الگوریتم
بسامد و تکرار

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

بسامد و تکرار

تخمین بسامد مقادیر

- از جمله تعیین پربسامدترین مقدار یا تشخیص مقادیر روند در بازه زمانی

مورد نیاز در بسیاری از مسائل مهم در برنامه‌های جریانی و جریان‌های داده بزرگ

در صورت بزرگ بودن جریان‌های داده

- یا دنباله بی‌نهایت از مقادیر
- تعداد مقادیر متمایز زیاد
- امکان‌پذیر نبودن راه‌حل‌های معمول، مانند مرتب‌سازی یا نگه داشتن شمارنده برای هر مقدار
- امکان‌پذیر نبودن ذخیره و پردازش مجدد چنین دنباله‌هائی در بیشتر موارد
- نیاز به الگوریتم‌های جریان داده تک‌گذره

بسامد و تکرار

جریان داده بزرگ با تعداد منحصربه‌فرد پایین

- صرفا شامل تعداد کمی از مقادیر متمایز
- کفایت حفظ شمارنده‌های دقیق بسامد با استفاده از یک شمارنده به ازای هر مقدار متمایز
- عدم نیاز به الگوریتم‌های خاص

بسامد و تکرار

ویژگی خاص برنامه‌های داده بزرگ با وظیفهٔ مدیریت جریان‌های داده بزرگ

- نیاز با ساختارهای داده و الگوریتم‌ها
- الف- اعمال بر یک گذر از داده‌ها
- ب- فضای زیرخطی (حداکثر چندلگاریتمی)، یا عدم رشد سریع با اندازه جریان ورودی
- ج-پشتیبانی از به‌روزرسانی‌های سریع و ساده با تضمینی از دقت

بسامد و تکرار

به دلیل محدودیت‌های فضا

ساختارهای فشرده داده

خلاصه‌ای از جریان داده (به عنوان مثال، طرح فشرده)

غیرممکنی محاسبه دقیق بیشتر توابع روی جریان

نیاز به تقریب احتمالاتی

بسامد و تکرار

جریان داده $D = \{x_1, x_2, \dots, x_n\}$

- دنباله‌ای از مقادیر با هر ماهیتی
- با فرض اندازه بسیار بزرگ تعداد مقادیر n مثلا میلیاردها، و وجود تعداد زیادی ناشناخته از مقادیر متمایز
- در صورت بی‌نهایت بودن جریان
- D در یک پنجره زمانی به عنوان زیرجریان

با رویکردی برای تخمین بسامد مقادیر در جریان داده عظیم

- امکان بررسی مسئلهٔ رایج یافتن فهرست مقادیر با بسامد بالا در جریان
- مسئله پربسامد

بسامد و تکرار

به دنبال مقداری با وقوع بیش از $\frac{n}{2}$ دفعه در جریان داده D

- مسئله اکثریت
- طرح آن توسط جی. استروتر مور در مجله الگوریتم‌ها در سال ۱۳۶۰ به عنوان مسئله‌ای تحقیقاتی
- فرض بر وجود چنین مقداری در جریان
- همیشه صادق نیست،
- منحصر به فرد بودن آن در صورت وجود

مناسب برای درک بهتری از مسائل بسامد مربوط به جریان‌های داده

مسئله یافتن k مقدار پربسامد در جریان

- از مسائل پیچیده در هنگام کار با داده بزرگ
- با رخداد بیش از $\frac{n}{k}$ بار
- همچنین معروف به مقادیر پربسامد
- $k \ll n$
- معمولا ۱۰، ۱۰۰ یا ۱۰۰۰

بسامد و تکرار

فرض جستجو برای مقادیر پربسامد

- وقوع بسیار بیشتر برخی از مقادیر نسبت به سایرین در جریان داده
- در غیر این صورت حل این مسئله معنا ندارد.

اثبات عدم وجود الگوریتم دقیق محاسبهٔ مقادیر پربسامد در یک گذر با استفاده از فضای زیرخطی

- با استفاده از پیچیدگی ارتباط

ارتباط بسیاری از کاربردهای عملی با مسئله مقادیر پربسامد

- از جمله جستجو، استخراج لاگ، تحلیل شبکه، مهندسی ترافیک و تشخیص ناهنجاری

مثال، تعیین پرکارترین k کاربر (برای مقدار دلخواه $k \ll n$)

- برای تارمانهٔ با ترافیک بالا
- امکان یکسانی بار چند تن از کاربران
- غیرممکنی دریافت پاسخ دقیق به این سوال با استفاده از فضای محدود

بسامد و تکرار

در عمل، مسئله ϵ-مقادیر پربسامد

- ϵ-تقریب از مسئله مقادیر پربسامد
- مقادیری با حداقل وقوع $\frac{n}{k}$ دفعه
- با وقوع تضمین شده حداقل $\frac{n}{k} - \epsilon \cdot n$
  - $\epsilon > 0$ و کوچک
- مثال، $\epsilon = \frac{1}{2k} > 0$ خروجی مسئله ϵ-مقادیر پربسامد عناصری با بسامد حداقل $\frac{n}{k}$
- تضمین وقوع حداقل $\frac{n}{2k} = \frac{n}{k} - \frac{n}{2k}$ دفعه $\frac{n}{k} - \epsilon \cdot n$

برای جریان داده‌های کوچک

- بدون توجه به تعداد مقادیر منحصربه‌فرد
- کفایت مرتب‌سازی مقادیر و پویش خطی،
- یافتن مقادیر با وقوع حداقل $\frac{n}{k}$ دفعه
  - مقادیر پربسامد

بسامد و تکرار

در جریان داده دلخواه D

- وجود از صفر تا k مقدار پربسامد
- احتمال فراوان وجود چند مقدار پربسامد
- بدون وجود مقدار اکثریت

مسئله اکثریت حالتی خاص از مسئله مقادیر پربسامد

- با شرط وجود چنین مقدار اکثریتی
- $k \approx 2 - \epsilon$،

مثال-تشخیص حمله

حمله انکار سرویس توزیع شده (DDoS)

تعداد فراوانی از سیستم‌ها

جهت اشغال منابع سیستم هدف با ارسال تعداد زیادی پرس و جو از بات‌نت سیل‌آسا

سیستم نام دامنه (DNS)

- از اهداف معمول
- نقش «دفترچه تلفن» اینترنت
- فراهم‌ساز ترجمه بین نام‌های دامنه به یاد ماندنی و آدرس‌های IP تارمانه‌ها

مثال-تشخیص حمله

پرس و جوهای DNS

- به عنوان جریان داده
- هر مقدار دارای یک دامنه مرتبط
- امکان گروه‌بندی پرس و جوها با استفاده از دامنه سطح بالای آنها
- بررسی پرکارترین دامنه‌ها در جریان پرس و جو
- امکان تشخیص سیل DNS تصادفی هنگام صدور پرس و جوها برای بسیاری از زیردامنه‌های مختلف غیر موجود از همان دامنه اصلی

بسامد و تکرار

مسئله «حداکثر تغییر»

- در برنامه‌های جریانی،
- تعیین مقادیری با بیشترین تغییر بسامد در جریان‌های داده مختلف یا پنجره‌های زمانی
- اهمیت در موتورهای جستجو
- موضوعات با بیشترین تغییر بسامد بین دو دوره زمانی متوالی
- نشان‌دهندهٔ افزایش یا کاهش محبوبیت موضوعات با سریعترین سرعت

مثال-هشتگ‌های روند توییتر

استفاده از هشتگ برای نمایه‌سازی موضوع در توییتر

امکان به افراد در پیگیری راحت موارد مورد علاقه

- معمولا با نماد #

تولید حدود ۶۰۰۰ توییت در هر ثانیه در توییتر

منجر به تولید تقریبا ۵۰۰ میلیارد پیام در روز

مرتبط بودن اکثر توییت‌ها با یک یا چند هشتگ

برای آگاهی از همه رویدادهای اخیر،

- اهمیت تعیین محبوب‌ترین موضوعات روز

با پردازش جریان داده توییت‌ها، تخمین بسامد هر هشتگ و یافتن مقادیر پربسامد

احتمال نیز به بررسی و مقایسه بسامدهای دیروز و امروز جهت تعیین موضوعات در حال روند

- به عنوان مثال، موضوعاتی با بیشترین افزایش بسامد از دیروز

بسامد و تکرار

در ادامه،

رویکردهای مختلفِ حل مسائل مربوط به بسامد در جریان‌های داده بزرگ

آغاز با الگوریتم‌های قطعی بسیار ساده

سپس، روش‌های احتمالاتی متأخر

- حل مسائل دنیای واقعی با کارائی بالا

الگوریتم اکثریت

وجود راه‌حل زمان خطی مسئله اکثریت

مقدار اکثریت (البته با فرض وجود آن) برابر میانه

نقطه ضعف نیاز به چندین گذر از جریان

نامناسب برای جریان داده‌های بزرگ

الگوریتم اکثریت

الگوریتم اکثریت،

- همچنین مشهور به الگوریتم رأی اکثریت بویر-مور
- پیشنهاد باب بویر و جی. استروتر مور در سال ۱۳۶۰
- برای حل مسئله اکثریت در یک گذر از جریان داده
- پیشنهاد راه‌حل مشابه به طور مستقل توسط مایکل جی. فیشر و استیون ال. سالزبرگ در سال ۱۳۶۱

داده‌ساختار الگوریتم اکثریت

- نسبتا ساده
- جفت متشکل از شمارنده عدد صحیح و به اصطلاح مقدار تحت نظر (c,x^*) S.
- نیاز به مقدار ثابتی از حافظه، اما اندازه متفاوت آن بسته به اندازه مقادیر

الگوریتم اکثریت

پشتیبانی داده‌ساختار از عملیات به‌روزرسانی ساده

به‌روزرسانی شمارنده بر اساس وضعیت قبلی خود و مقدار فعلی x

انتخاب نامزد مقدار تحت نظر

الگوریتم ۱- به‌روزرسانی داده‌ساختار اکثریت

ورودی: مقدار $x \in D$

اگر $c = ۰$ آنگاه

$x^* \leftarrow x$

اگر $x = x^*$ آنگاه

$c \leftarrow c + ۱$

در غیر این صورت

$c \leftarrow c - ۱$

الگوریتم اکثریت

با این نوع داده‌ساختار، سادگی توصیف الگوریتم

برای هر مقدار x در جریان D

فعال‌سازی روش به‌روزرسانی الگوریتم ۱

- با شرط وجود مقدار اکثریت

برگرداندن آخرین مقدار تحت نظر به عنوان مقدار اکثریت

یکی نبودن مقدار شمارنده بسامد با مقدار اکثریت

الگوریتم ۲- الگوریتم اکثریت

ورودی: جریان داده D

خروجی: مقدار اکثریت

$c \leftarrow ۰$

$x^* \leftarrow \phi$

برای $x \in D$ انجام بده

$Update(x)$

برگرداندن x^*

الگوریتم اکثریت

در الگوریتم اکثریت

هر مقدار «غیراکثریت» که به دنبال می‌آید کاهندهٔ شمارنده **c** یا بازنشانی آن به صفر

منجر به انتخاب دوبارهٔ مقدار تحت نظر x^*

از دیدگاه غیردقیق،

- امکان ابهام در چگونگی پاسخ صحیح دادن چنین الگوریتمی
- ابهام در وجود خطر حذف مقادیر اکثریت در بعضی از جریان داده‌ها.

الگوریتم اکثریت

مقادیر «غیراکثریت» قادر به حذف یک نسخه از مقدار اکثریت قبلی

به دلیل تکرار بیش از $\frac{n}{2}$ از مقدار اکثریت در جریان داده

عدم وجود مقادیر «غیراکثریت» کافی برای حذف همه

برقرار ماندن حداقل یک نسخه از مقدار اکثریت در پایان

توضیح‌دهندهٔ برابر نبودن مقدار برگشتی شمارنده به عنوان تقریبی از بسامد مقدار اکثریت

الگوریتم اکثریت

در صورت عدم وجود مقدار اکثریت

- خروجی الگوریتم اکثریت مقداری دلخواه از جریان داده

اعمال چنین الگوریتمی در مواقع عدم اطمینان به وجود مقدار اکثریت

- نیاز به گذر دیگری از جریان داده با یک شمارنده ساده
- جهت تایید اکثریت بودن خروجی الگوریتم ۲ با وقوع بیش از $\frac{n}{۲}$ دفعه

مثال- الگوریتم اکثریت

مجموعهٔ داده با $n = 10$ مقدار

$\{4,4,3,5,6,4,4,4,4,2\}$،

مقدار اکثریت به وضوح $x = 4$

شش بار از ده بار

مثال- الگوریتم اکثریت

طبق الگوریتم،

- تخصیص جفت $S = (c,x^*) = (0,\phi)$
- خواندن مقادیر از مجموعه داده
- اولین مقدار $x_1 = 4$
  - خالی بودن شمارنده c
  - ذخیرهٔ آن به عنوان مقدار تحت نظر $x^* = 4$
  - افزایش شمارنده $c = 1$
- مقدار بعدی x_2 دوباره ۴
  - برابری با مقدار تحت نظر
  - صرفا افزایش شمارنده $c = 2$
- سومین مقدار ورودی $x_3 = 3$
  - متفاوت با $x^* = 4$
  - کاهش شمارنده $c = 1$
- پردازش $x_4 = 5$
  - کاهش دوباره شمارنده $c = 0$.

مثال- الگوریتم اکثریت

پردازش مقدار $x_5 = 6$
- مقدار شمارنده فعلی صفر
- به‌روزرسانی مقدار تحت نظر به $x^* = 6$
- تنظیم شمارنده آن $c = 1$

پس از مدیریت مقادیر $x_6 = 4$ و $x_7 = 4$
- مقدار تحت نظر دوباره $x^* = 4$ با شمارنده $c = 1$

مقدار بعدی افزایش شمارنده به $c = 2$

برابر نبودن آخرین مقدار با ۴
- کاهش شمارنده به مقدار $c = 1$

در پایان، مقدار اکثریت صحیح ۴ به عنوان مقدار تحت نظر
- شمارنده باقی‌مانده c تخمین‌گر بسامد نیست و حاوی مقدار کاملا متفاوت

الگوریتم پربسامد

اریک دی. دمین، الخاندرو لوپز-اورتیز و جی. یان مونرو در سال 1381

- معرفی الگوریتم پربسامد به عنوان تعمیمی از الگوریتم اکثریت
- بعدها کاشف به عمل آمد! مشابه الگوریتم پیشنهادی جایادیو میسرا و دیوید گریس در سال 1361
- اکنون مشهور به الگوریتم «میسرا-گریس»
- گریس متولد ۱۳۱۸
  - اولین کتاب درباره کامپایلر در ۱۳۵۰
  - کتاب علم برنامه‌نویسی
- میسرا متولد ۱۳۲۶
  - برنامه‌های همروند

الگوریتم پربسامد

برای رسیدگی به مسئله مقادیر پربسامد

به جای نگه داشتن تنها یک شمارنده مانند الگوریتم اکثریت

- نگه‌داری مجموعه‌ای از مقادیر تحت نظر X^* و آرایه‌ای از p شمارنده $C = \{c_i\}_{i=1}^{p}$

الگوریتم پربسامد

استفاده میسرا و گریس از درخت‌های جستجوی متوازن برای نمایش داده‌ساختار بسامد

استفاده محققان بعدی از جدول‌های درهم

- پیاده‌سازی آن در قالب لغت‌نامه

الگوریتم پربسامد

در پردازش مقداری جدید از جریان داده،

ابتدا بررسی وجود قبلی آن

در صورت جدید بودن مقدار ورودی
- افزودن آن به X^*
- به شرط پر نبودن فضا
- به دلیل حفظ حداقل p مقدار

در صورت افزوده نشدن
- همچنان به دنبال حضور آن یا تاثیر حضور آن در جریان
  - با کاهش شمارنده‌های همه مقادیر در مجموعه مقادیر تحت نظر

در صورت وجود مقدار از قبل در X^*
- افزایش شمارنده متناظر

حذف تمامی مقادیری با شمارنده برابر صفر در پایان روش

الگوریتم پربساد

الگوریتم۳- به‌روزرسانی داده‌ساختار پربسامد

ورودی: مقدار $x \in D$

ورودی: داده‌ساختار پربسامد با p شمارنده

اگر $x \notin X^*$ آنگاه

اگر $\exists\, m: c_m = ۰$ آنگاه

$x_m^* \leftarrow x$

اگر $x \in X^*$ آنگاه

$\exists\, m: x_m^* = x$

$c_m \leftarrow c_m + ۱$

در غیر این صورت

برای $j \leftarrow ۱$ تا p انجام بده

اگر $c_j > ۰$ آنگاه

$c_j \leftarrow c_j - ۱$

برای $j \leftarrow ۱$ تا p انجام بده

اگر $c_j = ۰$ آنگاه

$X^* \leftarrow X^* \setminus \{x_j^*\}$

الگوریتم پربسامد

استفاده الگوریتم پربسامد از داده‌ساختار بسامد به طول p

برای کشف مقادیری با وقوع حداقل به اندازهٔ $\frac{n}{p+1}$ دفعه در جریان داده به طول n

جهت تعیین حداکثر $k - 1$ مقدار پربسامد با وقوع حداقل $\frac{n}{k}$ دفعه در جریان داده

- نیاز به استفاده از $p = k - 1$ شمارنده

الگوریتم اکثریت حالت خاص الگوریتم میسرا-گریس

- k=2

الگوریتم پربسامد

الگوریتم ۴- الگوریتم پربسامد

ورودی: جریان داده D

ورودی: داده‌ساختار پربسامد با $k - ۱$ شمارنده

خروجی: مقادیر پربسامد

$C = \{c_i\}_{i=۱}^{k-۱}, c_i \leftarrow ۰$

$X^* \leftarrow \emptyset$

برای $x \in D$ انجام بده

Update(x)

برگرداندن X^*

الگوریتم پربسامد

شهود الگوریتم پربسامد بسیار شبیه به الگوریتم اکثریت

- با توجه به شرط وقوع مقادیر پربسامد بیش از $\frac{n}{k}$ دفعه

مثال -یافتن مقادیر پربسامد

جریان داده با $n = 18$ مقدار

$\{4,4,4,4,6,2,3,5,4,4,3,3,4,2,3,3,3,2\}$

برای شناسایی مقادیر پربسامدی که تعداد وقوع حداقل $\frac{n}{3} = 6$ دفعه

تخصیص $p = 2$ شمارنده

شناسایی حداکثر دو مقدار از سه مقدار پربسامد احتمالی

	1	2
X*		
C	0	0

مثال -یافتن مقادیر پربسامد

با مقدار ۴

- نبود X^* و نبود مقداری در داده‌ساختار
- افزودن مقدار ۴ به مجموعه مقادیر تحت نظر و افزایش شمارنده مرتبط $c_1 = 1$

	1	2
X*	4	
C	1	0

مثال -یافتن مقادیر پربسامد

پردازش سه مقدار بعدی برابر با ۴

- بودن مقدار در X^* از قبل
- افزایش شمارنده c_1 متناظر

	1	2
X^*	4	
C	4	0

مثال -یافتن مقادیر پربسامد

مقدار بعدی ۶

- نبود در مقدار تحت نظر
- وجود فضای خالی در مجموعه X^*
- درج مقدار ۶ در آن
- افزایش شمارنده $c_2 = 1$

	1	2
X^*	4	6
C	4	1

مثال -یافتن مقادیر پربسامد

مقدار ۲

- نبود در X^*
- نبود فضا در مجموعه و عدم امکان افزودن آن نمی‌توانیم آن را اضافه کنیم
- کاهش شمارنده‌های همه مقادیر موجود در X^*
- حذف مقادیر با شمارنده‌هایی متناظر برابر صفر رسیده از مجموعه تحت نظر
- پس حذف مقدار ۶ از مجموعه

	1	2
X^*	4	
C	3	0

مثال -یافتن مقادیر پربسامد

مقدار ۳ از جریان داده

- نبود مقدار در X^*
- داشتن فضا در مجموعه
- افزودن مقدار
- افزایش شمارنده مرتبط $c_2 = 1$

	1	2
X*	4	3
C	3	1

مثال -یافتن مقادیر پربسامد

ادامه به روشی مشابه،

- پردازش همه مقادیر باقی‌مانده
- داده‌ساختار نهایی

	1	2
X*	4	3
C	3	3

مقادیر پربسامد شناسایی شده ۴ و ۳

منعکس نشدن بسامدهای واقعی مقادیر با شمارنده‌ها

ویژگی‌ها

هزینه زمانی الگوریتم

- عملیات دیکشنری $O(1)$ در هر به‌روزرسانی و هزینه کاهش شمارنده‌ها
- بهینه‌سازی سرعت الگوریتم،
  - می‌توان همه شمارنده‌ها را یکباره و در زمان ثابت با سازماندهی آنها به ترتیب مرتب شده و استفاده از کدگذاری تفاضلی تفاوت کاهش داد
  - ، جایی که تنها اطلاعات ذخیره شده در مورد این است که یک شمارنده خاص چقدر بزرگتر از شمارنده کوچک بعدی است.
- کمینه کردن حرکات قابل توجه در ترتیب هنگام افزایش و کاهش شمارنده
  - به معنای گروه‌بندی همه شمارنده‌های مساوی
- عدم نیاز هر شمارنده به ذخیره یک مقدار
  - بلکه ذخیرهٔ گروه خود
- امکان تقویت الگوریتم پربسامد با زمان اجرا $O(1)$

ویژگی‌ها

نیاز به گذر دوم از جریان داده جهت بدست آوردن بسامد هر مقدار پرتکرار

- غیرقابل اجرا در مدیریت جریان‌های داده عظیم

مطالعه راه‌حل‌های مسائل مربوط به بسامد با ساختارهای داده احتمالاتی بسیار مؤثر

- مناسب برای جریان‌های داده بزرگ

طرح کلی شمارش COUNT SKETCH

طرح کلی شمارش

- الگوریتمی فضا-کارآمد
- برای حل بسیاری از مسائل جریان داده مربوط به بسامد

پیشنهاد موسی چاریکار، کوین چن و مارتین فاراچ-کلتون در سال ۱۳۸۱

نیاز به داده‌ساختاری فضا-کارآمد

- قادر به شمارش تقریبی مقادیر با بسامد بالا در جریان داده

طرح کلی شمارش COUNT SKETCH

درک بهتر مسئلهٔ حل شده با طرح کلی شمارش

- امکان استفاده از فیلتر بلوم شمارنده
- برای محاسبه بسامد مقادیر در جریان داده
- عدم دقت کافی برای ساختن تخمین‌گرهای بسامد دقیق

داده‌ساختاری با آرایهٔ $C = \{c^i\}_{i=1}^{m}$ از m شمارنده

- p تابع درهم‌ساز $h_1, h_2, \dots, h_p$

نگاشت مقادیر به $\{1, 2, \dots, m\}$

نمایه‌سازی مقدار x از جریان داده

- مانند فیلتر بلوم شمارنده
- شامل محاسبه $\{h_j(x)\}_{j=1}^{p}$
- افزایش شمارنده‌های متناظر $j = 1 \dots p$ در آرایه $c^{h_j(x)}, j$

طرح کلی شمارش

نیاز به یافتن بسامد $f(x)$ مقدار x

- محاسبهٔ مقادیر هر تابع درهم‌ساز برای آن
- محاسبهٔ مقادیر شمارنده‌های متناظر $c^1, c^2, \ldots, c^m$
  - نقش تخمین‌های بسامد را بازی می‌کنند.

به دلیل خاصیت غیرکاهشی شمارنده‌ها و استفاده از آرایه‌های یکسان توابع درهم‌ساز

- تخمین‌هایی بزرگتر از بسامد واقعی $f(x)$

$$f(x) \leq c^i, i = 1, \cdots, m$$

نابرابری حاصل تصادم‌های محتمل درهم‌ساز

- حین به‌روزرسانی مقادیر شمارنده‌ها
- «خطای یک‌طرفه» رایج در تخمین‌ها منجر به تبدیل همه مقادیر به تخمین‌های کران بالا

طرح کلی شمارش

طرح کلی شمارش

- بر اساس تخمین کران‌های پایین و بالا
- جهت اجتناب از مواقعی که تصادم مقادیر بسامد بالا منجر به تخریب تخمین مقادیر بسامد پایین
  - نیاز به انتخابی تصادفی جهت کاهش یا افزایش شمارنده
- به منظور کاهش وردائی،
  - استفاده از میانه تخمین‌ها

طرح کلی شمارش-داده‌ساختار

داده‌ساختاری برای ذخیره بسامدهای m مقدار با بسامد بالا

- دو بخش

متشکل از آرایه $m \times p$ از شمارنده‌های $\{c_j^i\}$

- یا آرایه‌ای از p جدول درهم‌ساز،
  - هر کدام دارای m سطل

طرح کلی شمارش-داده‌ساختار

استفاده از p تابع درهم‌ساز $h_1, h_2, \ldots, h_p$

- نگاشت مقادیر به $\{1, 2, \ldots, m\}$

p تابع درهم‌ساز $s_1, s_2, \ldots, s_p$ با نگاشت مقادیر به $\{+1, -1\}$

جهت پشتیبانی از تقریب دو طرفه نزدیک به مقدار بسامد واقعی

- فرض استقلال یک به یک توابع درهم‌ساز چه h_i و چه s_i و همچنین مستقل از یکدیگر

طرح کلی شمارش-داده‌ساختار

امکان به‌روزرسانی شمارنده‌ها برای هر مقدار نمایه شده

در نتیجه: تخمین تعداد دفعاتی مشاهدهٔ مقدار در گذشته

استفاده به عنوان تخمین بسامد مقدار

در هر زمان با نمایه‌سازی مقدار جدید x

- امکان کاهش یا افزایش شمارنده‌های $c_j^{h_j(x)}$ برای هر ردیف j از طرح فشرده
  - تعیین کاهش یا افزایش بر اساس مقدار خروجی $s_j(x)$
- امکان تخمین بیش از و یا کمتر از حد بسامد مقدار x در شمارنده‌ها

طرح کلی شمارش

الگوریتم ۵- به‌روزرسانی طرح کلی شمارش

ورودی: مقدار $x \in D$

ورودی: طرح کلی شمارش با شمارنده‌های $p \times m$

برای $j \leftarrow$ ۱ تا p انجام بده

$$i \leftarrow h_j(x)$$

$$c_j^i \leftarrow c_j^i + s_j(x)$$

طرح کلی شمارش

با فرض زمان ثابت محاسبهٔ هر تابع درهم‌ساز $\{h_j\}_{j=1}^{p}$ و $\{s_j\}_{j=1}^{p}$

- زمان اجرای روش به‌روزرسانی الگوریتم ۵ برابر O(p)

مثال- ساختن طرح کلی شمارش

مجموعه داده‌ای با $n = 18$ مقدار

$$\{4,4,4,4,2,3,5,4,6,4,3,3,4,2,3,3,3,2\}$$

داده‌ساختار

- استفاده از $m = 5$ شمارنده
- استفاده از $p = 3$ تابع درهم‌ساز مبتنی بر مورمور۳ و ف‌ن‌و۱الف و ام‌دی۵
  - برای تصمیم‌گیری انتخاب شمارنده به‌روزرسانی

$$h_1(x) = MurmurHash3(x)\%\ 5\ +\ 1,$$
$$h_2(x) = FNV1a(x)\%\ 5\ +\ 1,$$
$$h_3(x) = MD5(x)\%\ 5\ +\ 1,$$

مثال- ساختن طرح کلی شمارش

سه تابع درهم‌ساز برای تعیین جهت به‌روزرسانی:

$$s_1(x) = MurmurHash3(x)\%\ 2\ ?\ -1 :\ 1,$$
$$s_2(x) = FNV1a(x)\%\ 2\ ?\ -1 :\ 1,$$
$$s_3(x) = MD5(x)\%\ 2\ ?\ -1 :\ 1.$$

مثال- ساختن طرح کلی شمارش

در ابتدا، داده‌ساختار متشکل از مقادیر صفر

	1	2	3	4	5
h_1	0	0	0	0	0
h_2	0	0	0	0	0
h_3	0	0	0	0	0

مثال- ساختن طرح کلی شمارش

اولین مقدار جهت پردازش مقادیر از مجموعه داده

- مقدار ۴
- محاسبهٔ مقادیر درهم آن $h_1(4)$، $h_2(4)$، و $h_3(4)$
- جهت تعیین شمارنده‌هایی به‌روزرسانی شونده

$$i_1 = h_1(4) = MurmurHash3(4)\%\ 5 + 1 = 3,$$
$$i_2 = h_2(4) = FNV1a(4)\%\ 5 + 1 = 3,$$
$$i_3 = h_3(4) = MD5(4)\%\ 5 + 1 = 1.$$

مثال- ساختن طرح کلی شمارش

دو تابع درهم‌ساز تولید مقداری یکسان

- اما نگه‌داری فهرست‌های اختصاصی شمارنده‌ها برای هر تابع درهم‌ساز
- عدم ایجاد مشکل

تعیین جهت به‌روزرسانی،

- محاسبهٔ مقادیر درهم‌ساز $s_1(4)$، $s_2(4)$ و $s_3(4)$

$$s_1(4) = MurmurHash3(4)\%\ 2\ ?\ -1 :\ 1 = 1$$
$$s_2(4) = FNV1a(4)\%\ 2\ ?\ -1 :\ 1 = 1,$$
$$s_3(4) = MD5(4)\%\ 2\ ?\ -1 :\ 1 = -1$$

مثال- ساختن طرح کلی شمارش

بنابراین،

- افزایش شمارنده‌های c_1^3 و c_2^3
- کاهش شمارنده c_3^1

حاصل:

	1	2	3	4	5
h_1	0	0	1	0	0
h_2	0	0	1	0	0
h_3	-1	0	0	0	0

مثال- ساختن طرح کلی شمارش

سه مقدار بعدی نیز ۴

بنابراین

- افزایش یا کاهش سه بارهٔ همان شمارنده‌ها

	1	2	3	4	5
h_1	0	0	4	0	0
h_2	0	0	4	0	0
h_3	-4	0	0	0	0

مثال- ساختن طرح کلی شمارش

مقدار بعدی در مجموعه داده ۲

- اندیس‌های متناظر آن $i_1 = 3$، $i_2 = 2$ و $i_3 = 3$
- مقادیر توابع درهم‌ساز جهت $s_1(2) = 1$، $s_2(2) = 1$ و $s_3(2) = -1$

بنابراین

- افزایش شمارنده‌های c_1^3 و c_2^2 و کاهش c_3^3
- وقوع تصادم نرم
- مقدار ۲ تغییر شمارندهٔ مورد استفادهٔ مقدار ۴ (در همان جهت)
- منجر به بیش‌تخمین مقدار موجود در شمارنده c_1^3 برای هر دو مقدار

	1	2	3	4	5
h_1	0	0	5	0	0
h_2	0	1	4	0	0
h_3	-4	0	-1	0	0

مثال- ساختن طرح کلی شمارش

ادامه روال برای سایر مقادیر

مقدار ۳

- کاهش شمارنده‌های $c_1^{۱}$ و $c_۲^{۳}$ و افزایش شمارنده $c_۳^{۴}$

مقدار ۵

- کاهش شمارنده‌های c_1^{3} و $c_۲^{۴}$ و افزایش $c_۳^{۴}$

مقدار ۶

- کاهش شمارنده‌های $c_1^{۴}$ و $c_۳^{3}$ و افزایش $c_۲^{۱}$

	1	2	3	4	5
h_1	-6	0	9	-1	0
h_2	1	3	1	-1	0
h_3	-7	0	-4	7	0

طرح کلی شمارش

در نظریه احتمال

روش معمول برای ساخت تقریب‌های بهتر از تعدادی آزمایش توزیع شده تصادفی

- استفاده از میانگین و میانه

استفاده الگوریتم طرح کلی شمارش برای محاسبه تخمین نهایی بسامد

- از میانه
- به دلیل پایداری و حساسیت کمتر نسبت به نقاط پرت

طرح کلی شمارش

الگوریتم ۶: تخمین بسامد با طرح کلی شمارش

ورودی: مقدار $x \in D$

ورودی: طرح کلی شمارش با شمارنده‌های $p \times m$

خروجی: تخمین بسامد

$$\hat{f} = \{\hat{f}_j\}_{i=۱}^{p}$$

برای $j \leftarrow ۱$ تا p انجام بده

$$i \leftarrow h_j(x)$$

$$\hat{f}_j \leftarrow s_j(x) \cdot c_j^i$$

برگرداندن میانه $(\hat{f}_۱, \hat{f}_۲, \dots, \hat{f}_p)$

طرح کلی شمارش

زمان اجرا

- به‌روزرسانی برای هر مقدار O(p)
- برای یافتن میانه p مقدار،
  - صرف زمان خطی با استفاده از یکی از الگوریتم‌های انتخاب

بنابراین زمان کلی پرس‌وجو O(p)

مثال- تخمین بسامد با طرح کلی شمارش

	1	2	3	4	5
h_1	-6	0	9	-1	0
h_2	1	3	1	-1	0
h_3	-7	0	-4	7	0

مثال- تخمین بسامد با طرح کلی شمارش

	1	2	3	4	5
h_1	-6	0	9	-1	0
h_2	1	3	1	-1	0
h_3	-7	0	-4	7	0

تخمین بسامد مقدار ۴

با شمارنده‌های متناظر c_1^3، c_2^3، و c_3^1

جهت‌های به‌روزرسانی $s_1(4) = 1$، $s_2(4) = 1$ و $s_3(4) = -1$

به عنوان تخمین، محاسبهٔ میانه مقادیر وزنی آن شمارنده‌ها

$$\hat{f} = median(s_1(4) \cdot c_1^3, s_2(4) \cdot c_2^3, s_3(4) \cdot c_3^1) = median(9,1,7) = 7$$

مثال- تخمین بسامد با طرح کلی شمارش

بنابراین، بسامد تخمینی مقدار ۴ برابر با ۷

- همان شمارش صحیح از مجموعه داده

مثال- تخمین بسامد با طرح کلی شمارش

مقدار ۲

- با شمارنده‌های متناظر c_1^3، c_2^2 و c_3^3
- با مقادیر توابع درهم‌ساز جهت به‌روزرسانی $s_1(2) = 1$، $s_2(2) = 1$ و $s_3(2) = -1$

بنابراین، تخمین بسامد برای مقدار ۲

- $\hat{f} = median(s_1(2) \cdot c_1^3, s_2(2) \cdot c_2^2, s_3(2) \cdot c_3^3) = median(9,3,4) = 4,$

تخمین بیش از حد مقدار واقعی ۳

طرح کلی شمارش- یافتن k مقدار پربسامد

کاربرد الگوریتم طرح کلی شمارش

- شناخته شده جهت یافتن k مقدار پربسامد
- یا مسئله بسامد

طرح کلی شمارش- یافتن k مقدار پربسامد

در گذری از جریان داده

- با داشتن آرایه منظم $m \times p$ از شمارنده‌ها و توابع درهم‌ساز $\{h_j\}_{j=1}^{p}$ و $\{s_j\}_{j=1}^{p}$،
- نگه‌داری مجموعه‌ای از X^* از k مقدار با بسامد بالا
- نمایه‌سازی هر مقدار x از جریان داده به داده‌ساختار طبق الگوریتم ۵
- در صورت عدم وجود مقدار در مجموعه X^* و وجود ظرفیت افزودن مقدار
  - درج آن
- در غیر این صورت،
  - تخمین بسامد با الگوریتم ۶
  - در صورت بزرگتر بودن از کوچکترین بسامد موجود در مجموعه
  - افزودن مقدار x به X^*
  - حذف مقدار با کوچکترین بسامد

طرح کلی شمار

الگوریتم ۷- دریافت مقادیر پربسامد با طرح کلی شمارش

ورودی: جریان داده D

ورودی: طرح کلی شمارش با شمارنده‌های $p \times m$

خروجی: مقادیر پربسامد

$X^* \leftarrow \emptyset$

برای $x \in D$ انجام بده

Update(x)

اگر $x \in X^*$ آنگاه

continue

اگر $|X^*| < k$ آنگاه

$X^* \leftarrow X^* \cup \{x\}$

در غیر این صورت

$\hat{f} \leftarrow Frequency(x)$

$(x^*_{min}, \hat{f}^*_{min}) \leftarrow \min_{x^* \in X^*} Frequency(x^*)$

اگر $\hat{f} > \hat{f}^*_{min}$ آنگاه

$X^* \leftarrow X^* \cup \{x\} \backslash \{x^*_{min}\}$

برگرداندن X^*

مثال – پربسامدترین مقادیر

$k = 3$ مقدار پربسامد در مجموعه داده

$\{4,4,4,4,2,3,5,4,6,4,3,3,4,2,3,3,3,2\}$.

مثال - پربسامدترین مقادیر

با توجه به الگوریتم ۷

- ایجاد داده‌ساختار طرح کلی شمارش و مجموعهٔ X^* برای ذخیره نامزدهای پربسامد

شروع به خواندن اعضای مجموعه داده

- اولین مقدار ۴
- بنابراین، مانند مثال قبل
  - افزودن شمارنده‌های c_1^3 و c_4^3 و کاهش شمارنده c_3^1

	1	2	3	4	5
h_1	0	0	1	0	0
h_2	0	0	1	0	0
h_3	-1	0	0	0	0

مثال - پربسامدترین مقادیر

خالی بودن مجموعه X^*

- درج مقدار ۴
- $X^* = [4]$

سه مقدار بعدی در جریان داده نیز برابر با ۴

تغییر شمارنده‌های آنها بدون اعمال تغییر در X^*

	1	2	3	4	5
h_1	0	0	4	0	0
h_2	0	0	4	0	0
h_3	-4	0	0	0	0

مثال - پربسامدترین مقادیر

مقدار بعدی ۲

- افزایش شمارنده‌های c_1^3 و c_2^2 و کاهش c_3^3
- نبود مقدار ۲ در مجموعه پربسامدترین نامزدها
- X^* دارای ظرفیت خالی
- افزودن مقدار ۲ به مجموعه
  - $X^* = [4,2]$

	1	2	3	4	5
h_1	0	0	5	0	0
h_2	0	1	4	0	0
h_3	-4	0	-1	0	0

مثال - پربسامدترین مقادیر

مقدار ورودی بعدی ۳

- نمایه‌سازی آن در داده‌ساختار
  - با کاهش شمارنده‌های c_1^1 و c_2^3 و افزایش شمارنده c_3^4
- مجموعه X^* شامل دو مقدار از سه مقدار ممکن
  - افزودن مقدار ۳ به مجموعه
  - $X^* = [4,2,3]$

	1	2	3	4	5
h_1	-1	0	5	0	0
h_2	0	1	3	0	0
h_3	-4	0	-1	1	0

مثال - پربسامدترین مقادیر

مقدار ۵ از مجموعه داده

تغییر طرح فشرده با کاهش شمارنده‌های c_1^3 و c_2^4 و افزایش شمارنده c_3^4

	1	2	3	4	5
h_1	-1	0	4	0	0
h_2	0	1	3	-1	0
h_3	-4	0	-1	2	0

مثال - پربسامدترین مقادیر

نبودن مقدار ۵ در مجموعه X^*

- اما مجموعه دارای حداکثر ظرفیت $k = 3$ مقدار تحت نظر
- نیاز به تخمین بسامدهای مقادیر موجود در مجموعه و مقدار ۵
  - با استفاده از الگوریتم ۶ برای داده‌ساختار فعلی

$$\hat{f}(5) = median(-c_1^3, -c_2^4, c_3^4) = median(-4,1,2) = 1,$$
$$\hat{f}(4) = median(c_1^3, c_2^3, -c_3^1) = median(4,3,4) = 4,$$
$$\hat{f}(2) = median(c_1^3, c_2^2, -c_3^3) = median(4,1,1) = 1,$$
$$\hat{f}(3) = median(-c_1^1, -c_2^3, c_3^4) = median(1,-3,2) = 1$$

مثال – پربسامدترین مقادیر

بزرگتر نبودن بسامد تخمینی مقدار فعلی ۵ از حداقل بسامد مقادیر موجود در مجموعه

- عدم تغییر مجموعه مقادیر تحت نظر
- $X^* = [4,2,3]$

مدیریت باقیمانده مقادیر ورودی به روش مشابه

شکل نهایی داده‌ساختار پس از اعمال مراحل تمامی مقادیر ورودی

	1	2	3	4	5
h_1	-6	0	9	-1	0
h_2	1	3	1	-1	0
h_3	-7	0	-4	7	0

مثال - پربسامدترین مقادیر

مجموعه سه مقدار پربسامد

$$X^* = [4,2,3].$$

مرتب‌نبود پربسامدترین مقادیر در X^*

امکان استفاده از الگوریتم ۶ جهت تخمین بسامد آنها

طرح کلی شمارش- حل مسئلهٔ پربسامد

اکنون امکان تکمیل مراحل و حل مسئله مقادیر پربسامد

یافتن k مقدار پربسامد

نگه‌داری شمارنده N از مقادیر پردازش شده

- جهت محاسبهٔ آستانهٔ بسامد $f^* = \frac{N}{k}$ در نمایه‌سازی هر مقدار جدید

در صورت بزرگتر بودن بسامد تخمینی مقدار فعلی از آستانه

- درج آن به عنوان نامزد مقادیر پربسامد در هرم X^*
- همچنین، حذف مقدار با بسامد کوچکتر از آستانه فعلی f^* از هرم حذف

طرح کلی شمارش- حل

الگوریتم ۸- تعیین مقادیر پربسامد با طرح کلی شمارش

ورودی: جریان داده D

ورودی: طرح کلی شمارش با $p \times m$ شمارنده

خروجی: مقادیر پربسامد

$N \leftarrow 0, X^* \leftarrow \emptyset$

برای $x \in D$ انجام بده

$N \leftarrow N + 1$

Update(x)

$\hat{f} \leftarrow Frequency(x)$

$f^* \leftarrow \frac{N}{k}$

اگر $\hat{f} \geq f^*$ آنگاه

$X^* \leftarrow X^* \cup \{(x, \hat{f})\}$

برای $(x^*, \hat{f}) \in X^*$ انجام بده

اگر $\hat{f} \leq f^*$ آنگاه

$X^* \leftarrow X^* \setminus \{(x^*, \hat{f})\}$

برگرداندن X^*

طرح کلی شمارش- حل مسئلهٔ حداکثر تغییر

الگوریتم طرح کلی شمارش مناسب یافتن مقادیر با بیشترین تغییر بسامد،

- مشهور به مسئله حداکثر تغییر

با داشتن جریان‌های داده از دو دوره قابل مقایسه

- امکان ایجاد داده‌ساختار طرح کلی شمارش برای هر یک از آنها
- نگه‌داری مقادیر با بیشترین اختلاف در هرم X^*

با نمایه شدن مقدار جدید در هر دفعه

- تخمین بسامد آنها با استفاده از الگوریتم ۶
- به‌روزرسانی هرم
  - جهت نگه‌داری مقادیر با بیشترین تغییر

خروجی الگوریتم k مقدار با بزرگترین مقادیر تغییر بسامد

طرح کلی شمارش- حل

الگوریتم ۹- تعیین تغییر بیش با طرح کلی شمارش

ورودی: جریان داده $D_{\text{قدیم}}$ و $D_{\text{جدید}}$

ورودی: دو طرح کلی شمارش با $p \times m$ شمارنده قدیم و جدید

خروجی: مقادیر پربسامد

$N_{\text{قدیم}} \leftarrow 0, X^*_{\text{قدیم}} \leftarrow \emptyset$

برای $x \in D_{\text{قدیم}}$ انجام بده

$N_{\text{قدیم}}++$

Update(x)

$\hat{f} \leftarrow Frequency(x)$

$f^* \leftarrow \frac{N_{\text{قدیم}}}{k}$

اگر $\hat{f} \geq f^*$ آنگاه

$X^*_{\text{قدیم}} \leftarrow X^*_{\text{قدیم}} \cup \{(x, \hat{f})\}$

برای $(x^*, \hat{f}) \in X^*_{\text{قدیم}}$ انجام بده

اگر $\hat{f} \leq f^*$ آنگاه

$X^*_{\text{قدیم}} \leftarrow X^*_{\text{قدیم}} \setminus \{(x^*, \hat{f})\}$

$N_{\text{جدید}} \leftarrow 0, X^*_{\text{جدید}} \leftarrow \emptyset$

برای $x \in D_{\text{جدید}}$ انجام بده

$N_{\text{جدید}}++$

Update(x)

$\hat{f} \leftarrow Frequency(x)$

$f^* \leftarrow \frac{N_{\text{جدید}}}{k}$

اگر $\hat{f} \geq f^*$ آنگاه

$X^*_{\text{جدید}} \leftarrow X^*_{\text{جدید}} \cup \{(x, \hat{f})\}$

برای $(x^*, \hat{f}) \in X^*_{\text{جدید}}$ انجام بده

اگر $\hat{f} \leq f^*$ آنگاه

$X^*_{\text{جدید}} \leftarrow X^*_{\text{جدید}} \setminus \{(x^*, \hat{f})\}$

$X^* = X^*_{\text{قدیم}} \cup X^*_{\text{جدید}}$

$T \leftarrow \phi$

برای $x \in X^*$

t محاسبهٔ اختلاف مقدار بسامد جدید و قدیم.

$T = T \cup \{t\}$

مرتب‌سازی نزولی |T|

برگرداندن k مقدار اول T بر اساس قدر مطلق آن

ویژگی‌ها

تضمین طرح کلی شمارش بر کران بالای خطای تخمین $\varepsilon \cdot n$ برای بسامدها
- با احتمال حداقل $1 - \delta$

کاهش احتمال تخمین بد با افزایش تعداد توابع درهم‌ساز p

با خطای استاندارد مطلوب δ
- توصیهٔ تعداد توابع درهم‌ساز
  - تعداد ردیف‌های داده‌ساختار

$$p = \lceil \ln \frac{1}{\delta} \rceil$$

ویژگی‌ها

هرچه m بزرگتر

- احتمال وقوع تصادم کمتر
- یا خطای تخمین $\epsilon \cdot n$ کمتر

در عین حال، با p بزرگتر

- ، تخمین‌گرهای بیشتری برای محاسبه مقدار نهایی
- قابل اعتمادتر
- توصیه در مورد تعداد شمارنده‌ها m

$$m \approx \lceil \frac{2.71828}{\epsilon^2} \rceil$$

ویژگی‌ها

کل فضای مورد نیاز داده‌ساختار $O(m \cdot p + 2p)$

نگهداری ماتریس شمارش به اندازه $p \times m$ و دو تابع درهم‌ساز به ازای هر ردیف

اگر دو داده‌ساختار دارای اندازه m یکسان

امکان افزودن و کاهیدن آنها به و از یکدیگر

مفید برای پردازش جریان توزیع شده

استفاده در Apache Hive و سایر نرم‌افزارهای انبار داده

تمایل برنامه‌های جدیدتر از جانشین آن،

- الگوریتم طرح کلی شمارش کمینه،
- به دلیل نیاز به فضای کمتر و زمان اجرای کمتر

طرح کلی شمارش کمینه

طرح کلی شمارش کمینه Count-min Sketch یا CM Sketch

- داده‌ساختار احتمالاتی ساده و فضا-کارآمد
- برای تخمین بسامد مقادیر در جریان‌های داده
- پرداختن به مسئله مقادیر پربسامد

در سال ۱۳۸۲

- پیشنهاد گراهام کورمودی و شان موتوکریشنان
- انتشار در سال ۱۳۸۴

طرح کلی شمارش کمینه

مانع اصلی کاربرد مستقیم فیلتر بلوم شمارنده در تخمین بسامد اشتراک یک آرایه واحد از شمارنده‌ها برای همه توابع درهم‌ساز دارای اشکال از سوی تصادم‌های سخت و نرم

طرح کلی شمارش کمینه

کیفیت تخمین

- سخت تحت تأثیر احتمال تصادم‌های درهم
- منجر به بیش‌تخمینی برای شمارنده‌ها

تعداد مقادیر در جریان داده عظیم

- قطعی بودن تصادم با مقادیر با بسامد بالا
- منجر به بی‌فایدگی و بی‌ثمری چنین تقریبی به دلیل بیش‌تخمینی بزرگ همه شمارنده‌ها

طرح کلی شمارش کمینه

طرح کلی شمارش کمینه

- در پی حل فقدان تخمین‌های با اطمینان بالا برای محاسبه بسامد با دقت کافی

جانشینی آرایه واحد m شمارنده با جدول درهمی از p آرایه از m شمارنده

به‌روزرسانی زیرمجموعه‌های مختلفی از شمارنده‌ها به جای به‌روزرسانی هر شمارنده با هر مقدار

- هدف m فشرده‌سازی جریان داده $D = \{x_1, x_2, \dots, x_n\}$
- به دلیل $m \ll n$، فشرده‌سازی «با اتلاف» منجر به خطا

کاهش خطاها،

- با استفاده از p تابع درهم‌ساز با آرایه‌ای اختصاصی از m شمارنده برای هر کدام
  - معرفی، آزمایش‌های مستقل زیاد

طرح کلی شمارش کمینه

داده‌ساختار فضا-کارآمد

- متشکل از آرایه $p \times m$ از شمارنده‌ها $\{c_j^i\}$
- p تابع درهم‌ساز یک به یک مستقل $h_1, h_2, \ldots, h_p$
- نگاشت مقادیر وروردی به بازهٔ $\{1, 2, \ldots, m\}$
- فراهم کردن امکان نمایه‌سازی مقادیر از جریان داده
- منجر به به‌روزرسانی شمارنده‌ها
- تعداد دفعاتی نمایه شدن مقدار به عنوان تخمین بسامد برای آن

طرح کلی شمارش کمینه- به‌روزرسانی

الگوریتم ۱۰- به‌روزرسانی طرح کلی شمارش-کمینه

ورودی: مقدار $x \in D$

ورودی: طرح کلی شمارش-کمینه با $p \times m$ شمارنده

برای $j \leftarrow ۱$ تا p انجام بده

$i \leftarrow h_j(x)$

$c_j^i \leftarrow c_j^i + ۱$

طرح کلی شمارش کمینه-به‌روزرسانی

با فرض محاسبهٔ هر تابع درهم‌ساز $\{h_j\}_{j=1}^{p}$ در زمان ثابت

- زمان اجرای روش به‌روزرسانی O(p)

مثال - ساختن طرح کلی شمارش کمینه

مجموعه داده با $n = 18$ مقدار

$$\{4,4,4,4,2,3,5,4,6,4,3,3,4,2,3,3,3,2\}$$

ایجاد طرح کلی شمارش کمینه

- از $m = 4$ شمارنده با استفاده از $p = 2$ تابع درهم‌ساز مبتنی بر مورمور۳ و فن‌و۱الف

$$h_1(x) = MurmurHash3(x)\%\ 4\ +\ 1,$$
$$h_2(x) = FNV1a(x)\%\ 4\ +\ 1$$

مثال - ساختن طرح کلی شمارش کمینه

در ابتدا، مقادیر داده‌ساختار برابر صفر

	1	2	3	4
h_1	0	0	0	0
h_2	0	0	0	0

مثال – ساختن طرح کلی شمارش کمینه

خواندن مقادیر از مجموعه داده می‌کنیم.

اولین مقدار ۴

- محاسبهٔ مقادیر درهم‌ساز برای تعیین شمارنده‌هایی نیازمند به‌روزرسانی

$$i_1 = h_1(4) = 4,$$
$$i_2 = h_2(4) = 4$$

مثال - ساختن طرح کلی شمارش کمینه

هر دو تابع درهم‌ساز تولید مقداری یکسان

- مشکلی پیش نیاوردن به دلیل اختصاصی بودن آرایه هر یک از شمارنده‌ها برای هر تابع درهم‌ساز
- افزایش شمارنده‌های c_1^4 و c_2^4

	1	2	3	4
h_1	0	0	0	1
h_2	0	0	0	1

مثال - ساختن طرح کلی شمارش کمینه

سه مقدار بعدی همگی برابر با ۴ هستند، بنابراین همان شمارنده‌ها را به‌روزرسانی می‌کنیم:

	1	2	3	4
h_1	0	0	0	4
h_2	0	0	0	4

مثال - ساختن طرح کلی شمارش کمینه

مقدار بعدی در مجموعه داده ۲

- نمایه‌های متناظر آن $i_1 = 4$ و $i_2 = 1$
- افزایش شمارنده‌های c_1^4 و c_2^1
- وجود تصادم نرم
  - به‌روزرسانی شمارنده‌ای با مقدار ۲ استفاده شده با مقدار ۴
- منجر به تخمین بیش از حد مقدار موجود در شمارنده c_1^4 برای هر دو مقدار

	1	2	3	4
h_1	0	0	0	5
h_2	1	0	0	4

مثال - ساختن طرح کلی شمارش کمینه

پردازش بقیهٔ مقادیر

به‌روزرسانی

- شمارنده‌های c_1^1 و c_2^3 برای مقدار ۳،
- شمارنده‌های c_1^1 و c_2^1 برای مقدار ۵،
- شمارنده‌های c_1^1 و c_2^2 برای مقدار ۶
- تصادم هر دو شمارنده برای مقدار ۶ با مقادیر دیگر
  - بنابراین انتظارتخمین بیش از حد مقدار آن

	1	2	3	4
h_1	8	0	0	10
h_2	1	4	6	7

طرح کلی شمارش کمینه

با هر بار نمایه شدن مقدار x

افزایش شمارنده‌های $c_j^{h_j(x)}$ برای هر ردیف j

- عدم کاهش
- شمارنده‌ها کرانی بالا از بسامدها

$$f(x) \leq c_j^{h_j(x)}, j = 1,2, \dots, p$$

طرح کلی شمارش کمینه

عدم امکان شمارنده‌ها به تخمین کمتر از حد بسامد واقعی f(x)

- معمولا تخمین بیش از حد
- دلیل: $m \ll n$ و وجود تصادم‌های زیادی به طوری که
- $h_j(x) = h_j(y)$ برای $x \neq y$

به معنای نمایه شدن مقدار y در داده‌ساختار،منجر به افزایش شمارنده مقدار x

حاصل: p تخمین دارای خطای یک‌طرفه

- همه آنها بیش‌تخمینی از مقدار واقعی

روش معمول برای ساختن تقریب بهتر از تعدادی تخمین،

- میانگین‌گیری
- امکان بدتر شدن تخمین با میانگین‌گیری

بهترین تخمین؟

طرح کلی شمارش کمینه

عدم امکان شمارنده‌ها به تخمین کمتر از حد بسامد واقعی f(x)

- معمولا تخمین بیش از حد
- دلیل: $m \ll n$ و وجود تصادم‌های زیادی به طوری که
- $h_j(x) = h_j(y)$ برای $x \neq y$

به معنای نمایه شدن مقدار y در داده‌ساختار،منجر به افزایش شمارنده مقدار x

حاصل: p تخمین دارای خطای یک‌طرفه

- همه آنها بیش‌تخمینی از مقدار واقعی

روش معمول برای ساختن تقریب بهتر از تعدادی تخمین،

- میانگین‌گیری
- امکان بدتر شدن تخمین با میانگین‌گیری

بهترین تخمین؟

- کوچکترین

طرح کلی شمارش کمینه

الگوریتم ۱۱- تخمین بسامد با طرح کلی شمارش-کمینه

ورودی: مقدار $x \in D$

ورودی: طرح کلی شمارش-کمینه با شمارنده‌های $p \times m$

خروجی: تخمین بسامد

$$\hat{f} = \{\hat{f}_j\}_{i=۱}^{p}$$

برای j از ۱ تا p انجام بده

$$i \leftarrow h_j(x)$$

$$\hat{f}_j \leftarrow c_j^i$$

برگرداندن $min\left(\hat{f}_۱, \hat{f}_۲, \dots, \hat{f}_p\right)$

طرح کلی شمارش کمینه

امکان یافتن کمینه p مقدار در زمان خطی

زمان اجرای روش تخمین بسامد O(p)

- همانند به‌روزرسانی.

مثال-طرح کلی شمارش کمینه-تخمین بسامد

داده‌ساختار قبلی

	1	2	3	4
h_1	8	0	0	10
h_2	1	4	6	7

مثال-طرح کلی شمارش کمینه-تخمین بسامد

تخمین بسامد مقدار ۴

شمارنده‌های متناظر آن c_1^4 و c_2^4

با استفاده از الگوریتم ۱۱

- محاسبهٔ کمینهٔ شمارنده‌ها به عنوان تخمین تعداد تکرار

$$\hat{f} = min\,(c_1^4, c_2^4) = min\,(10,7) = 7$$

مثال-طرح کلی شمارش کمینه-تخمین بسامد

بسامد تخمینی مقدار ۴ برابر با ۷

- برابر شمارش صحیح از مجموعه داده

مقدار ۶

شمارنده‌های متناظر c_1^1 و c_2^2

در مثال قبلی هر دو آنها به دلیل تصادم

- مورد استفادهٔ مقادیر دیگر

تخمین بسامد مقدار ۶

$$\hat{f} = min\,(c_1^1, c_2^2) = min\,(8,4) = 4,$$

مثال-طرح کلی شمارش کمینه-تخمین بسامد

بسامد واقعی مقدار ۶ برابر ۱

بیش‌تخمینی قابل توجه

در صورت قصد نگه‌دریا با دقت بهتر و نادرسازی چنین تصادم‌ها

- نیاز به توابع درهم‌ساز و شمارنده‌های بیشتر
- به تبع افزایش زمان محاسباتی و ذخیره‌سازی

یادآوری طرح کلی شمارش

با داشتن نحوه تخمین بسامد عناصر

- الگوریتم طرح کلی شمارش کمینه
  - امکان تعیین پربسامدترین مقادیر
- مشابه طرح کلی شمارش،
  - ساده‌ترین رویکرد مستلزم حفظ مجموعه‌ای از نامزدهای پربسامدترین مقادیر و همچنین داده‌ساختار اصلی

سپس، خواندن از جریان داده

- به‌روزرسانی با هر مقدار ورودی
- در صورت نبودن مقدار در مجموعه تحت‌نظر و نرسیدن به حد ظرفیت
  - افزوند آن
- در صورت بودن مجموعه در حداکثر ظرفیت خود
  - افزوند مقدار فعلی با شرط بزرگ‌تر بودن بسامد تخمینی آن از حداقل بسامد موجود در مجموعه
  - جانشینی مقدار با کوچکترین بسامد
- در پایان، مقادیر موجود در X^* به عنوان پربسامدترین مقادیر در جریان داده

طرح کلی شمارش کمینه- مسئلهٔ پربسامد

در گذری از جریان داده

- تخصیصی آرایه $p \times m$ از شمارنده‌های c
- P تابع درهم‌ساز
- اختصاص شمارنده N جهت ذخیرهٔ تعداد مقادیر دیده شده تاکنون
- هرم X^* نگه‌داری حداکثر k مقدار پربسامد بالقوه
- استفاده از آستانه بسامد $f^* = \frac{N}{k}$ برای تصمیم‌گیری پربسامدی مقدار

به ازای هر مقدار x در جریان داده،

- اجرای روش به‌روزرسانی و به دنبال آن تخمین بسامد
  - اگر $\hat{f}(x) \geq f^*$، آنگاه واجد شرایط بودن مقدار به عنوان نامزد مقدار پربسامد
  - در صورت نبود مقدار در هرم، ذخیره آن همراه با بسامد آن
  - در غیر این صورت، به‌روزرسانی بسامد ذخیره شده با مقدار جدید

طرح کلی شمارش کمینه- مسئلهٔ پربسامد

افزای شمارنده N با هر مقدار پردازش شده

- افزایش مذکور منجر به کاهش، بسامد تخمینی برای برخی از مقادیر در هرم به زیر f^*
- حذف چنین مقادیری از هرم در هر مرحله

در پایان پردازش،

- همه مقادیر موجود در هرم به عنوان مقادیر پربسامد
- مطابق تعریف، وجود حداکثر k مقدار پربسامد در جریان داده

طرح کلی شمارش ک

الگوریتم ۱۲- مقادیر پربسامد با طرح کلی شمارش-کمینه

ورودی: جریان داده D

ورودی: طرح کلی شمارش-کمینه با شمارنده‌های $p \times m$

خروجی: مقادیر پربسامد

$N \leftarrow ۰, X^* \leftarrow \emptyset$

برای $x \in D$ انجام بده

$N \leftarrow N + ۱$

Update(x)

$\hat{f} \leftarrow Frequency(x)$

$f^* \leftarrow \frac{N}{k}$

اگر $\hat{f} \geq f^*$ آنگاه

$X^* \leftarrow X^* \cup (x, \hat{f})$

برای $(x^*, \hat{f}) \in X^*$ انجام بده

اگر $\hat{f} \leq f^*$ آنگاه

$X^* \leftarrow X^* \setminus (x^*, \hat{f})$

برگرداندن X^*

طرح کلی شمارش کمینه- مسئلهٔ پربسامد

هرم برای مسئله ϵ-مقادیر پربسامد با $\epsilon = \frac{1}{2k}$

- نیاز به کار اضافی $O\left(\log\frac{1}{\epsilon}\right)$ به ازای هر مقدار

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

تنظیمات مثال قبلی

یافتن $k = 3$ مقدار پربسامد در حین پردازش مجموعه داده

$\{4,4,4,4,2,3,5,4,6,4,3,3,4,2,3,3,3,2\}$

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

ایجاد داده‌ساختار، شمارنده N از مقادیر پردازش شده، هرم X^* ذخیره کنندهٔ حداکثر k نامزد مقادیر پربسامد

خواندن داده‌ها

اولین مقدار ۴

- افزایش شمارنده‌های مربوطه c_1^4 و c_2^4

	1	2	3	4
h_1	0	0	0	1
h_2	0	0	0	1

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

تاکنون پردازش N = 1 مقدار

- آستانه f^* هرم X^* برابر با $\frac{1}{3}$
- تخمین بسامد مقدار ۴ از برابر ۱
  - بالاتر از آستانه
- افزودن مقدار و بسامد آن به هرم
  - $X^* = [(4,1)]$.

مقدار بعدی دوباره ۴

- افزایش همان شمارنده‌ها

	1	2	3	4
h_1	0	0	0	2
h_2	0	0	0	2

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

تاکنون پردازش N = 2 مقدار

- آستانه f^* هرم X^* برابر با $\frac{2}{3}$
- تخمین بسامد مقدار ۴ از برابر 2
  - بالاتر از آستانه
- وجود مقدار
  - به‌روزرسانی بسامد آن در هرم
  - $X^* = [(4,2)]$

پردازش ۱۴ مقدار بعدی به روش مشابه

- (تا N = 16)
- عدم تغییر در تعداد مقادیر هرم
- مقدار ۴ تنها نامزد مقدار پربسامد تاکنون
  - $X^* = [(4,7)]$

	1	2	3	4
h_1	7	0	0	9
h_2	1	3	6	7

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

مقدار بعدی در مجموعه داده ۳

- افزایش شمارنده‌های c_1^1 و c_2^3 آن

	1	2	3	4
h_1	7	0	0	9
h_2	1	3	6	7

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

تاکنون پردازش $N = 17$ مقدار

- آستانه f^* هرم X^* برابر با $\frac{17}{3}$ یا حدود ۵,۳۳
- تخمین بسامد مقدار ۳ برابر $\hat{f} = min(7,6) = 6$
  - بالاتر از آستانه
- افزودن مقدار و بسامد آن به هرم
  - $X^* = [(4,7),(3,6)]$.

همه مقادیر در هرم بسامدهای به اندازه کافی بزرگ

- عدم حذف هیچ‌یک

مثال- طرح کلی شمارش کمینه- مسئله پربسامد

خواند آخرین مقدار برابر ۲

تاکنون پردازش $N = 18$ مقدار

- آستانه f^* هرم X^* برابر با $\frac{18}{3}$ یا 6
- تخمین بسامد مقدار ۲
  - کوچکتر از آستانه
- عدم تغییر در هرم
  - $X^* = [(4,7),(3,6)]$.

فهرست نهایی مقادیر پربسامد برابر با $X^* = [(4,7),(3,6)]$

ویژگی‌ها

طرح کلی شمارش کمینه

- هم تقریبی و هم احتمالاتی
- در نتیجه: تاثیر دو پارامتر، خطای ϵ در پاسخ به پرس‌وجوی خاص و احتمال خطای δ، بر نیازهای فضا و زمان
- تضمین عدم تجاوز خطای تخمین بسامدها از ϵn با احتمال حداقل $1 - \delta$

ویژگی‌ها

استفادهٔ گسترده از طرح کلی شمارش کمینه

- تحلیل ترافیک و برنامه‌های استخراج درون‌جریانی مبتنی بر چارچوب‌های پردازش جریان توزیع شده
- اجرا شده بر مواردی مانند Apache Spark، Apache Storm، Apache Flink
- همچنین پیاده‌سازی‌هایی در پایگاه‌های داده پرکاربدر مانندRedis و PostgreSQL

ویژگی‌ها

مشابه طرح کلی شمارش

- افزایش تعداد توابع درهم‌ساز p منجر به کاهش احتمال تخمین بد
- با خطای استاندارد مطلوب δ
  - توصیه برای تعداد توابع درهم‌ساز متناظر ردیف‌های داده‌ساختار

$$p = \lceil \ln 1/\delta \rceil$$

ویژگی‌ها

بزرگتر بودنm منجر به احتمال وقوع تصادم کمتر

- کاهش خطای بیش‌تخمینی $\epsilon \cdot n$

p بزرگتر،

- منجر به تخمین‌های بیشتری برای محاسبه مقدار کمینهٔ حداقل
- منجر به قابل اعتمادتر شدن نتایج

توصیه در مورد تعداد شمارنده‌ها m

$$m \approx \left\lceil \frac{2.71828}{\epsilon} \right\rceil$$

ویژگی‌ها

و مقایسه با تعداد لازم طرح کلی شمارش

سازگارتر بودن طرح کلی شمارش کمینه نسبت به طرح کلی شمارش با فضا

دارای داده‌ساختار آرایه دو بعدی به اندازه $p \times m$ و p تابع درهم‌ساز

- نیاز به فضای $O(mp + p)$
  - با فرض ذخیرهٔ هر تابع درهم‌ساز در فضای $O(1)$

مثال - تخمین فضای مورد نیاز

برای داشتن خطای استاندارد δ حدود یک درصد،

نیاز به حداقل $p = \left\lceil \ln \frac{1}{0.01} \right\rceil = 5$ تابع درهم‌ساز

انتظار نمایه شدن ۱۰ میلیون ($n = 10^7$) مقدار

بیش‌تخمینی ثابت ۱۰ د

نیاز به $\epsilon = \frac{10}{10^7} = 10^{-6}$

- تعداد توصیه شده شمارنده‌ها

$$m = \frac{2.71828}{10^{-6}} \approx 2718280$$

مثال - تخمین فضای مورد نیاز

نتیجه: نیاز به نگه‌داری آرایه شمارنده‌ای به اندازه 2718280×5

با داشتن شمارنده‌های صحیح ۳۲ بیتی،

- کل داده‌ساختار نیاز به ۵۴,۴ مگابایت حافظه

ویژگی‌ها

امکان ادغام دو طرح کلی شمارش کمینه با اندازه یکسان

- با جمع ماتریسی ساده
- نتیجه: داده‌ساختاری برای اتحاد مجموعه‌های داده آنها

نتیجه: مزیت طرح کلی شمارش کمینه در کارهای MapReduce و جریانی موازی برای برنامه‌های داده بزرگ

ویژگی‌ها

شناسائی داده بزرگ

با حجم زیادی از داده با سرعت بالای ورودی

منجر به پیچیده شدن فضا و زمان به‌روزرسانی

پیاده‌سازی‌های عملی طرح کلی شمارش کمینه

- مصرف حافظه فقط تا چند صد مگابایت حافظه
- امکان مدیریت ده‌ها میلیون به‌روزرسانی در ثانیه