجموعه

تخمین تعداد منحصر به فرد-مثال

از شاخص‌های کلیدی ارزشمند برای هر وب‌سایت

- تعداد بازدیدکنندگان منحصربه‌فردی بازدیدکننده در بازهٔ زمانی مشخص
- جهت سادگی فرض استفاده هر بازدیدکننده از یک نشانی IP در تمامی بازدیدها
- با شمارش تعداد IP-های منحصربه‌فردکه با پروتکل اینترنت نسخه IPv6 با رشته‌های ۱۲۸ بیتی
- بدست آوردن تعداد بازدیدکنندگان
  - آسان؟
  - آیا می‌توان با تکیهٔ صرف بر روش‌های کلاسیک برای شمارش دقیق تعداد را بدست آورد؟
  - وابستگی پاسخ به میزان استفاده از تارمانه

تخمین تعداد منحصر به فرد-مثال

آمار ترافیک اسفند ۱۳۹۵ سه وب‌سایت خرده فروشی پر استفاده در ایالات متحده

- amazon.com، ebay.com و walmart.com
- مطابق SimilarWeb
  - میانگین تعداد بازدید از این وب‌سایت‌ها حدود ۱,۴۴ میلیارد
  - میانگین تعداد صفحات مشاهده شده در هر بازدید ۸,۲۴ بود. سخن کوتاه،
- آمار اسفند ۱۳۹۵ مراجعه به حدود ۱۲ میلیارد آدرس IP۱۲۸ بیتی
- نگه‌داری همه آنها اندازه‌ای در حدود ۱۹۲ گیگابایت
- فرض به ازای هر ۱۰ بازدیدکننده، یک نفر منحصربه‌فرد
- انتظار مجموعه‌ای از تعداد منحصربه‌فرد حدود ۱۴۴ میلیون
- حافظه مورد نیاز برای ذخیره فهرست مقادیر منحصربه‌فرد ۲۳ گیگابایت

تخمین تعداد منحصر به فرد-مثال

مطالعه همبستگی‌ها در توالی‌های DNA
- از پژوهش‌های مهم در تحقیقات ژنوم انسانی

مولکول‌های DNA شامل دو رشتهٔ جفت شده
- هر کدام از چهار واحد شیمیایی پایه DNA، با علامت A (آدنین)، G (گوانین)، C (سیتوزین) و T (تیمین)
- ژنوم انسان حدود ۳ میلیارد جفت پایه

توالی‌یابی به معنای تعیین ترتیب دقیق جفت‌های پایه در قطعاتی از DNA

توالی DNA رشته‌ای از نمادهای A، G، C، T
- به اندازه دلخواه طولانی
- نمونه‌ای از مجموعه داده‌ای بالقوه بی‌نهایت

مسئله اندازه‌گیری همبستگی
- در پی تعیین تعداد زیررشته‌های متمایز با اندازه ثابت در قطعه‌ای DNA

با فرض
- هر توالی با زیررشته‌های متمایز کم، همبستگی بیشتری نسبت به توالی دیگری با اندازه مشابه اما با زیررشته‌های متمایز بیشتر

چنین آزمایش‌هایی نیاز به اجرای چندباره روی فایل‌های بزرگ متعدد

نیاز به حافظه محدود یا حتی ثابت و زمان اجرای کم برای تسریع تحقیقات
- غیرقابل اجرا با الگوریتم‌های شمارش دقیق

تخمین تعداد منحصر به فرد

دستاوردهای احتمالی تخمین دقیق تعداد منحصربه‌فرد نیاز زمان پردازش و حافظه زیاد

گرایش برنامه‌های کلان‌داده به سمت رویکردهای عملی‌تر

- غالبا بر مبنای الگوریتم‌های احتمالاتی و عرضهٔ پاسخی تقریبی

نیاز به داشتن درکی از اندازه مجموعه داده و تعداد مقادیر متمایز احتمالی در تصمیم‌گیری

- دنبالهٔ بی‌نهایت (بالقوه بی‌نهایت) رشته‌های تک حرفی a، d، s، ... بر اساس حروف الفبای لاتین
- راحتی امکان تخمین تعداد منحصربه‌فرد
- کران بالای آن با تعداد حروف، که در الفبای لاتین فعلی در انگلیسی۲۶
- عدم نیاز به استفاده از هیچ رویکرد احتمالاتی
- راه‌حل مبتنی بر لغت‌نامهٔ ساده برای محاسبه دقیق تعداد منحصربه‌فرد

تخمین تعداد منحصر به فرد

برای پرداختن به مسئله تعداد منحصربه‌فرد،

تاثیر بسیاری از روش‌های احتمالاتی مطرح از الگوریتم فیلتر بلوم

عمل بر اساس مقادیر درهم

مشاهدهٔ الگوهای رایج در توزیع آنها

«حدس‌های» منطقی در مورد تعداد مقادیر منحصربه‌فرد بدون نیاز به ذخیره همه آنها

شمارش اقلام متمایز در پایگاه‌های داده

برای سنجش کاردینالیته (اندازهٔ مجموعه) از پایگاه‌های داده و نحوهٔ استفادهٔ SQL از کلیدواژهٔ DISTINCT

اعمال کلیدواژه بر روی یک ستون در جدول

- دستور SELECT DISTINCT بازگرداندن تمام اقلام متمایز آن ستون
- دستور SELECT COUNT DISTINCT برگرداندن تعداد اقلام متمایز موجود در ستون مورد نظر
- رواج پرس‌وجوهای حاوی COUNT DISTINCT
  - به‌ویژه در حوزهٔ تجارت الکترونیک
  - آمار استفاده از وب‌سایت
- ذخیرهٔ داده‌های بازدید کاربران اغلب در جدولی به نام DAILY_VISITS
  - معمولا بسیار بزرگ
- دارای ویژگی‌هایی نظیر session_id، timestamp، product_id، user_ip_address، visit_duration و غیره
- اجرای عملیات SELECT زیر:

```
SELECT COUNT (DISTINCT user_ip_address) WHERE product_id = ########
FROM DAILY_VISITS
```

شمارش اقلام متمایز در پایگاه‌های داده

دریافت تعداد آدرس‌های IP متمایز (یعنی کاربران) دسترسی‌یافته به محصول با شناسهٔ ######## در یک روز معین

- در وب‌سایت شلوغ، رشد جدول بازدیدهای روزانه تا چند میلیارد ردیف
- زمان زیاد اجرا و تاخیر این پرس‌وجوی

شمارش اقلام متمایز در پایگاه‌های داده

تأخیر عمدتاً ناشی از عملیات مرتب‌سازی

- COUNT DISTINCT کلاسیک در بیشتر پایگاه‌های داده (مانند Azure SQL/SQL Server)
- مگر آنکه ستون از قبل مرتب
- پس از مرتب‌سازی ستون، قرارگیری تمام رکوردهای تکراری در کنار یکدیگر
- کفایت پیمایش ترتیبی جهت شناسایی و شمارش اقلام متمایز
- هزینهٔ عملیات مرتب‌سازی در جدولی با n ردیف، $O(n\ logn)$
- مقیاس‌ناپذیری نه تنها برای چند میلیارد ردیف بلکه حتی برای چند میلیون ردیف
- پرس‌وجوهای ساده نیز دارای COUNT DISTINCT-ها و GROUP BY-های متعدد روی ستون‌های مختلف
- عدم تاثیر مرتب‌سازی یک ستون بر کاهش پیچیدگی مرتب‌سازی ستون دیگر
- جهت تسریع کار استفاده از جدول درهم
  - نیاز جدول درهم به فضایی خطی متناسب تعداد مقادیر متمایز k
  - امکان افزایش k تا n
  - مقرون به صرفه نبودن استفاده از درهم

شمارش اقلام متمایز در پایگاه‌های داده

نیاز صرف به دانستن تعداد اقلام متمایز و عدم نیاز به فهرست کردن خود اقلام متمایز

- پابرجایی پیچیدگی
- مسئلهٔ متمایزبودن مقادیر (element-distinctness)
- با داشتن آرایه‌ای از n عنصر،
  - تعیین متمایز بودن تمام مقادیر آن
  - دارای کران پایین $\Omega(n\ log2\ n)$

شمارش اقلام متمایز در پایگاه‌های داده

برای رفع مسائل مقیاس‌پذیری

نسخه‌های جدیدتر سیستم‌های مدیریت پایگاه‌داده و انباره‌های داده

- تخمین کاردینالیته روی آورده‌اند: $SQL\ Server\ 2019$
  - دارای عملیات APPROX_COUNT_DISTINCT
  - استفاده از فضای بسیار کمی و سرعت بالا
- Google BigQuery
  - استفاده از این رویکرد تقریبی و احتمالاتی در COUNT DISTINCT به عنوان پیش‌فرض
  - استفاده از EXACT_COUNT_DISTINCT برای موقعیت‌هایی که مطلقاً به پاسخ دقیق نیاز است
- استفاده از ابرلگ‌لگ در این تخمین‌ها
  - ابداع فلاژوله و همکاران
  - صرفه‌جویی بالا در فضا (در حد کیلوبایت) با وجود کار روی مجموعه‌داده‌هایی در حد تریلیون
  - میزان خطا نسبتاً پایین
  - در مرتبه‌ای $O(\frac{1}{\sqrt{m}})$ که m نشان‌دهندهٔ تعداد شمارنده‌ها یا بیت-نگاشت‌های چند بیتی
- انتخاب رایج برای m مقدار $۲^{۱۴}$

شمارش اقلام متمایز در پایگاه‌های داده

مشاهدهٔ نمونه‌های متعددی از صرفه‌جویی در فضا به بهای کاهش دقت

- معنابخشی دیگر ابرلگ‌لگ به کارایی فضا
- باقیماندن تقریباً همیشه در محدودهٔ چند کیلوبایت
- به دست دادن کاردینالیته واقعی با میانگین نسبت خطای کوچک (مثلاً ۲± درصد)

شمارش خطی

الگوریتم شمارش خطی

- اولین رویکرد احتمالاتی برای مسئله تعداد منحصربه‌فرد
- مورتون آستراهان، ماریو شولنیک و کیو-یانگ وانگ در سال ۱۳۶۶
- پیشنهاد الگوریتم عملی را کیو-یانگ وانگ، براد واندر-زاندن، و هوارد تیلور در سال ۱۳۶۹

طرح اصلی پیشنهادی آنها

- موجب برتری نسبت به روش‌های دقیق
- استفاده درهم مقادیر با تابعی درهم‌ساز
- فراهم شدن امکان حذف تکراری‌ها بدون نیاز به مرتب‌سازی مقادیر
- دارای هزینه نیست: کاهش دقت پاسخ
- احتمال خطا به دلیل تصادم‌های احتمالی درهم
- عدم امکان تشخیص تکراری‌ها خاصه «تکراری‌های تصادفی»

استفاده از چنین جدول درهم به همراه روش پیمایش مناسب

- منجر به بهبود عملکرد فضا و زمان نسبت به روش کلاسیک

شمارش خطی

برای مجموعه داده‌های با تعداد منحصربه‌فرد عظیم

- امکان بزرگ بودن چنین جدول‌های درهم
- نیاز به حافظه‌ای با رشد خطی با تعداد مقادیر متمایز در مجموعه

برای سیستم‌هایی با حافظه محدود

- نیاز به ذخیره‌سازی دیسک یا توزیع شده
- به دلیل دسترسی کند به دیسک یا شبکه کاهش مزایای جدول‌های درهم

شمارش خطی

مشابه ایده فیلتر بلوم برای دور زدن چنین مشکلی

الگوریتم شمارش خطی

- ذخیره نکردن خود مقادیر درهم
- ذخیرهٔ بیت‌های متناظر آنها
- جانشینی جدول درهم با آرایهٔ بیتی (تحت «شمارندهٔ خطی») به طول m
- فرض بر تناسب m با تعداد مقادیر متمایز مورد انتظار n
- اما نیاز به یک بیت به ازای هر مقدار برای اکثر موارد

شمارش خطی

در ابتدا، همه بیت‌ها در «شمارندهٔ خطی» برابر با صفر

درج مقدار جدید x

- محاسبهٔ مقدار درهم‌ساز h(x) متناظر
- برابر یک قرار دادن مقدار بیت در نمایه متناظر در شمارنده

الگوریتم

الگوریتم ۱– درج در شمارنده خطی

ورودی: مقدار $x \in D$

ورودی: شمارنده خطی با تابع درهم‌ساز h

$j \leftarrow h(x)$

اگر $\text{LINEARCOUNTER}[j] = 0$ آنگاه

$LINEARCOUNTER[j] \leftarrow 1$

شمارش خطی

به دلیل استفاده از تک تابع درهم‌ساز h

- امکان تصادم
  - دو مقدار ورودی متفاوت منجر به مقداردهی یک بیت در آرایه
- در نتیجه تعداد تصادم‌های فراوان
- عدم امکان به دست آوردن مستقیم تعداد دقیق (یا حتی نزدیک به دقیق) مقادیر متمایز از چنین طرح فشرده‌ای

شمارش خطی

ایده الگوریتم منجر به توزیع مقادیر در سطل‌ها

- بیت‌های نمایه شدهٔ مقادیر درهم
- آرایهٔ بیتی نمایشگر مقداردهی شدن مدخل‌ها
- مشاهده تعداد یک‌ها در آرایه تخمین تعداد منحصربه‌فرد

شمارش خطی

در مرحله اول الگوریتم شمارش خطی،

- ایجاد داده‌ساختار

با داشتن چنین طرح فشرده‌ای

- تخمین تعداد منحصربه‌فرد با استفاده از کسر مشاهده شده بیت‌های خالی V به صورت زیر

$$n \approx -m \cdot \ln V$$

شمارش خطی

تاثیر تصادم‌ها بر تخمین تعداد منحصربه‌فرد در الگوریتم شمارش خطی

با هر تصادم کاهش تعداد بیت‌هایی که باید تنظیم شوند

کسر مشاهده شده بیت‌های تنظیم نشده بزرگتر از مقدار واقعی

نداشتن تصادم درهم

- تعداد نهایی بیت‌های تنظیم شده برابر تعداد منحصربه‌فرد مورد نظر
- اما اجتناب‌ناپذیری تصادم‌ها
- معادله به مثابهٔ تخمینی بیش از حد از تعداد دقیق منحصربه‌فرد

مقدار صحیح بودن تعداد منحصربه‌فرد

- گرد کردن نتیجه آن به نزدیک‌ترین عدد صحیح کوچکتر

شمارش خطی

الگوریتم ۲- تخمین تعداد منحصربه‌فرد با شمارش خطی

ورودی: مجموعه داده D

خروجی: تخمین تعداد منحصربه‌فرد

$$LINEARCOUNTER[i] \leftarrow ۰, i = ۰ \dots m - ۱$$

برای هر $x \in D$ انجام بده

$$Add(x, LINEARCOUNTER)$$

$$Z \leftarrow count_{i=۰ \dots m-۱} (LINEARCOUNTER[i] = ۰)$$

برگرداندن $\left\lfloor -m \times \ln \frac{Z}{m} \right\rfloor$

شمارش خطی-مثال

مجموعه داده‌ای شامل بیست نام از پایتخت‌ها

- **برلین**، برلین، **پاریس**، برلین، **لیسبون**، **کیف**، پاریس، **لندن**، **رم**، **آتن**، **مادرید**، **وین**، رم، رم، لیسبون، برلین، پاریس، لندن، کیف، **واشنگتن**
- تعداد منحصربه‌فرد کوچک
  - تعداد دقیق منحصربه‌فرد ۱۰ است

برای داشتن خطای استاندارد حدود ده درصد

- نیاز به انتخاب طول داده‌ساختار حداقل به اندازه تعداد مقادیر منحصربه‌فرد مورد انتظار انت
  - $m = 2^4$
- تابع درهم‌ساز h با مقادیر در $\{0,1, \dots, 2^4 - 1\}$از تابعی مبتنی بر مورمور۳ نوع ۳۲ بیتی

$$h(x) = MurmurHash3(x)\%\, m$$

شمارش خطی-مثال

مقادیر درهم

شهر	h (شهر)	شهر	h (شهر)
آتن	۱۲	لندن	۱۴
برلین	۷	لیسبون	۱۵
پاریس	۸	مادرید	۱۴
رم	۱	واشنگتن	۱۱
کیف	۱۳	وین	۶

شمارش خطی-مثال

شهرهای لندن و مادرید دارای مقداری یکسان

انتظار چنین تصادم‌هایی مورد انتظار و کاملاً طبیعی

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۰	۱	۰	۰	۰	۰	۱	۱	۱	۰	۰	۱	۱	۱	۱	۱

شمارش خطی-مثال

با توجه به الگوریتم شمارش خطی

- محاسبهٔ کسر V بیت‌های خالی

$$V = \frac{7}{16}$$

تعداد منحصربه‌فرد تخمین زده شده برابر

$$n \approx -16 \times \ln\frac{7}{16} \approx 13$$

شمارش خطی-ویژگی‌ها

محاسبهٔ V

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

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

در نتیجه، میانگین نسبت یک‌ها برابر است با

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

و N تعدد بیت‌های متناظر یک در فیلتر

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

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

شمارش خطی-ویژگی‌ها

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

$$-\frac{n}{m} = \ln\left(1 - \frac{N}{m}\right)$$

$$n = -m \ln\left(1 - \frac{N}{m}\right), V = 1 - \frac{N}{M}$$

$$\Longrightarrow n = -m \ln V$$

شمارش خطی-ویژگی‌ها

در صورتی که تابع درهم‌ساز h دارای زمان محاسبه‌ای ثابت
- صادق برای غالب توابع درهم‌ساز

زمان پردازش هر مقدار برابر مرتبهٔ O(N)
- N تعداد کل مقادیر از جمله تکراری‌ها.
- الگوریتم دارای پیچیدگی زمانی O(N)

شمارش خطی-ویژگی‌ها

مانند هر الگوریتم احتمالاتی دیگر
- دارای تعدادی پارامتر قابل تنظیم جهت تاثیر بر عملکرد آن

دقت مورد انتظار تخمین
- به اندازهٔ آرایهٔ بیتی m
- نسبت آن به تعداد مقادیر متمایز $\alpha = \frac{m}{n}$
  - $\alpha \geq 1$
  - m > n که در عمل جالب نیست

احتمالی غیرصفر اشباع pr
- پر شدن آرایهٔ بیتی یا احتمال اشباع
- مخدوش‌سازی الگوریتم و بی‌نهایت کردن مقدار معادله
- وابستگی احتمال اشباع pr به ضریب بار و در نتیجه به اندازهٔ m
- انتخاب به اندازهٔ کافی بزرگ
  - جهت ناچیز کردن احتمال اشباع

شمارش خطی

خطای استاندارد δ

معیاری از درستی تخمین حاصل از شمارش خطی

تعادل بین آن و اندازه آرایه بیتی m

کاهش خطای استاندارد منجر به تخمین‌های دقیق‌تر، اما افزایش حافظه موردنیاز

جدول ۳. تعادل بین دقت و اندازه آرایه بیتی

n	m	
	$\delta = ۱٪$	$\delta = ۱۰٪$
۱۰۰۰	۵۳۲۹	۲۶۸
۱۰۰۰۰	۷۹۶۰	۱۷۰۹
۱۰۰۰۰۰	۲۶۷۲۹	۱۲۷۴۴
۱۰۰۰۰۰۰	۱۵۴۱۷۱	۱۰۰۸۸۰
۱۰۰۰۰۰۰۰	۱۰۹۶۵۸۲	۸۳۱۸۰۹
۱۰۰۰۰۰۰۰۰	۸۵۷۱۰۱۳	۷۰۶۱۷۶۰

شمارش خطی

وابستگی به انتخاب m بسیار پیچیده و نداشتن راه‌حل تحلیلی

نویسندگان الگوریتم مقادیر از پیش محاسبه شده‌ای برای احتمال قابل قبول تکمیل

پیشنهاد $pr_{\text{اشباع}} = 0.7\%$

- استفاده به عنوان مرجع
- به دلیل صفر نبودن احتمال تکمیل، نادر بودن پر شدن آرایه بیتی
- هنگام کار با مجموعه‌های داده کوچک،
  - امکان نمایه‌سازی دوبارهٔ همه مقادیر با تابع درهم‌ساز متفاوت
  - یا افزایش اندازهٔ شمارندهٔ خطی
- ناکارآمدی چنین راه‌حل‌هایی برای مجموعه‌های داده عظیم
  - همراه با پیچیدگی زمانی نسبتاً بالا

شمارش خطی

مناسب بودن شمارش خطی

- بزرگ نبودن تعداد منحصربه‌فرد مجموعه داده‌ای
- امکان استفاده از آن در بهبود الگوریتم‌های دیگر برای تعداد منحصربه‌فردهای عظیم
- تخمین تعداد منحصربه‌فرد تقریباً متناسب با مقدار دقیق
  - دلیل اصطلاح «خطی»

شمارش عناصر منحصربه‌فرد در فیلتر بلوم

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

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.

شمارش احتمالاتی

فیلیپ فلاژوله و جی. نایجل مارتین

- سال ۱۳۶۴

معرفی الگوریتم شمارشی بر اساس مشاهده الگوهای متداول در نمایش‌های درهم شده مقادیر نمایه شده

اعمال تابع درهم‌ساز h بر مقدار ورودی

تبدیل آن به عددی از اعداد صحیح با توزیع نزدیک-یکنواخت در بازهٔ $\{0,1, \dots ,2^{\ell} - 1\}$

- یا به طور معادل، تبدیل به رشته‌ای از مجموعه رشته‌های دودوئی به طول ℓ
- $h(x) = i = \sum_{j=0}^{\ell-1} 2^j \times i_j = (i_{\ell-1} \dots i_1 i_0)_2, i_j \in \{0,1\}$

شمارش احتمالاتی

فلاژوله و مارتین تشخیص دادند

- هم‌احتمال بودن حضور الگوهای $10^j = 1\overbrace{00\ldots0}^{j\text{ بار}}$ در رشته‌های دودوئی
- وجود چنین مقداری در رشتهٔ دودوئی با احتمال $2^{-(j+1)}$
- در صورت ثبت شدن برای هر مقدار نمایه شده
  - امکان ایفای نقش تخمین‌گر تعداد منحصربه‌فرد
  - مرتبط کردن هر الگو با شاخص آن، به نام «رتبه»:

$$\text{رتبه}(i) = \begin{cases} \min_{i_j \neq 0} j\,, \forall i > 0 \\ \ell, \qquad i = 0 \end{cases}$$

اساسا متناظر موقعیت سمت راست‌ترین ۱

- ۱ با کمترین اهمیت

شمارش احتمالاتی- مثال

عدد صحیح ۸ بیتی ۴۲ با:

$$42$$
$$= 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= (00101010)_2$$

ظاهر شدن یک‌ها در موقعیت‌های ۱، ۳ و ۵. با توجه به تعریف، رتبهٔ ۴۲ برابر با

$$\text{رتبه}(42) = min\,(1,3,5) = 1$$

شمارش احتمالاتی

رخدادهای الگوی 10^j،

- یا متناظرا $j + ۱ = (۰)$رتبه،
- نمایه شده در نمایش‌های دودوئی مقادیر درهم هر مقدار
- امکان ذخیره به صورت فشرده در داده‌ساختار آرایه‌ای شمارنده به طول ℓ
  - مشهور به «طرح ف‌م»

در ابتدا، همه بیت‌ها در شمارنده برابر با صفر

هنگام درج مقدار جدید x در داده‌ساختار،

محاسبهٔ مقدار درهم آن با تابع درهم‌ساز h

محاسبه رتبه متناظر

برابر یک کردن بیت متناظر در آرایه

شمارش احتمالاتی

الگوریتم ۳: درج مقدار در شمارنده ساده

ورودی: مقدار $x \in D$

ورودی: شمارنده ساده با تابع درهم‌ساز h

$j \leftarrow$ رتبه $\big(h(x)\big)$

اگر $COUNTER[j] == ۰$ آنگاه

$COUNTER[j] \leftarrow ۱$

شمارش احتمالاتی

یک در شمارنده در موقعیت $j + 1$

- به معنای حداقل یک بار مشاهدهٔ الگوی 10^j در میان مقادیر درهم همهٔ ورودی‌های نمایه شده

شمارش احتمالاتی

الگوریتم ۴- محاسبه رتبه X

ورودی: عدد صحیح X

خروجی: رتبه X یا کوچکترین j که بیت j-ام برابر ۱

$j \leftarrow ۰$

$l = len(x)$

تازمانی که $j < l$ ادامه بده

اگر $(x \gg j \;\&\; ۱) == ۱$

برگرداندن j

$j \leftarrow j + ۱$

برگرداندن l // عملاً رخ نمی‌دهد چون h(x) غیرصفر

شمارش احتمالاتی- مثال

مجموعه داده بیست نام از پایتخت‌ها اعم از

برلین، برلین، پاریس، برلین، لیسبون، کیف، پاریس، لندن، رم، آتن، مادرید، وین، رم، رم، لیسبون، برلین، پاریس، لندن، کیف، واشنگتن

از مورمور۳- ۳۲ بیتی به عنوان تابع درهم‌ساز h

نگاشت به $\{0,1, \dots ,2^{۳۲} - 1\}$

استفاده از شمارنده‌ای به طول $\ell = 32$

با استفاده از مقادیر درهم

محاسبهٔ رتبه‌های هر مقدار

شمارش احتمالاتی-مثال

شهر	h(شهر)	رتبه	شهر	h(شهر)	رتبه
آتن	۴۱۶۱۴۹۷۸۲۰	۲	لندن	۳۴۵۰۹۲۷۴۲۲	۱
برلین	۳۶۸۰۷۹۳۹۹۱	۰	لیسبون	۶۲۹۵۵۵۲۴۷	۰
پاریس	۲۶۷۳۲۴۸۸۵۶	۳	مادرید	۲۹۷۰۱۵۴۱۴۲	۱
رم	۵۰۱۲۲۷۰۵	۰	واشنگتن	۴۰۳۹۷۴۷۹۷۹	۰
کیف	۳۴۹۱۲۹۹۶۹۳	۰	وین	۳۲۷۱۰۷۰۸۰۶	۱

شمارش احتمالاتی-مثال

شمارنده

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۱	۱	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰	۳۱
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

شمارش احتمالاتی

با فرض توزیع یکنواخت مقادیر

فرض تعداد مقادیر متمایز نمایه شده تاکنون برابر n

ظاهر شدن مقدار ۱ در مدخل اول در حدود $\frac{n}{2}$ دفعه

ظاهر شدن مقدار یک در مدخل دوم در حدود $\frac{n}{2^2}$ دفعه

ادامه به همین منوال

ظهور یک در موقعیت j برابر $\frac{n}{2^j}$ دفعه

شمارش احتمالاتی

اگر $j \gg log_2 n$

- احتمال مشاهدهٔ مقدار ۱ در موقعیت j-ام میل به صفر
- در نتیجه مقدار $Counter[j]$ قریب به یقین برابر صفر

اگر $j \ll log_2 n$

- مقدار Counter[k] قریب به یقین برابر با یک

اگر مقدار $j \approx log_2 n$

- برابر بدون احتمال مشاهدهٔ ۱ یا ۰ در آن موقعیت

شمارش احتمالاتی

سخن کوتاه

- پس از درج تمام مقادیر مجموعه داده در آرایهٔ شمارنده
- سمت راست‌ترین موقعیتی با مقدار صفر ظاهر می‌شود
- به عنوان شاخص تخمین $log_2 n$
- فلاژوله و مارتین در عمل، نیاز به اعمال ضریب تصحیح ϕ جهت بهبود تخمین

تخمین کاردینالیتی (تعداد مقادیر متمایز)

$$n \approx \frac{1}{\phi} 2^R$$

- $\phi \approx 0.77351$

شمارش احتمالاتی

سخن کوتاه

- استفاده فلاژوله و مارتین از موقعیت کم‌ارزش‌ترین بیت صفر (چپ‌ترین صفر در آرایه) به عنوان تخمین کاردینالیتی
- تعریف الگوریتم بر همین اساس

با توجه به مشاهدهٔ قبلی

- امکان استفاده از موقعیت پرارزش‌ترین بیت ۱ (راست‌ترین ۱) به همین منظور
- اما دارای توزیع یکنواخت‌تر و در نتیجه خطای استاندارد بزرگ‌تر

شمارش احتمالاتی

الگوریتم ۵- محاسبه موقعیت سمت چپ‌ترین صفر

ورودی: شمارنده ساده به طول ℓ

خروجی: سمت چپ‌ترین موقعیت صفر

برای j از ۰ تا $\ell - 1$ انجام بده

اگر $COUNTER[j] == 0$ آن‌گاه

برگرداندن j

برگرداندن ℓ

شمارش احتمالاتی-مثال

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۱	۱	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰	۳۱
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

با استفاده از الگوریتم 5، در شمارنده ظاهر شدن چپ‌ترین مقدار ۰ در موقعیت $R = 4$

تخمین تعداد منحصربه‌فرد

$$n \approx \frac{1}{0.77351} 2^{۴} \approx 20.68$$

- تعداد دقیق منحصربه‌فرد مجموعه ۱۰
- خطای زیاد تخمین
  - دلیل عدد صحیح بودن مقادیر R
- برای رتبه‌های بسیار نزدیک
  - امکان حصول نتایج دارای تفاوتی از مرتبه توان‌های دودوئی
  - در مثال حاضر، R=3 برگرداندن تخمین نسبتا مناسب ۱۰,۳۴

شمارش احتمالاتی

تخمین تعداد منحصربه‌فرد بر اساس شمارنده معمول

- بدست آوردن مقادیر میانگین بسیار نزدیکی، اما با وردایی نسبتاً بالا
  - در مثال قبل، با خطای استاندارد δ دارای اختلاف از مرتبه دو برابر

ضعف رویکرد تک‌شمارنده

- فقدان تخمین‌های با اطمینان بالا برای تعداد منحصربه‌فرد
- انجام پیش‌بینی تنها بر اساس یک تخمین

بهبود منطقی

- استفاده از تعداد زیادی شمارنده
  - منجر به افزایش تعداد تخمین‌ها
- پیش‌بینی نهایی n با میانگین‌گیری از پیش‌بینی‌های R_i حاصل شمارنده‌های مزبور $\{\text{شمارنده}_i\}_{i=0}^{m-1}$

سخن کوتاه، معادله اصلاح شده الگوریتم شمارش احتمالاتی:

$$n \approx \frac{1}{\phi} 2^{\frac{1}{m}\sum_{i=0}^{m-1} R_i}$$

شمارش احتمالاتی

تعداد منحصربه‌فرد n

- مقدار تخمینی با همان کیفیت اما با وردایی بسیار کوچکتر
- اشکال عملی برای ساخت m شمارنده ساده مستقل،
  - نیاز به محاسبه مقادیر mتابع درهم‌ساز متفاوت
- محاسبهٔ هر تابع درهم‌ساز در O(1)
- محاسبه همهٔ آنها دارای پیچیدگی زمانی O(m) و هزینه‌های پردازندهٔ نسبتاً بالا

شمارش احتمالاتی

راه‌حل بهینه‌سازی الگوریتم شمارش احتمالاتی

- اعمال «میانگین‌گیری تصادفی»
- جانشینی تمامی m تابع درهم‌ساز با صرفا یک تابع
- تعریف m شمارنده
- تقسیم مقدار حاصل به دو بخش خارج‌قسمت و باقیمانده
- باقیمانده r برای انتخاب یکی از m شمارنده و خارج‌قسمت q برای محاسبه رتبه
- یافتن شاخص مناسب برای به‌روزرسانی در آن شمارنده

شمارش احتمالاتی

الگوریتم ۶: استفاده از میانگین‌گیری برای به‌روزرسانی شمارنده‌ها

ورودی: مقدار $x \in D$

ورودی: آرایه‌ای از m شمارنده ساده با تابع درهم‌ساز h

$f_r \leftarrow h(x) \% m$

$f_q \leftarrow \left\lfloor \frac{h(x)}{m} \right\rfloor$

$j \leftarrow rank(f_q)$

$COUNTER_{f_r}[j] \leftarrow ۱$

شمارش احتمالاتی

اعمال الگوریتم میانگین‌گیری تصادفی به شمارش احتمالاتی،

- با فرض کفایت منصفانه بودن توزیع مقادیر مبتنی بر خارج‌قسمت

انتظار نمایه شدن $\frac{n}{m}$ از کل مقدار در هر شمارنده $\left\{\text{شمارنده}_i\right\}_{i=0}^{m-1}$

در نتیجه: معادله قبلی تخمین خوبی برای $\frac{n}{m}$

- نه مستقیما n
- در نتیجه پیشنهاد فلاژوله و مارتین

$$n \approx \frac{m}{\phi}\ 2^{\bar{R}}\ = \frac{m}{\phi}\ 2^{\frac{1}{m}\sum_{i=0}^{m-1} R_i}$$

شمارش احتمالاتی

الگوریتم ۷: الگوریتم فلاژوله-مارتین (PCSA)

ورودی: مجموعه داده D

ورودی: آرایه‌ای از m شمارنده ساده با تابع درهم‌ساز h

خروجی: تخمین تعداد منحصربه‌فرد

برای $x \in D$ انجام بده

$f_r \leftarrow h(x)\%\, m$

$f_q \leftarrow \left\lfloor \frac{h(x)}{m} \right\rfloor$

$j \leftarrow rank(f_q)$

$COUNTER_{f_r}[j] \leftarrow ۱$

برای r از ۰ تا $m - ۱$ انجام بده

$R \leftarrow LeftMostZero(COUNTER_{f_r})$

$S \leftarrow S + R$

$S \leftarrow \frac{۱}{\mathrm{m}} S$

برگرداندن $\frac{m}{\phi}\, ۲^{S}$

شمارش احتمالاتی

الگوریتم شمارش احتمالاتی با میانگین‌گیری تصادفی (PCSA)

- همچنین الگوریتم فلاژوله-مارتین
- در مقایسه با نسخه‌ای که از m تابع درهم‌ساز
  - کاهش پیچیدگی زمانی به حدود $O(1)$

شمارش احتمالاتی-مثال

تخمین تعداد منحصربه‌فرد با میانگین‌گیری تصادفی

اعمال میانگین‌گیری تصادفی با شبیه‌سازی $m = 3$ تابع درهم‌ساز

استفاده از باقیمانده r برای انتخاب یکی از سه شمارنده و از خارج‌قسمت q برای محاسبه رتبه

شهر	h(شهر)	r	Q	رتبه(q)	شهر	h(شهر)	r	q	رتبه(q)
آتن	۴۱۶۱۴۹۷۸۲۰	۰	۱۳۸۷۱۶۵۹۴۰	۲	لندن	۳۴۵۰۹۲۷۴۲۲	۲	۱۱۵۰۳۰۹۱۴۰	۲
برلین	۳۶۸۰۷۹۳۹۹۱	۱	۱۲۲۶۹۳۱۳۳۰	۱	لیسبون	۶۲۹۵۵۵۲۴۷	۰	۲۰۹۸۵۱۷۴۹	۰
پاریس	۲۶۷۳۲۴۸۸۵۶	۰	۸۹۱۰۸۲۹۵۲	۳	مادرید	۲۹۷۰۱۵۴۱۴۲	۲	۹۹۰۰۵۱۳۸۰	۲
رم	۵۰۱۲۲۷۰۵	۱	۱۶۷۰۷۵۶۸	۴	واشنگتن	۴۰۳۹۷۴۷۹۷۹	۲	۱۳۴۶۵۸۲۶۵۹	۰
کیف	۳۴۹۱۲۹۹۶۹۳	۱	۱۱۶۳۷۶۶۵۶۴	۲	وین	۳۲۷۱۰۷۰۸۰۶	۱	۱۰۹۰۳۵۶۹۳۵	۰

شمارش احتمالاتی-مثال

هر شمارنده مدیریت اطلاعات مربوط به حدود یک سوم شهرها

توزیع به اندازه کافی منصفانه

پس از نمایه‌سازی همه مقادیر و تنظیم بیت‌های مناسب در شمارنده‌های مربوطه،

به دست آوردن شمارنده‌ها

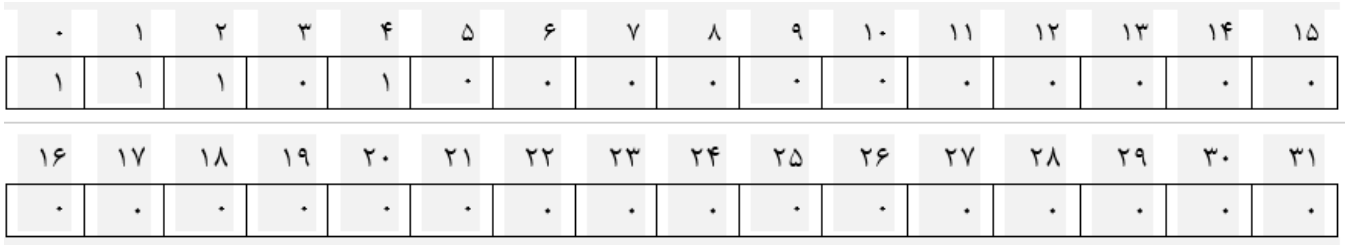
شمارش احتمالاتی-مثال

شمارنده ۰

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۰	۱	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰	۳۱
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

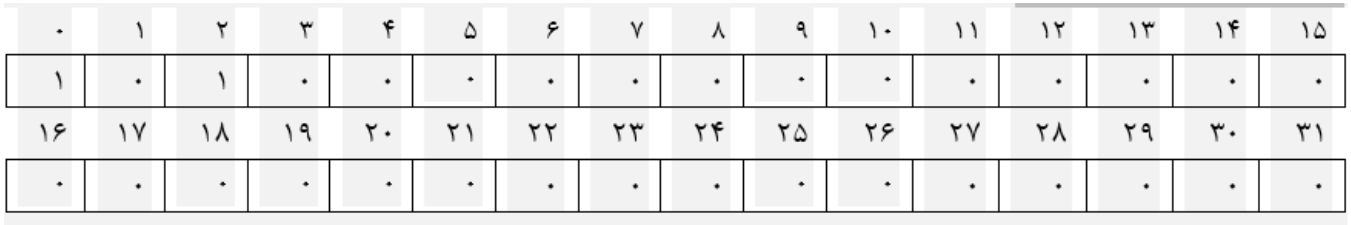
شمارنده ۱



۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۱	۱	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰	۳۱
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

شمارنده ۲



۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
۱	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

۱۶	۱۷	۱۸	۱۹	۲۰	۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰	۳۱
۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰

شمارش احتمالاتی-مثال

سمت چپ‌ترین موقعیت‌های صفر برای هر شمارنده

- $R_0 = 1$
- $R_1 = 3$
- $R_2 = 1$

$$n \approx \frac{۳}{\phi}\ 2^{\frac{1}{3}\sum_{i=0}^{2} R_i} \approx \frac{۳}{0.77351}\ 2^{\frac{1+3+1}{3}} \approx 12.31$$

تخمین حاصل نزدیک‌تر به مقدار واقعی تعداد منحصربه‌فرد ۱۰

- بدون استفاده از شمارنده‌های زیاد بهتر از مثال قبلی

شمارش احتمالاتی-ویژگی‌ها

مناسب بودن الگوریتم فلاژوله-مارتین برای مجموعه‌های داده با تعداد منحصربه‌فرد زیاد

تقریب خوب در نسبت‌های $\frac{n}{m} > 20$

جهت بهبود نتایج در مجموعه داده‌های با تعداد منحصربه‌فردهای کوچک

- امکان استفاده از الگوریتم‌های غیرخطی در الگوریتم
- بیورن شویرمان و مارتین ماوه در سال ۱۳۸۶ پیشنهاد بهبودی احتمالی
- متناسب‌سازی با افزودن عبارتی برای تعداد منحصربه‌فردهای کوچک
- همگرا شدن به صفر برای تعداد منحصربه‌فردهای بزرگ

$$n \approx \frac{m}{\phi}\left(2^{\bar{R}} - 2^{-\chi \times \bar{R}}\right)$$

$$\chi \approx 1.75$$

شمارش احتمالاتی-ویژگی‌ها

رابطه معکوس خطای استاندارد δ الگوریتم فلاژوله-مارتین با تعداد شمارنده‌های مورد استفاده

$$\delta \approx \frac{0.78}{\sqrt{m}}$$

شمارش احتمالاتی-ویژگی‌ها

مقادیر مرجع خطای استاندارد برای تعداد شمارنده‌های پرکاربرد

طول ℓ هر شمارنده

$$\ell > log_2\left(\frac{n}{m}\right) + 4$$

- کفایت $\ell = 32$

شمارش تعداد منحصربه‌فردهای بسیار فراتر از 10^9 با استفاده از ۶۴ شمارنده

$$\left(\frac{n}{m}\right)/2^{\ell} \leq 2^{-4}$$

$$\frac{n}{m} \leq 2^{\ell-4}$$

$$\ell - 4 \geq \log\frac{n}{m_2}$$

$$\ell \geq \log_2\left(\frac{\mathrm{n}}{m}\right) + 4$$

شمارش احتمالاتی-ویژگی‌ها

ادغام شمارنده‌های متناظر مجموعه داده‌های مختلف با اعمال عملیات OR بیتی

- نتیجه: شمارندهٔ واحد برای کل مجموعه داده‌ها

عدم پشتیبانی از حذف مانند فیلتر بلوم،

- افزودن امکان حذف با پیروی از رویکرد فیلتر بلوم شمارنده
  - پشتیبانی از حذف‌های احتمالا درست
- گسترش آرایه‌های بیتی داخلی با شمارنده‌ها
  - منجر به افزایش فضای ذخیره‌سازی

روش لگ‌لگ

الگوریتم‌های احتمالاتی تخمین تعداد منحصربه‌فرد پرکاربرد

- با استفاده عملی
- خانواده لگ‌لگ از الگوریتم‌ها
- شامل الگوریتم لگ‌لگ پیشنهادی ماریان دوراند و فیلیپ فلاژوله در سال ۱۳۸۲
- دو روش جانشین آن ابرلگ‌لگ و ابرلگ‌لگ++
- استفاده از رویکردی مشابه الگوریتم شمارش احتمالاتی

تخمین تعداد منحصربه‌فرد n با مشاهده حداکثر تعداد صفرهای ابتدایی در نمایش دودوئی مقادیر

نیاز به حافظه کمکی

پیمایش یکبارهٔ داده‌ها جهت تولید تخمینی از تعداد منحصربه‌فرد

روش لگ‌لگ

مطابق منوال روش‌های قبلی تعیین تعداد منحصر بفرد اجزا

پیش‌پردازش هر مقدار در مجموعه داده با اعمال تابع درهم‌سازی

- تبدیل مقادیر به اعداد صحیح با توزیع کافی یکنواخت در بازهٔ $\{0,1, \dots, 2^{\ell-1}\}$ یا به طور معادل، بر روی مجموعه رشته‌های دودوئی به طول ℓ

$$h(x) = i = \sum_{j=0}^{\ell-1} i_j 2^j = (i_{\ell-1} \dots i_1 i_0)_2, i_j \in \{0,1\}$$

روش لگ‌لگ

مراحل الگوریتم‌ها مشابه PCSA

ابتدا، تقسیم مجموعه دادهٔ اولیه یا جریان ورودی به تعدادی زیرمجموعه

- نمایه شدن هر کدام با یکی از m شمارنده

تابع درهم‌ساز واحد

- همانند میانگین‌گیری تصادفی
  - انتخاب شمارنده برای مقدار خاص x
  - استفاده از بخشی از مقدار درهم $h(x)$
- بخش دیگر برای به‌روزرسانی شمارنده مربوطه

روش لگ‌لگ

همه الگوریتم‌های مورد بحث بر اساس مشاهدهٔ الگوهای 10^j

- در ابتدای مقادیر برای شمارنده خاص
- هر الگو با شاخص آن، به نام رتبه،

نمایه شدن رتبه معادل موقعیت ۱ با کمترین اهمیت در نمایش دودوئی مقدار درهم

مشاهده تعداد منحصربه‌فرد در هر شمارنده بر اساس رتبه‌های دیده شده

تخمین نهایی تعداد منحصربه‌فرد از چنین مشاهداتی با استفاده از تابع ارزیابی

روش لگ‌لگ

ذخیره‌سازی

- شمارنده‌ها در الگوریتم شمارش احتمالاتی نسبتا پرهزینه برای نگهداری
- در مقابل: الگوریتم لگ‌لگ راه‌حلی کارآتر از نظر ذخیره‌سازی
- همراه با تابع ارزیابی بهتر و رویکرد تصحیح سوگیری

روش لگ‌لگ

الگوریتم لگ‌لگ

بر اساس محاسبهٔ رتبهٔ هر مقدار ورودی

با تابع درهم‌ساز واحد h

فرض n تعداد کل مقادیر نمایه شده در شمارنده

- ادعا $\frac{n}{2^j}$ مقدار اختیار کردن رتبهٔ j
- پس رتبهٔ بیشینه مشاهده شده نشانه و تقریب مناسبی از مقدار $\log_2 n$

$$R = max_{x \in D} \left(\text{رتبه} \left(h(x) \right) \right) \approx log_2 n$$

روش لگ‌لگ

تخمین مذکور دارای خطایی با مرتبهٔ ۱٫۸۷ برابر مقدار واقعی

- عملا بی‌معنی

برای کاهش خطا،

- استفاده از سطل‌بندی مبتنی بر میانگین‌گیری تصادفی
- تقسیم مجموعه داده به $m = 2^p$ زیرمجموعه $\{S_0, S_1, \ldots, S_{m-1}\}$
- پارامتر دقت p تعداد بیت‌های استفاده شده در پیمایش

بنابراین، برای هر مقدار x از مجموعه داده، p بیت اول مقدار درهم ℓ-بیتی h(x)

یافتن شاخص j زیرمجموعه مناسب

$$j = \left(i_{p-1} \ldots i_1 \, i_0 \right)_2$$

نمایه شدن بقیه (ℓ - p) بیت در شمارنده j-ام

محاسبه رتبه و مشاهده R_j

روش لگ‌لگ

با توزیع منصفانه،

- هر زیرمجموعه دارای n/m عضو
- پس، R_j-ها در شمارنده‌ها $\{COUNTER[j]\}_{j=0}^{m-1}$ نشانه‌ای از مقدار $\log_2 \frac{n}{m}$
- استفاده از میانگین حسابی آنها و تصحیح سوگیری، کاهش وردائی مشاهده

$$n = \alpha_m \cdot m \cdot 2^{\frac{1}{m}\sum_{j=0}^{m-1} R_j}$$

α_m، و $\Gamma(\cdot)$ تابع گاما

$$\alpha_m = \left(\Gamma\left(-\frac{1}{m}\right) \times \frac{1-2^{\frac{1}{m}}}{\log 2}\right)^m \approx 0.39701 - \frac{2\pi^2+(ln2)^2}{48m}$$

$$\Gamma(\alpha) = \int_0^\infty e^{-x} x^{\alpha-1} dx$$

$$\Gamma(n) = (n-1)!$$

برای اکثر موارد عملی $m \geq 64$ کفایت استفاده از $\alpha_m \approx 0.39701$

روش لگ‌لگ

الگوریتم ۸- تخمین تعداد منحصربه‌فرد با LogLog

ورودی: مجموعه داده D

ورودی: آرایه‌ای از m شمارنده LogLog با تابع درهم‌ساز h

خروجی: تخمین تعداد منحصربه‌فرد

$$COUNTER\,[j] \leftarrow 0, j = 0 \dots m - 1$$

برای $x \in D$ انجام بده

$$i \leftarrow h(x) = (i_{\ell-1} \dots i_1 i_0)_2, i_k \in \{0,1\}$$

$$j \leftarrow \left(i_{p-1} \dots i_1 i_0\right)_2$$

$$r \leftarrow rank\left(\left(i_{\ell-1} \dots i_{p+1} i_p\right)_2\right)$$

$$COUNTER[j] \leftarrow max(COUNTER[j], r)$$

$$R \leftarrow \frac{1}{m}\sum_{j=0}^{m-1} Counter[j]$$

برگرداندن $\alpha_m \cdot m \cdot 2^R$

روش لگ‌لگ-مثال

فرض کنید m=4 پس در نتیجه p = log4=2 بیت جهت انتخاب شمارنده

خروجی درهم طولی برابر هشت بیت

پس بیت‌های صفر و یک برای شمارنده و بقیه شش بیت برای محاسبهٔ رتبه

شهر	h(x) (۸ bits)	(b_1b_0)	($b_7b_6b_5b_4b_3b_2$)
Tehran	۰۱۱۰۱۰۱۰	۱۰ = ۲	۰۱۱۰۱۰
Paris	۱۰۱۱۰۰۱۱	۱۱ = ۳	۱۰۱۱۰۰
London	۰۰۰۱۱۱۰۰	۰۰ = ۰	۰۰۰۱۱۱
Berlin	۱۱۰۰۱۰۱۰	۱۰ = ۲	۱۱۰۰۱۰
Rome	۰۰۱۰۱۱۰۱	۰۱ = ۱	۰۰۱۰۱۱
Madrid	۰۱۰۰۱۱۱۰	۱۰ = ۲	۰۱۰۰۱۱
Kyiv	۱۰۰۰۰۱۱۱	۱۱ = ۳	۱۰۰۰۰۱
Vienna	۱۱۱۰۰۰۰۰	۰۰ = ۰	۱۱۱۰۰۰

روش لگ‌لگ-مثال

حساب رتبه بیت‌های بالایی

شهر	$(b_7b_6b_5b_4b_3b_2)$	رتبه
Tehran	۰۱۱۰۱۰	۱
Paris	۱۰۱۱۰۰	۲
London	۰۰۰۱۱۱	۰
Berlin	۱۱۰۰۱۰	۱
Rome	۰۰۱۰۱۱	۰
Madrid	۰۱۰۰۱۱	۰
Kyiv	۱۰۰۰۰۱	۰
Vienna	۱۱۱۰۰۰	۳

روش لگ‌لگ-مثال

بروز شدن شمارنده‌ها با عملیات بیش

شمارنده j	شهرهای تخصیص یافته	رتبه	بیش شمارنده
۰	لندن با r=۰ و وین با r=۳	{۰,۳}	۳
۱	رم با r=۰	{۰}	۰
۲	طهران با r=۱ و برلین با r=۱ و مادرید با r=۰	{۰,۱,۱}	۱
۳	پاریس با r=۲ و کیف با r=۰	{۰,۲}	۲

روش لگ‌لگ-مثال

پس شمارنده دارای مقدار [3,0,1,2] است

محاسبهٔ میانگین را به صورتی زیر حساب می‌کنیم.

$$R = \frac{1}{m}\sum \text{شمارنده}[j] = \frac{1}{4}(3+0+1+2) = 1.5$$

نتیجه نهایی تخمین برای m=4 و $\alpha_4 \approx 0.39701$ برابر است با

$$\hat{n} = \alpha_m \times m \times 2^R = 0.39701 \times 4 \times 2^{1.5}$$

روش لگ‌لگ-ویژگی‌ها

رابطه معکوس خطای استاندارد δ الگوریتم لگ‌لگ با تعداد شمارنده‌های استفاده شده m

$$\delta \approx \frac{1.3}{\sqrt{m}}$$

$m = 256$ خطای استاندارد حدود هشت درصد

$m = 1024$ حدود چهار درصد

روش لگ‌لگ-ویژگی‌ها

در صورت شمارش تا n

فضای ذخیره‌سازی موردنیاز الگوریتم لگ‌لگ حدود $O(\log_2 \log_2 n)$

روش لگ‌لگ-ویژگی‌ها

در صورت شمارش تا n

فضای ذخیره‌سازی موردنیاز الگوریتم لگ‌لگ حدود $O(\log_2 \log_2 n)$

مقدار دقیق کل فضای مورد نیاز الگوریتم برای شمارش تا n

$$m.\log_2 \log_2 \frac{n}{m}\left(1 + O(1)\right)$$

روش لگ‌لگ-ویژگی‌ها

در صورت شمارش تا n

فضای ذخیره‌سازی موردنیاز الگوریتم لگ‌لگ حدود $O(\log_2 \log_2 n)$

مقدار دقیق کل فضای مورد نیاز الگوریتم برای شمارش تا n

$$m.\log_2 \log_2 \frac{n}{m}\left(1 + O(1)\right)$$

در مقایسه با الگوریتم شمارش احتمالاتی

- نیاز هر شمارنده به ۱۶ یا ۳۲ بیت
- الگوریتم لگ‌لگ نیاز به شمارنده‌های بسیار کوچکتر $\{COUNTER[j]\}_{j=0}^{m-1}$
  - معمولاً هر کدام پنج بیت
- با وجود مصرف حافظهٔ کمتر الگوریتم لگ‌لگ نسبت به الگوریتم شمارش احتمالاتی
  - دارای میزان ناچیزی از کاهش دقت نسبت به شمارش احتمالاتی

روش لگ‌لگ

نیاز به شمارش تعداد منحصربه‌فرد تا $2^{۳۰}$

- یعنی حدود ۱ میلیارد، با دقت حدود چهار درصد
- برای چنین خطای استانداردی، $m = 1024$ سطل مورد نیاز
- هر کدام $\frac{n}{m} = 2^{20}$ مقدار
- تعداد بیت موردنیاز $\log_2 \log_2 ۲^{20} \approx 4.32$،
  - یا حدود ۵ بیت به ازای هر سطل (مقداری کمتر از ۳۲)
- تخمین تعداد منحصربه‌فرد تا حدود $10^{۹}$ با خطای استاندارد چهار درصد
  - نیاز الگوریتم به ۱۰۲۴ سطل ۵ بیتی
  - در مجموع ۶۴۰ بایت

روش ابرلگ‌لگ

فیلیپ فلاژوله، اریک فیوزی، اولیویه گاندوئه، و فردریک مونیه

- سال ۱۳۸۶
- پیشنهاد الگوریتم ابرلگ‌لگ بر اساس الگوریتم لگ‌لگ

استفاده از تابع درهم‌ساز ۳۲ بیتی، تابع ارزیابی متفاوت و تصحیحات سوگیری مختلف

مشابه الگوریتم لگ‌لگ،

- استفاده ابرلگ‌لگ از تصادفی‌سازی برای تقریب تعداد منحصربه‌فرد مجموعه داده
- طراحی برای مدیریت تعداد منحصربه‌فرد تا 10^9 با یک تابع درهم‌ساز ۳۲ بیتی
- تقسیم مجموعه داده به $m = 2^p$ زیرمجموعه، با دقت $p \in 4 \dots 16$

روش ابرلگ‌لگ

تمایز تابع ارزیابی الگوریتم ابرلگ‌لگ از لگ‌لگ استاندارد

استفاده الگوریتم لگ‌لگ اصلی از میانگین حسابی

استفاده ابرلگ‌لگ از نسخه نرمال شده میانگین هارمونیک

$$\hat{n} \approx \alpha_m \cdot m^2 \cdot \left(\sum_{j=0}^{m-1} 2^{-Coutner[j]} \right)^{-1}$$

که در آن

$$\alpha_m = \left(m \int_0^{\infty} \left(\log_2 \left(\frac{2+x}{1+x} \right) \right)^m dx \right)^{-1} = \frac{1}{2ln2 \left(1 + \frac{1}{m}(3ln2 - 1) + O(m^{-2}) \right)}$$

شهود استفاده از میانگین هارمونیک

- کاهش وردائی به دلیل ویژگی آن در هموار کردن توزیع‌های احتمال چوله

جدول- α_m برای پرکاربردترین مقادیر m

m	α_m
$۲^۴$	۰.۶۷۳
$۲^۵$	۰.۶۹۷
$۲^۶$	۰.۷۰۹
$\geq ۲^۷$	$\frac{۰.۷۲۱۳\times m}{m+۱.۰۷۹}$

روش ابرلگ‌لگ

نیاز به تصحیح تخمین به دلیل خطاهای غیرخطی برای محدوده‌های کوچک و بزرگ

فلاژوله و همکاران در نتیجهٔ آزمایش تجربی

- برای تعداد منحصربه‌فردهای کوچک $n < \frac{5}{2}m$ برای دستیابی به تخمین‌های بهتر،
- تصحیح الگوریتم ابرلگ‌لگ با شمارش خطی با استفاده از تعدادی از شمارنده‌های COUNTER[j] غیرصفر
- در صورت صفر بودن شمارنده، امکان نتیجه‌گیری خالی بودن آن زیرمجموعه خاص با قطعیت

روش ابرلگ‌لگ

تصحیحات

$$n = \begin{cases} LinearCounter, & \hat{n} \leq \frac{5}{2}m \bigwedge \exists j : Counter[j] \neq 0 \\ -2^{23}\log\left(1 - \frac{\hat{n}}{2^{32}}\right), & \hat{n} > \frac{1}{30}2^{32} \\ \hat{n}, & \text{دغا} \end{cases}$$

روش ابرلگ‌لگ

برای $n = 0$

- عدم کفایت تصحیح کافی و برگرداندن همیشه تقریباً $0.7m$

روش ابرلگ‌لگ

استفاده الگوریتم ابرلگ‌لگ از تابع درهم‌ساز ۳۲ بیتی

با نزدیک شدن تعداد منحصربه‌فرد به $۲^{۳۲} \approx ۴ \times ۱۰^{۹}$

رسیدن تابع درهم‌ساز تقریباً به حد خود و افزایش احتمال تصادم

برای چنین محدوده‌های بزرگ، تخمین تعداد مقادیر درهم مختلف

- استفاده از آن برای تقریب تعداد منحصربه‌فرد
- عدم امکان نمایش مقادیر بالاتر و تاثیر بر دقت

روش ابرلگ‌لگ

تابع درهم‌ساز با نگاشت دامنه به مقادیر ℓ بیت

امکان کدگذاری حداکثر 2^{ℓ} مقدار مختلف

در صورت نزدیک شدن تعداد منحصربه‌فرد تخمینی n به چنین حدی

- محتمل‌تر شدن تصادم‌های درهم‌ساز

روش ابرلگ‌لگ

هیچ نشانی دال بر عملکرد بهتر توابع درهم‌ساز پرکاربرد

- (مانند مورمور۳، MD5، SHA-1، SHA-256)
- نسبت به سایر روش‌ها در الگوریتم‌های ابرلگ‌لگ یا اصلاحات آن وجود ندارد.

روش ابرلگ‌لگ

الگوریتم ۹- تخمین تعداد منحصربه‌فرد با ابرلگ‌لگ

ورودی: مجموعه داده D

ورودی: آرایه‌ای از m شمارنده LogLog با تابع درهم‌ساز h

خروجی: تخمین تعداد منحصربه‌فرد

$COUNTER\,[j] \leftarrow ۰, j = ۰ \dots m - ۱$

برای $x \in D$ انجام بده

$$i \leftarrow h(x) = (i_{۳۱} \dots i_۱ i_۰)_۲$$

$$j \leftarrow (i_{p-۱} \dots i_۱ i_۰)_۲$$

$$r \leftarrow rank\left((i_{۳۱} \dots i_{p+۱} i_p)_۲\right)$$

$$COUNTER[j] \leftarrow max(COUNTER[j], r)$$

$$R \leftarrow \sum_{j=۰}^{m-۱} ۲^{-Counter[j]}$$

$$\hat{n} = \alpha_m \cdot m^۲ \cdot \frac{۱}{R}$$

$$n \leftarrow \hat{n}$$

اگر $\hat{n} \leq \frac{۵}{۲} m$ آنگاه

$$Z \leftarrow \text{تعداد}_{j=۰\dots m-۱}(COUNTER[j] = ۰)$$

اگر $Z \neq ۰$ آنگاه

$$n \leftarrow m \cdot log\left(\frac{m}{Z}\right)$$

در غیر این صورت اگر $\hat{n} > \frac{۱}{۳۰} ۲^{۳۲}$ آنگاه

$$n \leftarrow -۲^{۳۲} \cdot \log\left(۱ - \frac{n}{۲^{۳۲}}\right)$$

برگرداندن n

روش ابرلگ‌لگ

مشابه الگوریتم لگ‌لگ، بین خطای استاندارد δ و تعداد شمارنده‌ها m نیاز به سبک-سنگینی

$$\delta \approx \frac{1.04}{\sqrt{m}}$$

عدم رشد خطی حافظه با تعداد مقادیر

- برخلاف الگوریتم‌هائی مانند الگوریتم شمارش خطی
- با تخصیص (ℓ-p) بیت برای مقادیر درهم و داشتن $m = 2^p$ شمارنده در مجموع،
- حافظه مورد نیاز برابر با

$$\lceil \log_2(\ell + 1 - p) \rceil \cdot 2^p \; bits$$

استفاده الگوریتم از توابع درهم‌ساز ۳۲ بیتی و دقت $p \in 4 \dots 16$ استفاده می‌کند،

- نیازهای حافظه برای داده‌ساختار ابرلگ‌لگ برابر با $5 \cdot 2^p$ بیت

روش ابرلگ‌لگ

الگوریتم ابرلگ‌لگ
- تخمین تعداد منحصربه‌فردهای بسیار فراتر از 10^9
- با دقت دو درصد
- با استفاده از تنها ۱٫۵ کیلوبایت حافظه

پایگاه داده معروف درون‌حافظه Redis
- حفظ ساختارهای داده ابرلگ‌لگ با ۱۲ کیلوبایت
- تقریب تعداد منحصربه‌فردها با خطای استاندارد ۰.۸۱

بهبود تخمین تعداد منحصر به فرد در ابرلگ‌لگ در مقایسه با لگ‌لگ در مجموعه‌های داده کوچک
- با این وجود تخیمن بیش از حد تعداد منحصربه‌فردهای واقعی

پیاده سازی انواع الگوریتم‌های ابرلگ‌لگ در پایگاه‌های داده شناخته شده
- مانند Amazon Redshift، Redis، Apache CouchDB، Riak

روش ابرلگ‌لگ++

اشتفان هویله، مارک نونکسر، و اسکندر هال
- سال ۱۳۹۲
- معرفی نسخه بهبود یافته ابرلگ‌لگ را به نام ابرلگ‌لگ++،
- متمرکز بر روی تعداد منحصربه‌فردهای بزرگ و تصحیح سوگیری

استفاده از تابع درهم‌ساز ۶۴ بیتی
- اهم اصلاحات در الگوریتم ابرلگ‌لگ++

مقادیر خروجی طولانی‌تر تابع درهم‌ساز
- امکان کدگذاری مقادیر بیشتر

امکان تخمین تعداد منحصربه‌فردهای بسیار بزرگتر از 10^9

با نزدیک شدن تعداد منحصربه‌فرد به $2^{64} \approx 1.8 \times 10^{19}$
- مشکل‌سازی تصادم‌های درهم برای ابرلگ‌لگ++

روش ابرلگ‌لگ++

استفاده الگوریتم ابرلگ‌لگ++ از تابع ارزیابی لگ‌لگ

- بهبود تصحیح سوگیری

اجرای مجموعه‌ای از آزمایش‌ها برای اندازه‌گیری سوگیری

- نتایج حال از داده‌های تجربی جمع‌آوری شده
- برای $n \leq 5m$ امکان تصحیح سوگیری الگوریتم ابرلگ‌لگ اصلی

روش ابرلگ‌لگ++

علاوه بر مقاله اصلی، هویله و همکاران

ذخیره مقادیر تعیین شده تجربی

- در آرایه‌هایی از تخمین‌های خام تعداد منحصربه‌فرد
  - RAWESTIMATEDATA
- عرضهٔ سوگیری‌های مرتبط BIASDATA برای بهبود تصحیح سوگیری در الگوریتم
- سعی پوشش هر مورد ممکن
- RAWESTIMATEDATA عرضهٔ آرایه‌ای از ۲۰۰ نقطه درون‌یابی
  - حاوی میانگین تخمین خام اندازه‌گیری شده در نقاط در بیش از ۵۰۰۰ مجموعه داده مختلف
- BIASDATA
  - شامل حدود ۲۰۰ سوگیری اندازه‌گیری شده و مطابق با RAWESTIMATEDATA

روش ابرلگ‌لگ++

- هر دو آرایه صفر-اندیس
- حاوی مقادیر از پیش محاسبه شده برای همه دقت‌های پشتیبانی شده $p \in 4 \ldots 18$

- تطابق اندیس صفر در آرایه‌ها با مقدار دقت ۴
- مثال، برای $m = 2^{10}$ و $p = 10$
  - یافتن داده‌های مورد نیاز در $RAWESTIMATEDATA[6]$ و $BIASDATA[6]$

روش ابرلگ‌لگ-

الگوریتم ۱۰- تصحیح سوگیری با استفاده از مقادیر تجربی

ورودی: تخمین $\hat{n}$ با دقت p

خروجی: تخمین تعداد منحصربه‌فرد تصحیح شده با سوگیری

$n_{low} \leftarrow 0, n_{up} \leftarrow 0, j_{low} \leftarrow 0, j\_up \leftarrow 0$

برای $j \leftarrow 0$ تا طول(RAWESTIMATEDATA [p - ٤]) انجام بده

اگر $\hat{n} \geq RAWESTIMATEDATA\,[p - ۴][j]$ آنگاه

$j_{low} \leftarrow j - ۱, j_{up} \leftarrow j$

$n_{low} \leftarrow RAWESTIMATEDATA[p - ۴][j_{low}]$

$n_{up} \leftarrow RAWESTIMATEDATA[p - ۴][j_{up}]$

$break$

$b_{low} \leftarrow BIASDATA[p - ۴][j_{low}]$

$b_{up} \leftarrow BIASDATA[p - ۴][j_{up}]$

$y = \text{درون‌یابی}\left((n_{low}, n_{low} - b_{low}), (n_{up}, n_{up} - b_{up})\right)$

برگرداندن $y(\hat{n})$

روش ابرلگ‌لگ++ -- مثال

تصحیح سوگیری با استفاده از مقادیر تجربی

فرض: حاصل تخمین تعداد منحصربه‌فرد $\hat{n} = 2018.34$ از روش اصلی

در پی تصحیح آن برای دقت $p = 10$

- $(m = 2^{10})$

روش ابرلگ‌لگ++

بررسی آرایه $RAWESTIMATEDATA[6]$

- تعیین اندیس‌های دو سر بازه حاوی مقدار $\hat{n}$
- بین مقادیر با اندیس‌های ۷۳ و ۷۴ آرایه
- $RAWESTIMATEDATA[6][73] = 2003.1804$ و $RAWESTIMATEDATA[6][74] = 2026.071$.

$$2003.1804 \leq \hat{n} \leq 2026.071$$

بازیابی سوگیری‌ها از $BIASDATA[6]$ در موقعیت‌های ۷۳ و ۷۴

- $BIASDATA[6][73] = 134.1804$ و $BIASDATA[6][74] = 131.071$.

تخمین صحیح در بازه

- $[2003.1804 - 134.1804, 2026.071 - 131.071] = [1869.0, 1895.0]$

روش ابرلگ‌لگ++

درون‌یابی برای محاسبه تقریب تصحیح شده

- استفاده از جستجوی k-نزدیک‌ترین همسایه یا صرفا با درون‌یابی خطی $y(x) = ax + b$
- $y(2003.1804) = 1869.0$ و $y(2026.071) = 1895.0$.

حاصل درون‌یابی

- $y = 1.135837\,x - 406.28725$

مقدار درون‌یابی شده برای تخمین تعداد منحصربه‌فرد

$$n = y(\hat{n}) = y(2018.34) \approx 1886.218$$

روش ابرلگ‌لگ++

با توجه به آزمایش‌های نویسندگان $HyperLogLog++$،

- بهتر بودن تخمین n_{lin} بر اساس الگوریتم شمارش خطی در مقایسه با مقدار تصحیحی با سوگیری n برای تعداد منحصربه‌فردهای کوچک

در صورت وجود حداقل یک شمارنده خالی

- محاسبهٔ تخمین خطی و فهرستی از آستانه‌های تجربی (در جدول ادامه)

استفاده از مقدار تصحیح شده با سوگیری n

- تخمین خطی n_{lin} بالاتر از آستانه χ_m برای m فعلی

روش ابرلگ‌لگ++ -- مثال

تصحیح سوگیری با آستانه

$m = 2^{10}$ محاسبهٔ مقدار تصحیح شده با سوگیری $n \approx 1886.218$

جهت تعیین ترجیح این مقدار به تخمین شمارش خطی

- نیاز به یافتن تعداد شمارنده‌های خالی Z
- فرض $Z = 73$
- بنابراین، تخمین خطی $n_{lin} = 2^{10} \cdot \log\left(\frac{2^{10}}{73}\right) \approx 2704$

مقایسه n_{lin} با آستانه $\chi_m = 900$ از جدول

- بسیار کمتر از مقدار محاسبه شده
- ترجیح تخمین تصحیح شده با سوگیری n به تخمین شمارش خطی n_{lin}

روش ابرلگ‌لگ++

آستانه‌های تجربی χm برای مقادیر دقت پشتیبانی شده

p	m	χm	p	m	χm	p	m	χm
۴	$۲^{۴}$	۱۰	۹	$۲^{۹}$	۴۰۰	۱۴	$۲^{۱۴}$	۱۱۵۰۰
۵	$۲^{۵}$	۲۰	۱۰	$۲^{۱۰}$	۹۰۰	۱۵	$۲^{۱۵}$	۲۰۰۰۰
۶	$۲^{۶}$	۴۰	۱۱	$۲^{۱۱}$	۱۸۰۰	۱۶	$۲^{۱۶}$	۵۰۰۰۰
۷	$۲^{۷}$	۸۰	۱۲	$۲^{۱۲}$	۳۱۰۰	۱۷	$۲^{۱۷}$	۱۲۰۰۰۰
۸	$۲^{۸}$	۲۲۰	۱۳	$۲^{۱۳}$	۶۵۰۰	۱۸	$۲^{۱۸}$	۳۵۰۰۰۰

روش ابرلگ‌لگ++

الگوریتم ۱۱- تخمین تعداد منحصربه‌فرد با ابرلگ‌لگ++

ورودی: مجموعه داده D

ورودی: آرایه‌ای از m شمارنده لگ‌لگ با تابع درهم‌ساز h

خروجی: تخمین تعداد منحصربه‌فرد

$COUNTER\,[j] \leftarrow 0, j = 0 \dots m-1$

برای $x \in D$ انجام بده

$i \leftarrow h(x) = (i_{63} \dots i_1 i_0)_2$

$j \leftarrow (i_{p-1} \dots i_1 i_0)_2$

$r \leftarrow rank\left((i_{63} \dots i_{p+1} i_p)_2\right)$

$COUNTER[j] \leftarrow max(COUNTER[j], r)$

$R \leftarrow \sum_{k=0}^{m-1} 2^{-Counter[j]}$

$\hat{n} = \alpha_m \cdot m^2 \cdot \frac{1}{R}$

$n \leftarrow \hat{n}$

اگر $\hat{n} \leq 5m$ آنگاه

$n \leftarrow CorrectBias\,(\hat{n})$

$Z \leftarrow$ تعداد $_{j=0\dots m-1}\,(COUNTER\}[j] = 0)$

اگر $Z \neq 0$ آنگاه

$n_{lin} \leftarrow m \cdot \log \frac{m}{Z}$

اگر $n_{lin} \leq \chi_m$ آنگاه

$n \leftarrow n_{lin}$

برگرداندن n

الگوریتم ۹- تخمین تعداد منحصربه‌فرد با ابرلگ‌ل

ورودی: مجموعه داده D

ورودی: آرایه‌ای از m شمارنده LogLog با تابع

خروجی: تخمین تعداد منحصربه‌فرد

$OUNTER\,[j] \leftarrow 0, j = 0 \dots m-1$

برای $x \in D$ انجام بده

$i \leftarrow h(x) = (i_{31} \dots i_1 i_0)_2$

$j \leftarrow (i_{p-1} \dots i_1 i_0)_2$

$r \leftarrow rank\left((i_{31} \dots i_{p+1} i_p)_2\right)$

$TER[j] \leftarrow max(COUNTER[j], r)$

$R \leftarrow \sum_{j=0}^{m-1} 2^{-Counter[j]}$

$\hat{n} = \alpha_m \cdot m^2 \cdot \frac{1}{R}$

$n \leftarrow \hat{n}$

اگر $\hat{n} \leq \frac{5}{2}m$ آنگاه

تعداد $_{j=0\dots m-1}(COUNTER[j] = 0)$

اگر $Z \neq 0$ آنگاه

$n \leftarrow m \cdot \log\left(\frac{m}{Z}\right)$

در غیر این صورت اگر $\hat{n} > \frac{1}{30} 2^{32}$ آنگاه

$n \leftarrow -2^{32} \cdot \log\left(1 - \frac{n}{2^{32}}\right)$

برگرداندن n

روش ابرلگ‌لگ++

دقت بهتر یا متناسب ابرلگ‌لگ++ با ابرلگ‌لگ

تعداد منحصربه‌فردهای بین ۱۲۰۰۰ و ۶۱۰۰۰،

- امکان خطای کمتر با تصحیح سوگیری
- اجتناب از افزایش ناگهانی خطا هنگام جابجایی بین الگوریتم‌های فرعی

روش ابرلگ‌لگ++

عدم نیاز ابرلگ‌لگ به ذخیرهٔ مقادیر درهم،

- فقط یک به اضافه حداکثر اندازه تعداد صفرهای ابتدایی
- عدم افزایش قابل توجه حافظه موردنیاز در مقایسه با ابرلگ‌لگ
- با توجه به جدول، نیاز به $6 \cdot 2^p$ بیت

تخمین تعداد منحصربه‌فرد حدود $7.9 * 10^9$ با میزان خطای ۱.۶۲۵درصد

- با استفاده از ۲,۵۶ کیلوبایت حافظه

روش ابرلگ‌لگ++

استفاده الگوریتم از رویکرد میانگین‌گیری تصادفی

تقسیم مجموعه داده به $m = 2^p$ زیرمجموعه $\{S_j\}_{j=0}^{m-1}$

هر کدام دارای شمارنده‌های مرتبط $\{COUNTER[j]\}_{j=0}^{m-1}$

هر شمارنده مدیریت اطلاعات مربوط به $\frac{n}{m}$ مقدار

عدم استفاده از بیشتر شمارنده‌ها وقتی $n \ll m$

- عدم نیاز به ذخیره‌سازی
- امکان استفاده از نمایش تنک
- درصورت کوچک‌تر بودن تعداد منحصربه‌فرد n از m
  - نیاز به حافظه بسیار کمتر ابرلگ‌لگ++ نسبت به روش‌های پیشین

الگوریتم ابرلگ‌لگ++ نسخه تنک

صرفا ذخیرهٔ جفت‌های $(j, COUNTER[j])$

نمایش آنها با الحاق الگوهای بیتی خود به عنوان یک عدد صحیح منفرد

ذخیره همه جفت‌ها در فهرست مرتب شده منفرد از اعداد صحیح

همیشه در پی محاسبهٔ حداکثر رتبه، عدم نیاز به ذخیره جفت‌های مختلف با اندیس یکسان

- در عوض ذخیره جفتی با حداکثر اندیس

در عمل، برای ارائه تجربه بهتر، امکان ایجاد فهرست مرتب نشده دیگر برای درج‌های سریع

مرتب‌سازی دوره‌ای و ادغام در فهرست اصلی

ذیل!

سوراو چاکرابورتی، وینودچاندران، کولدیپ میل

- ۱۴۰۱

معرفی الگوریتمی برای مسئلهٔ زیر

- ورود یک به یک دنباله‌ای از مقادیر $(a_1, a_2, \ldots, a_m)$
- به دنبال یافتن تعداد متمایز آنها
- $A = \{a_1, a_2, \ldots, a_m\}$ مجموعه عناصر موجود در این دنباله
  - با چشمپوشی از تکرارها
- به دنبال اندازه مجموعه یعنی $|A|$
- $|A|$ احتمالاً بسیار بزرگتر از تعداد مقادیر قابل‌نگهداری در حافظه در هر لحظه

راهبرد مناسب محاسبهٔ تخمین ناریب از $|A|$

ذیل

الگوریتمی جالب و در عین حال بسیار ساده

به پیشنهاد کنوث در سال ۱۴۰۲ معروف به الگوریتم CVM

الگوریتم دارای بافری با اندازهٔ محدود

- جهت حفظ زیرمجموعه‌ای تصادفی از آن مقادیر
- $\Pr[(1-\epsilon).|A| \leq c \leq (1+\epsilon).|A|] \geq 1-\delta$

مقدار ورودی	h(x) (عدد صحیح 0–255)
A	120
B	200
C	45
D	200
E	12
F	240
G	78

A, B, C, D, A, E, B, C, F, G

{A, B, C, D, E, F, G} و n=7

روش‌های شمارش خطی، PCSA، بهبود شویرمان، لگ‌لگ و ابرلگ‌لگ

شمارش خطی با m=16

PCSA و بقیه موارد با چهار شمارنده

میزان خطای هر یک را حساب کنید

مقدار ورودی	h(x) (عدد صحیح 0–255)
A	120
B	200
C	45
D	200 (تصادم با B)
E	12
F	240
G	78

A, B, C, D, A, E, B, C, F, G

{A, B, C, D, E, F, G} و n=7

روش‌های شمارش خطی، PCSA، بهبود شویرمان، لگ‌لگ و ابرلگ‌لگ

شمارش خطی با m=16

PCSA و بقیه موارد با چهار شمارنده

میزان خطای هر یک را حساب کنید

مقدار ورودی	h(x) (عدد صحیح 0–255)	h(x) (دو مقداری هشت بیتی)
A	120	01111000
B	200	11001000
C	45	00101101
D	200 (تصادم با B)	11001000
E	12	00001100
F	240	11110000
G	78	01001110

A, B, C, D, A, E, B, C, F, G

{A, B, C, D, E, F, G} و n=7

روش‌های شمارش خطی، PCSA، بهبود شویرمان، لگ‌لگ و ابرلگ‌لگ

شمارش خطی با m=16

PCSA و بقیه موارد با چهار شمارنده

میزان خطای هر یک را حساب کنید

```python
class CVM:
    def __init__(self, epsilon, delta):
        self.s = ceil((24 / epsilon**2) * log(8 / delta))
        self.buffer = set()
        self.threshold = 1.0

    def add(self, x):
        self.buffer.add(x)

        while len(self.buffer) > self.s:
            # Purge each element with probability 0.5
            survivors = {e for e in self.buffer
                         if random.random() < 0.5}
            self.buffer = survivors
            self.threshold *= 2

    def estimate(self):
        return len(self.buffer) * self.threshold
```