رتبه و چندک

محسن هوشمند

فهرست

رتبه

درک اطلاعات موجود در حجم انبوهی از داده‌های بدون ساختار برای انسان دشوار است. ازاین‌رو، خلاصه‌سازی داده‌ها از طریق استخراج شاخص‌ها و کمیت‌های آماری، یکی از مهم‌ترین روش‌های تحلیل و مواجهه با چنین داده‌هایی به شمار می‌رود. یکی از شاخص‌های مهم محاسبهٔ رتبه و چندک است. در فصل جاری، داده‌ساختارها و الگوریتم‌هائی مورد بررسی قرار می‌گیرند که امکان محاسبه ویژگی‌های مبتنی بر رتبه داده‌ها را با استفاده از مقدار اندکی حافظه و طی یک بار پیمایش داده‌ها فراهم می‌کنند.
چندک‌ها پرکاربردترین ویژگی‌های رتبه هستند. چندک-q $(0 \leq q \leq 1)$ مقداری از دنباله است که در آن کسری q از مقادیر دنباله کمتر یا مساوی آن هستند و $(1 - q)$ باقی‌مانده بیشتر یا مساوی آن هستند. همچنین، اگر دنباله دارای n مقدار باشد، مقدار چندک-q برابر عضوی از دنباله است که رتبه آن $q \cdot n$ است. چندک هنگامی که دنباله مرتب شده به صد قسمت مساوی تقسیم شود را صدک می‌خوانیم. بنابراین ۹۵-اُمین صدک همان چندک-۰.۹۵ است. چندک‌های ۰ و ۱ به ترتیب حداقل و حداکثر مقادیر در دنباله هستند و چندک ۰.۵ به عنوان میانه شناخته می‌شود.

ایان مونرو و مایکل پاترسون در سال ۱۳۵۹ اثبات کردند[1] که برای یافتن چندکی خاص دقیقا در p گذر یا پیمایش از داده‌ها به حافظه $O(n^{\frac{1}{p}})$ نیاز است. این بدان معناست که هیچ الگوریتم تک پیمایشی نمی‌تواند تولید مقدار دقیق چندک را در فضای زیرخطی تضمین کند. این یافته الگوریتم‌های تقریبی محاسبهٔ چندک را توجیه می‌کند. خطا در محاسبه چندک در عمل مورد غریبی نیست. زیرا محاسبهٔ چندک معمولا در داده‌های نوفه‌دار لازم می‌شود و توزیع‌های داده ناشناخته را تقریب می‌زند. بنابراین، در بیشتر موارد ما به چندک ϵ-تقریبی q علاقه‌مندیم، به این معنی که عنصری با رتبه آن در $[(q - \epsilon) \cdot n, (q + \epsilon) \cdot n]$ باشد، که در آن n تعداد مقادیر و $0 < \epsilon < 1$ پارامتر خطا است. لازم به ذکر است که ممکن است بیش از یک مقدار واجد شرایط چندک باشد.

تخمین ویژگی‌های مختلف رتبه مانند تلخیص‌های چندک نقش مهمی در روش‌های تشخیص نقاط پرت جریانی ایفا می‌کند. برای مثال، اگر تراکنش‌های تجارت الکترونیک آنلاین را برای تشخیص تقلب در کارت اعتباری نظارت می‌کنیم، به مکان‌های پرداخت غیرمعمول علاقه‌مندیم، آنهایی که در صدک نود و نهم توزیع مکان معمول مشتریان قرار نمی‌گیرند.

مثال ۱- تشخیص تقلب: تقلب مالی از مسائل اساسی پیش روی صنعت مالی است. به عنوان مثال، در سال ۱۳۹۴، تقلب در کارت‌های اعتباری و نقدی جهانی منجر به زیان‌هایی به مبلغ ۲۱.۸۴ میلیارد دلار شد. بسیاری از برنامه‌ها برای مداقه و شناسایی نشانه‌های تقلب مالی طراحی شده‌اند. چنین برنامه‌هایی اغلب از متغیرهای خاص متعددی استفاده می‌کنند که «درجه دورافتادگی» هر یک برای هر مشاهده بررسی می‌شود. برای مثال، متغیرهایی مانند کل مبلغ هزینه شده در کارت اعتباری و مقدار هزینه شده در روز را می‌توان استفاده کرد. برای هر مشاهده، درجه دورافتادگی را می‌توان با چندک‌های برخی از توزیع‌های هزینه تقریب زد. بنابراین، مشاهدات مشکوک به تقلب را می‌توان به عنوان نقاط پرت از طریق مقایسه با برخی چندک‌های بالا، به عنوان مثال، چندک ۰.۹۵، شناسایی کرد.

نظارت بر ترافیک وب حوزهٔ کاربردی و مهم دیگر خلاصه‌های رتبه است. با بررسی تلخیص‌ها مشکلات را بدون بررسی داده‌های اصلی و واقعی می‌توان زودتر تشخیص داد.

مثال ۲- نظارت بر تارمانه: تارمانه‌های بزرگ روزانه میلیون‌ها کاربر را مدیریت می‌کنند. برای مثال، در شهریور و مهر ۱۳۹۶ ویکی‌پدیا روزانه حدود ۵۰۰ میلیون بازدید را در تمام زبان‌های خود پردازش کرد، که تقریبا ۵.۷ هزار درخواست در ثانیه در بیش از ۳۰۰ سرور

[1] Selection and sorting with limited storage, Theoretical Computer Science, V. 12, 1980

در سراسر جهان بود. تأخیر در تارمانه یا به زبان دقیق‌تر تأخیر بین لحظهٔ ایجاد محتوا و لحظهٔ انتقال به بازدیدکننده، از جمله گزینه‌های مهم عملکرد هر تارمانه است. توزیع احتمالی میزان تأخیر معمولا چوله است، در نتیجه نظارت معمولا با رهگیری و بررسی چندک‌ها یا صدک‌های خاص بالا طراحی می‌شود. رایج‌ترین سوالات عبارتند از:

تأخیر برای ۹۵ درصد از درخواست‌ها برای یک سرور وب منفرد چقدر است؟
تأخیر برای ۹۹ درصد از درخواست‌ها برای کل وب‌سایت چقدر است؟
تأخیر برای ۹۵ درصد از درخواست‌ها برای کل وب‌سایت در ۱۵ دقیقه گذشته چقدر بود؟
در حالی که همه این سوالات را می‌توان با محاسبه چندک پاسخ داد، از نظر فنی تفاوت‌هایی دارند که ممکن است نیاز به استفاده از روش‌های مختلف داشته باشد. برای مثال، در حالی که برای سوال اول، می‌توان یک خلاصه را برای هر جریان منفرد محاسبه کرد، سوال دوم به الگوریتم‌های توزیع شده نیاز دارد که بتوانند آمار را روی داده‌های جریان‌های متعدد محاسبه کنند. در مقابل، سوال سوم فقط به زیرمجموعه‌ای از داده‌های جریان نیاز دارد که در یک پنجره زمانی تعریف می‌شود و چنین زیرمجموعه‌ای همیشه مشمول تغییر است.

وظیفه یافتن چندک-q، یا به عبارت دیگر، عناصری از دنباله مرتب شده از n مقدار که رتبه آنها $q.n$ است، که در آن $q \in (0,1)$، پرس‌وجوی چندک نامیده می‌شود. پرس‌وجوی میانه موردی خاص از پرس‌وجوی چندک با $q = 0.5$ است.

مسئلهٔ محاسبه چندک‌ها موضوع تازه‌ای نیست و در چند دهه گذشته، روش‌ها و الگوریتم‌های متعددی برای حل آن معرفی شده‌اند. با این حال، جریان‌های نامحدود داده که در کاربردهای داده بزرگ رایج هستند، چالش‌های تازه‌ای ایجاد کرده‌اند. این چالش‌ها به‌ویژه زمانی اهمیت پیدا می‌کنند که حافظه در دسترس محدود باشد و تنها یک بار پیمایش داده‌ها امکان‌پذیر باشد. اگرچه الگوریتم طرحواره شمارش-کمینه که پیش‌تر معرفی شد، امکان برآورد تقریبی چندک‌ها را فراهم می‌کند، اما در مقایسه با الگوریتم‌هایی که در این فصل بررسی می‌شوند، به فضای حافظه بسیار بیشتری نیاز دارد.
از سوی دیگر، می‌توان رتبه مقداری مشخص را در دنباله مرتب‌شده‌ای شامل n مقدار تعیین کرد که این مسئله با عنوان «پرس‌وجوی چندک معکوس» شناخته می‌شود. اگر رتبه مقدار مورد نظر را با $rank(x)$ و تعداد کل مقادیر را با n نشان دهیم، محاسبه چندک متناظر q به‌سادگی امکان‌پذیر است:

$$q = \frac{1}{n} \times rank(x)$$

برای بسیاری از برنامه‌ها، یافتن تعداد مقادیر از دنبالهٔ مرتب شده از n مقدار که در محدوده [a, b] هستند، که اغلب به عنوان پرس‌وجوی بازه شناخته می‌شود، نیز مهم است. در واقع، برای محاسبه چنین عددی، کافی است رتبه‌های مرزهای بازه را محاسبه کرده و تفاوت آنها را برگردانیم.

در ابتدا، با الگوریتم نمونه‌گیری تصادفی معرفی می‌شود، سپس با تلخیص-چ q-digest مبتنی بر درخت ساده ادامه داده و در نهایت، الگوریتم جدید تلخیص-د t-digest را معرفی می‌کنیم که از خوشه‌بندی برای تخمین کارآمد آمار مبتنی بر رتبه در جریان‌های نامحدود استفاده می‌کند.

نمونه‌گیری تصادفی

تکنیک نمونه‌گیری تصادفی، انتخاب بدون جایگزینی زیرمجموعه‌ای تصادفی از داده‌ها، در بسیاری از الگوریتم‌های علم رایانش تعبیه شده است. برای مسائل رتبه، می‌توان از این تکنیک برای گزارش چندک‌های محاسبه شده روی نمونه‌ها، به عنوان تقریبی از چندک‌های کل جریان داده استفاده کرد. مزیت یافتن چندک روی چنین نمونه‌های با تعداد کمتر این است که می‌توان پرس‌وجوهای چندک رتبه را با استفاده از الگوریتم‌های قطعی کلاسیک پاسخ داد. هر چند، برای داشتن تضمین‌هائی معین در مورد خطای چنین تقریبی، نمونه تصادفی باید به روش خاصی بدست آید، که حتی می‌تواند به داده‌ها وابسته باشد.

علاوه بر این، بسیاری از روش‌های نمونه‌گیری نیازمند اطلاع قبلی از اندازه مجموعه داده هستند. این موضوع در جریان‌های پیوسته داده که در بسیاری از کاربردهای داده بزرگ به کار می‌روند، محدودیت جدی به شمار می‌آید. یکی از راه‌حل‌های این مسئله، روش ساده «نمونه‌گیری مخزنی» است که جفری ویتر در سال ۱۳۶۴ معرفی کرد. این روش امکان تولید نمونه را بدون نیاز به اطلاع از اندازه مجموعه داده فراهم می‌کند. با این حال، استفاده مستقیم از آن برای محاسبه چندک‌ها به فضای حافظه قابل‌توجهی نیاز دارد. الگوریتم نمونه‌گیری تصادفی، که اغلب به عنوان «مرل» MRL شناخته می‌شود را گورمیت سینگ مانکو، سریدهار راجاگوپالان و بروس لیندسی در سال ۱۳۷۸ منتشر کردند و به مسئله نمونه‌گیری صحیح و تخمین چندک پرداخت. این شامل فن نمونه‌گیری غیریکنواخت و الگوریتم یافتن چندک قطعی است.

مانکو و همکاران نسخه‌ای غیریکنواخت از نمونه‌گیری مخزنی را به منظور پشتیبانی از پردازش جریان‌های پیوسته داده با نیاز اندک به حافظه معرفی کردند. در این روش، مقادیری که زودتر در جریان داده ظاهر می‌شوند، نسبت به سایر مقادیر با احتمال بیشتری در نمونه قرار می‌گیرند. این اصلاح، علاوه بر کاهش نیاز به حافظه، دقتی به مراتب بیشتر از روش اصلی نمونه‌گیری مخزنی فراهم می‌کند.

مهم‌ترین نقطه‌ضعف الگوریتم مرل این است که پارامترهای پیکربندی آن از طریق حل مسئله بهینه‌سازی پیچیده تعیین می‌شوند و اجرای آن نیز به چندین رویه پیچیده متکی است. در این بخش، نسخه‌ای ساده‌تر از الگوریتم مرل را بررسی می‌کنیم که گه لو، لو وانگ، که یی، و گراهام کورمودی در سال ۱۳۹۲ پیشنهاد دادند. این الگوریتم در مقالات اصلی با نام تصادف Random معرفی شده است.

الگوریتم تصادف داده‌ها را از جریان داده در قطعات با اندازه متغیر پردازش می‌کند و نمونه‌گیری روی آنها انجام می‌دهد که در نهایت نمونه‌گیری غیریکنواخت را تولید می‌کند.

برای ذخیره نمونه‌های مقادیر، الگوریتم داده‌ساختار را حفظ می‌کند که از b واحد داده ساده $B_1, B_2, \dots, B_b$ تشکیل شده است که بافر نامیده می‌شوند، هر یک از اینها حداکثر k مقدار را ذخیره می‌کنند و می‌توانند با سطح L مرتبط شوند که در آن پر شده‌اند.

پارامتر سطح L احتمال انتخاب مقادیر را منعکس می‌کند و به تعداد مقادیر n که تاکنون پردازش شده‌اند و حداکثر ارتفاع مجاز h درختی که دنباله عملیات انجام شدهٔ الگوریتم را نشان می‌دهد، بستگی دارد:

$$L = L(n,h) = \max\left(0, \left\lceil \log \frac{n}{k2^{h-1}} \right\rceil\right) \quad (1)$$

که در آن $L(0,h) = 0$. برای پر کردن بافر خالی B_i^L، $i \in 0 \dots b$ در سطح L، k مقدار تصادفی را از $k \times 2^L$ مقدار ورودی متوالی، یکی از هر بلوک 2^L، انتخاب می‌کنیم و آنها را در B_i^L ذخیره می‌کنیم. در پایان رویه، بافر ممکن است کمتر از k مقدار داشته باشد زیرا دنباله ورودی به اندازه کافی مقدار نداشته است، اما اگر حداقل یک مقدار در بافر باشد، به عنوان پر برچسب‌گذاری می‌شود.

احتمال انتخاب مقداری خاص از جریان داده ورودی و ذخیره در بافر مستقیما به سطح L بستگی دارد، زیرا اندازه قطعه 2^L را که مقادیر از آن استخراج می‌شوند کنترل می‌کند. این پیاده‌سازی عملی نمونه‌گیری غیریکنواخت مورد استفاده در الگوریتم است.

الگوریتم ۱- پر کردن بافرهای خالی
ورودی: جریان داده D
ورودی: بافر خالی B^L با اندازه k در سطح L
خروجی: بافر پر شده B^L و برچسب آن
برای i از 0 تا $k-1$ انجام بده
  $S \leftarrow next(2^L, D)$// خواندن 2^L مقدار بعدی از D
  اگر $S = \emptyset$ آنگاه
    break
  $x \leftarrow$ نمونه$(s \in S)$// انتخاب تصادفی مقدار از S
  $B^L \leftarrow B^L \cup \{x\}$
خالی ← برچسب
اگر $0 <$ تعداد آن‌گاه(B^L)
  پر ← برچسب
برگرداندن B^L، برچسب

دو بافر از یک سطح L ادغام می‌شود تا فضای بافر بازیابی شود، و ادغام حاصل منجر به بافری جدید با اندازه مشابه در سطح $L+1$ می‌شود. برای ادغام دو بافر، دنباله مقادیر را از هر دو بافر مرتب می‌کنیم و به طور تصادفی نیمی از مقادیر را به عنوان مثال، با انتخاب همه مقادیر در موقعیت‌های زوج یا فرد انتخاب می‌کنیم. بافرهای ادغام شده به عنوان خالی علامت‌گذاری می‌شوند و بافر خروجی به عنوان پر علامت‌گذاری می‌شود.

الگوریتم ۲- ادغام دو بافر غیر خالی
ورودی: بافرهای غیر خالی B_i^L ، B_j^L با اندازه k در سطح L
خروجی: بافر پر شده B^{L+1} در سطح L و برچسب آن
$S \leftarrow sort(B_i^L \cup B_j^L)$
$free(B_i^L)$
$free(B_j^L)$
$B^{L+1} \leftarrow$ نمونه(S,k)// انتخاب تصادفی k مقدار از بافرهای پیوسته
برگرداندن B^{L+1}، پر

عملیات ادغام نیاز به $O(k \log k)$ برای مرتب‌سازی بافرها دارد و پر کردن بافر سطح بعدی در زمان $O(k)$ انجام می‌شود. در نهایت، فرآیند ساخت داده‌ساختار شامل تعدادی مراحل جمعیت بافر و عملیات ادغام است. در ابتدا با بافرهای با برچسب خالی آغاز می‌کنیم. پردازش جریان ورودی با تنظیم سطح فعال L با استفاده از معادله (۱) شروع می‌شود که به دلیل عدم ورود اعضا، مقدار آن در ابتدا برابر با صفر است. اگر بافر خالی B وجود داشته باشد، آن را با استفاده از الگوریتم ۱ با خواندن $k \times 2^L$ مقدار از جریان پر می‌کنیم.

وقتی همه بافرها پر شدند، پایین‌ترین سطح با حداقل دو بافر پر را پیدا می‌کنیم و دو بافر از آن سطح را به طور تصادفی انتخاب شده‌اند ادغام می‌کنیم.

تعداد کل عملیات ادغام در کل جریان داده $O\left(\frac{n}{k}\right)$ است که برای هر به‌روزرسانی حدود $O(1)$ است. مرتب‌سازی برای هر به‌روزرسانی O(log k) زمان می‌برد. بنابراین، زمان سرشکنی O(log k) است.

مثال ۳- ساخت بافرهای نمونه: مجموعه داده‌ای از ۲۵ عدد صحیح {۰,۰,۳,۴,۱,۶,۰,۵,۲,۰,۳,۳,۲,۳,۰,۲,۵,۰,۳,۱,۰,۳,۱,۶,۱} را در نظر می‌گیریم. از ارتفاع h = 3 برای پیش‌برد فرایند مدیریت جریان داده استفاده می‌کنیم و b = 4 بافر B_1، B_2، B_3، B_4 نگهداری می‌شود که هر یک k=4 مقدار دارند. بنابراین، با ساده‌سازی معادله (۱)، سطح فعال برابر است با:

$$L = L(n) = max\,(0, \lceil log\,(n) - 4 \rceil)$$

در ابتدا، تعداد مقادیر پردازش شده n = 0 است، بنابراین از L = 0 شروع می‌کنیم و اولین $N_1 = 4$ مقدار را از جریان ورودی {۰,۰,۳,۴} می‌خوانیم و بافری خالی، مثلا B_1 را پر می‌کنیم. از آنجایی که ظرفیت بافر نیز ۴ است، نیازی به نمونه‌برداری مقادیر تصادفی نداریم و همه آنها ذخیره می‌شوند.

B_1^0				B_2				B_3				B_4			
0	0	3	4												

دوم، دوباره باید سطح فعال را تعریف کنیم. تعداد مقادیر پردازش شده $n = N_1 = 4$ است و سطح فعال صفر می‌ماند: $L = L(4) = max\,(0, 2 - 4) = 0$. ما $N_2 = 4$ مقدار بعدی {۱,۶,۰,۵} را می‌خوانیم و به همین ترتیب، بافر B_2 را پر می‌کنیم.

B_1^0				B_2^0				B_3				B_4			
0	0	3	4	1	6	0	5								

بنابراین، قبلا $n = N_1 + N_2 = 8$مقدار را پردازش کرده‌ایم، سطح فعلی L = L(8) = max (0,3 - 4) = 0 است و ما $N_3 = 4$ مقدار بعدی {۲,۰,۳,۳} را به بافر B_3 اندیس می‌کنیم.

B_1^0				B_2^0				B_3^0				B_4			
0	0	3	4	1	6	0	5	2	0	3	3				

به همین ترتیب، پس از پردازش $n = N_1 + N_2 + N_3 = 12$ عنصر، سطح فعال هنوز صفر است، دوباره $N_4 = 4$ مقدار بعدی {۲,۳,۰,۲} را می‌خوانیم و تنها بافر خالی باقی‌مانده B_2 را پر می‌کنیم.

B_1^0				B_2^0				B_3^0				B_4^0			
0	0	3	4	1	6	0	5	2	0	3	3	2	3	0	2

در این مرحله هیچ بافر خالی باقی نمانده است، بنابراین باید عملیات ادغام را انجام دهیم. پایین‌ترین لایه‌ای که حداقل دو بافر دارد سطح ۰ است که از آن به طور تصادفی دو بافر را انتخاب می‌کنیم، برای مثال، B_2^0 و B_3^0. ابتدا، همه مقادیر را از این بافرها ادغام کرده و مرتب می‌کنیم:

$$\{۱,۶,۰,۵\} \cup \{2,0,3,3\} = \{1,6,0,5,2,0,3,3\} \rightarrow \{0,0,1,2,3,3,5,6\}$$

سپس، بافرهای B_2^0 و B_3^0 را آزاد می‌کنیم و بافر B_3 را در سطح ۱ با ۵۰ درصد از مقادیر قبلی آنها پر می‌کنیم، برای سادگی اجازه دهید مقادیر فرد را بگیریم.

B_1^0				B_2				B_3^1				B_4^0			
0	0	3	4					0	1	3	5	2	3	0	2

بنابراین، قبلا $n = N_1 + N_2 + N_3 + N_4 = 16$ مقدار را پردازش کرده‌ایم، اما لایه فعال صفر می‌ماند، و B_2 را با $N_5 = 4$ مقدار بعدی از جریان داده پر می‌کنیم: {۵,۰,۳,۱}.

B_1^0				B_2^0				B_3^1				B_4^0			
0	0	3	4	5	0	3	1	0	1	3	5	2	3	0	2

بار دیگر هیچ بافر خالی وجود ندارد، بنابراین باید ادغامی دیگر انجام دهیم.

سطح ۰ شامل سه بافر پر است و ما به طور تصادفی دو تای آنها را انتخاب می‌کنیم، به عنوان مثال، B_1^0 و B_4^0، سپس همه مقادیر آنها را ادغام و مرتب می‌کنیم:

$$\{۰,۰,۳,٤\} \cup \{2,3,0,2\} = \{0,0,3,4,2,3,0,2\} \rightarrow \{0,0,0,2,2,3,3,4\}$$

بافرهای B_1^0 و B_4^0 را به عنوان خالی برچسب می‌زنیم و بافر B_4 را در سطح ۱ با ۵۰ درصد از مقادیر آنها با گرفتن مقادیر در موقعیت‌های زوج پر می‌کنیم.

B_1				B_2^0				B_3^1				B_4^1			
				5	0	3	1	0	1	3	5	0	2	3	4

در مرحله بعد، قبلا $n = N_1 + N_2 + N_3 + N_4 + N_5 = 20$ را پردازش کرده‌ایم، بنابراین سطح فعال $L = L(20) = 1$ است، و $4.32 - 4 = 8$ مقدار بعدی را از جریان داده می‌خوانیم. در این مورد، مقادیر کافی در جریان داده باقی نمانده است، $N_6 = 4 \cdot 2^1$ را می‌خوانیم و B_1 را با نمونه‌گیری یک مقدار از هر گروه دو عنصری پر می‌کنیم. {۰,۳,۱,۶,۱} را با نمونه‌گیری یک مقدار از هر گروه دو عنصری پر می‌کنیم.

B_1^1				B_2^0				B_3^1				B_4^1			
3	1	1		5	0	3	1	0	1	3	5	0	2	3	4

با داده‌ساختار می‌توان به پرس‌وجوی چندک معکوس پاسخ داد و رتبه مقدار داده شده x را می‌توان به صورت مجموع وزنی لایه از شمارش مقادیر کوچکتر از x برای هر بافر غیر خالی تخمین زد:

$$rank(x) = \sum_{i=1}^{k} 2^{L(B_i)} \cdot \left|\{e < x : e \in B_i^L(B_i)\}\right| \quad (2)$$

مثال ۴- پرس‌وجوی چندک معکوس با نمونه‌گیری تصادفی: جریان داده از مثال ۳ را در نظر می‌گیریم و پرس‌وجوی چندک معکوس را برای تخمین رتبه مقدار ۴ انجام می‌دهیم. داده‌ساختار به شکل زیر است.

B_1^1				B_2^0				B_3^1				B_4^1			
3	1	1		5	0	3	1	0	1	3	5	0	2	3	4

با استفاده از معادله (۲) رتبه را محاسبه می‌کنیم:

$$rank(4) = 2^1 \times 3 + 2^0 \times 3 + 2^1 \times 3 + 2^1 \times 3 = 21$$

بنابراین، برای مقدار ۴ رتبه تخمینی $rank(4) = 21$ است.

برای پاسخ به پرس‌وجوی چندک و یافتن چندک-q از داده‌ساختار، فقط باید به دنبال مقداری باشیم که رتبه تخمینی آن حاصل شده از معادلهٔ ۲، نزدیک‌ترین مقدار به $q \times n$ باشد. در واقع، باید تعدادی پرس‌وجوی چندک معکوس برای هر یک از مقادیر در داده‌ساختار انجام دهیم، اما می‌توانیم از جستجوی دودویی برای تسریع فرآیند استفاده کرد و به محض یافتن مقداری که به اندازه کافی نزدیک است، متوقف شویم.

مثال ۵- پرس‌وجوی چندک با نمونه‌گیری تصادفی: جریان داده از مثال ۳ را در نظر می‌گیریم و چندک ۰.۶۵ را محاسبه می‌کنیم. تعداد کل مقادیر در داده‌ساختار $n = 25$ است، بنابراین مقدار مرزی $q \times n = 0.65 \times 25 = 16.25$ است.

مقادیر {۰,۱,۲,۳,۴,۵} در داده‌ساختار وجود دارند. با مقدار ۰ شروع می‌کنیم و رتبه آن را تخمین می‌زنیم که طبق معادله (۲) برابر با صفر است: rank(0) = 0. سپس، مقدار ۱ را بررسی می‌کنیم و تخمین رتبه آن rank(1) = 5 است. رتبه مقدار ۲ برابر rank(2)

12 = است، در حالی که rank(3) = 14 است. و قبلا از مثال ۴ می‌دانیم rank(4) = 21. در نهایت، رتبه مقدار ۵ برابر rank(5) = 23 است.
بنابراین، نزدیک‌ترین مقدار به مقدار مرزی ۱۶.۲۵ مقدار ۳ با rank(3) = 14 است. مقدار ۳ را به عنوان تقریبی از چندک ۰.۶۵ گزارش می‌کنیم.

باید توجه داشت که با استفاده از جستجوی دودویی روی مجموعه مرتب شده اعداد، و همچنین با در نظر گرفتن اینکه رتبه تابعی یکنواخت از آرگومان خود است، فرآیند را تسریع بخشید.

ویژگی‌ها

برای محاسبهٔ تقریب-ϵ چندک-q، الگوریتم تصادف به مقدار ثابتی از حافظه نیاز دارد که متناسب با $b \times k$ است و صرفا به ϵ وابسته است. با خطای تقریب داده شده، درخت محاسباتی با ارتفاع $h = \log\frac{1}{\epsilon}$ داشته باشیم و تعداد بهینه بافرها برابر است با

$$b = \log\frac{1}{\epsilon} + 1,$$

و اندازه هر بافر برابر است با

$$k = \frac{1}{\epsilon}\sqrt{\log\frac{1}{\epsilon}}$$

به دلیل ماهیت تصادفی این الگوریتم، تقریب‌های چندک با احتمال خطای ثابتی که به $\frac{\varepsilon}{2}$ محدود می‌شود، گزارش می‌شوند. این خطا ناشی از مراحل نمونه‌گیری تصادفی و ادغام تصادفی است.

تلخیص چ

تلخیص چندک، یا q-digest الگوریتم خلاصه جریان مبتنی بر درخت است که نیشیت شریواستاوا، چیرانجیب بوراگوهین، دیویاکانت آگروال، و سابهاش سوری در سال 1383 برای استفاده در زمینه نظارت بر داده‌های توزیع شده از حسگرها پیشنهاد کردند.

تلخیص چ مسئلهٔ محاسبه چندک را به مثابهٔ مسئله‌ای لحاظ می‌کند که در آن داده‌ها با تعدادی سطل خلاصه می‌شوند. الگوریتم مجموعه‌ای از چنین سطل‌ها را در داده‌ساختار درختی نگه می‌دارد، سطل‌های کوچک را ادغام می‌کند و سطل‌های بزرگ را تقسیم می‌کند. شایان توجه است که الگوریتم مذکور الگوریتمی احتمالی نیست و در واقع الگوریتمی قطعی با اتلاف است.

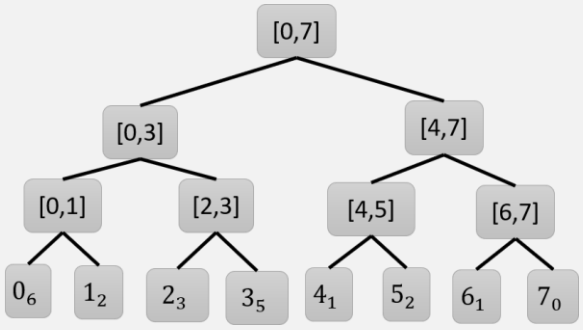
الگوریتم با مقادیر صحیح در بازه‌ای از پیش معین کار می‌کند. تقسیم دودویی بازهٔ اعداد صحیح $[0, N-1]$ را می‌توان به عنوان درخت دودویی کامل و مجازی نشان داد که مقدار ریشه آن با کل بازه [0, N - 1]، فرزندان چپ و راست آن با بازه‌های $\left[0, \left\lfloor\frac{N-1}{2}\right\rfloor\right]$ و $\left[\left\lfloor\frac{N-1}{2}\right\rfloor + 1, N-1\right]$ مطابقت دارند، و گره‌های برگ مقادیر اعداد صحیح منفرد را نشان می‌دهند. عمق درخت log N است.

هر گره v در چنین درخت دودویی سطل است که دارای بازهٔ مرتبط $[v_{min}, v_{max}]$ است. همچنین، هر سطل دارای شمارنده‌ای متناظر v_{count} است و تعداد مقادیر (از جمله تکراری‌ها) را که در آن گره نمایه شده‌اند را نمایش می‌دهد.

مثال ۶- تقسیم‌بندی دودوئی تلخیص چ: مجموعه داده‌ای از n = 20 عدد صحیح از بازهٔ $[۰, ۷]$ را در نظر می‌گیریم که در مثال ۳ بررسی کردیم:

{۰,۰,۳,۴,۱,۶,۰,۵,۲,۰,۳,۳,۲,۳,۰,۲,۵,۰,۳,۱}

بخش‌بندی دودویی بازه، درخت دودوئی زیر را می‌سازد و داده‌های ورودی را سطل‌بندی می‌کند:



گره‌های برگ از چپ به راست مقادیر [0,N - 1] را نشان می‌دهند و اعداد نمایه بسامد مقادیر را در مجموعه داده نشان می‌دهند.

در نتیجه، نمایش داخلی داده‌ها شامل فراوانی‌هایی است که نشان می‌دهند هر مقدار چند بار مشاهده شده است. در بدترین حالت، محدودیت حافظه ایجاب می‌کند که این اطلاعات برای O(n) یا O(N) مقدار، هر کدام که کوچک‌تر باشد، ذخیره شوند. همچنین باید توجه داشت که در عمل، درخت‌های دودویی مذکور معمولاً بسیار تنک و نامتوازن هستند. ازاین‌رو، ذخیره‌سازی مستقیم آن‌ها بدون استفاده از روش‌های فشرده‌سازی، از نظر مصرف حافظه چندان کارآمد نیست.

الگوریتم تلخیص چ راهی برای فشرده‌سازی و ذخیره فشرده چنین درخت پارتیشن دودویی پیشنهاد می‌کند. داده‌ساختار آن اطلاعاتی در مورد توزیع مقادیر رمزگذاری می‌کند و نسخه‌ای از درخت دودویی را نشان می‌دهد که فقط شامل سطل‌هائی است که «خاصیت تلخیص» زیر را برآورده می‌کنند:

$$\begin{cases} v_{count} \le \lfloor n\sigma \rfloor, & \text{به جز سطلهای برگ} \\ v_{count} + v^{p}_{count} + v^{s}_{count} > \left\lfloor \frac{n}{\sigma} \right\rfloor, & \text{به جز ریشه} \end{cases} \quad (3)$$

که در آن v^p والد و v^s همزاد v، n تعداد کل مقادیر، و $\sigma \in [1, n]$ پارامتر طراح و مشخص کنندهٔ میزان و سطح فشرده‌سازی است. استثنا از این ویژگی، سطل‌های ریشه و برگ هستند. سطل ریشه می‌تواند خاصیت تلخیص (۳) را نقض کند، ولی با این وجود، همچنان در داده‌ساختار گنجانده می‌شود. سطل‌های برگ با شمارش‌های بزرگتر از مقدار مرزی $\left\lfloor \frac{n}{\sigma} \right\rfloor$ (مقادیر پربسامد) نیز گنجانده می‌شوند.

در واقع، خاصیت تلخیص نوعی تعادلی بین گنجاندن چند سطل بزرگ، و بسیاری از سطل‌های کوچک حاوی اطلاعات چند مقدار کم‌بسامد برقرار می‌کند.

به بیان ساده، نخستین محدودیت در ویژگی تلخیص (۳) باعث می‌شود همه سطل‌ها، به‌جز گره‌های برگ، حذف شوند. تنها استثنا، گره‌های برگی هستند که شمارش مقادیر پربسامد را در خود نگه می‌دارند، زیرا در این حالت نگهداری مقادیر فرزند و برخورداری از شمارنده‌های دقیق‌تر ارزشمند است.

از سوی دیگر، بر اساس محدودیت دوم، اگر دو سطل مجاور که با یکدیگر همزاد هستند، شمارش‌های کمی داشته باشند، نگهداری دو شمارنده مجزا برای آن‌ها توجیهی ندارد. در چنین شرایطی، مناسب‌تر است این دو سطل در گره والد ادغام شوند تا ضمن حفظ اطلاعات، میزان فشرده‌سازی مورد نظر نیز حاصل شود.

در نتیجه، برای حفظ این ساختار، لازم است سطل‌ها به‌صورت سلسله‌مراتبی ادغام و تعداد آن‌ها کاهش یابد. این فرایند با پیمایش همه سطل‌ها از پایین به بالا انجام می‌شود و در هر مرحله نقض شدن خاصیت تلخیص برای سطل مدنظر بررسی می‌شود. در عمل، این پیمایش تنها از پایین به بالا انجام می‌شود، در نتیجه بررسی محدودیت دوم کفایت می‌کند.

به جز سطل ریشه، برای هر سطل v که خاصیت تلخیص را نقض می‌کند، زیردرخت آن را با فشرده‌سازی شمارش‌ها از آن، والد v^p و همزاد v^s ادغام و به سطل والد ارتقا می‌یابند:

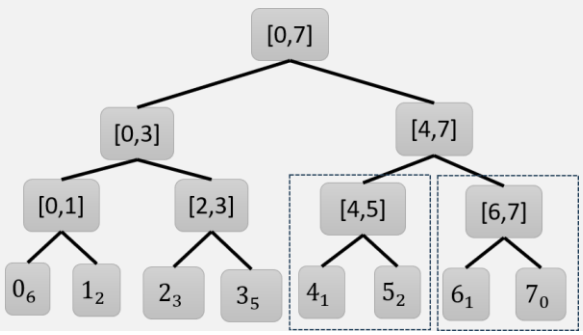
$$v^p_{count} = v^p_{count} + v_{count} + v^s_{count},$$

و سطل v و همزاد v^s را از ساختار حذف می‌شود.

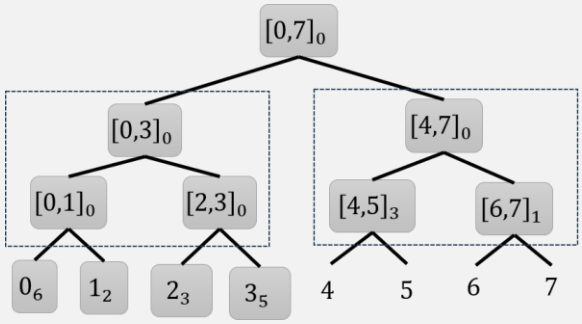
الگوریتم ۴- فشرده‌سازی تلخیص چ
ورودی: داده‌ساختار q-digest بازهٔ [0, N - 1]
ورودی: ضریب فشرده‌سازی σ
خروجی: داده‌ساختار q-digest فشرده شده
سطح $\leftarrow log\ N\ -\ 1$
تازمانی که (سطح > ۰) انجام بده
برای [سطح]$Q - DIGEST$ $v \in$ انجام بده
اگر $v_{count} + v^p_{count} + v^s_{count} \leq \left\lfloor \frac{n}{\sigma} \right\rfloor$ آنگاه
$v^p_{count} \leftarrow v^p_{count} + v_{count} + v^s_{count}$
$Q - DIGEST \leftarrow Q - DIGEST \setminus \{v, v^s\}$
سطح $\leftarrow$ سطح $-$ ۱
برگرداندن Q-DIGEST

فشرده‌سازی دارای زمان مجانبی $O(m \cdot log\ N)$ است، که در آن $m = |Q - DIGEST|$ تعداد سطل‌ها در داده‌ساختار است. بنابراین، هزینه به‌روزرسانی نظری به ازای هر مقدار حدود O(log N) است. با این حال، در عمل، اجرای عملیات به‌روزرسانی زمان بیشتری نیاز دارد، زیرا هر مقدار ابتدا در گره برگ درج می‌شود. سپس، در فرایند فشرده‌سازی، الگوریتم با جابه‌جا کردن تدریجی آن به سمت بالا، جایگاه مناسب آن را در ساختار پیدا می‌کند..

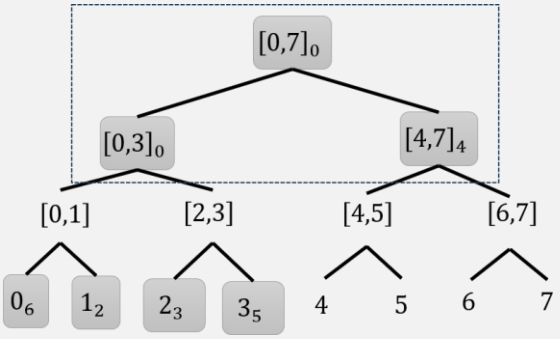
مثال ۷- فشرده‌سازی درخت با تلخیص چ: مجموعه داده با n = 20 مقدار از مثال ۶ را در نظر می‌گیریم که در آن مقدار بسامد سطل‌های مشاهده نشده به طور پیش‌فرض صفر هستند.



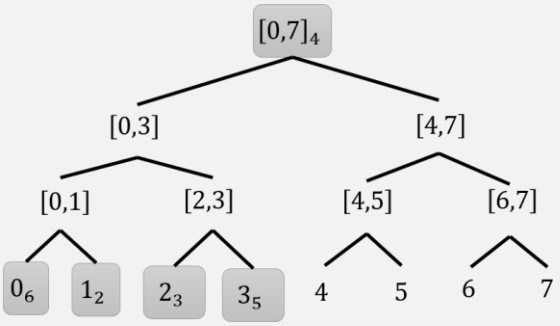
در صورتی که با $\sigma = 5$ فشرده‌سازی صورت پذیرد، مقدار مرزی برابر با $\left\lfloor \frac{n}{\sigma} \right\rfloor = \left\lfloor \frac{20}{5} \right\rfloor = 4$ است. با پیمایش از پایین به بالا، ابتدا سطح چهارم را در نظر می‌گیریم، که در آن فقط سطل‌های ۰ تا ۳ شرط دوم خاصیت تلخیص (۳) را برآورده می‌کنند. با توجه به الگوریتم ۳، فرزندان سطل‌های [۴, ۵] و [۶, ۷] که با هم ناقض خاصیت تلخیص‌اند، در والدین خود ادغام و از ساختار حذف می‌شوند. بنابراین، ساختار در این مرحله عبارت است از (سطل‌های موجود در جعبه‌های خط توپر در داده‌ساختار فشرده گنجانده شده‌اند):



علاوه بر این، در سطح سوم، همه سطل‌ها محدودیت‌های (۳) را نقض می‌کنند. بنابراین، آنها نیز در والدین خود فشرده و در ساختار قرار نمی‌گیرند:



در سطح دوم، با بررسی خاصیت تلخیص دو فرزند سطل ریشه، مشخص می‌شود دوباره محدودیت‌های (۳) را نقض می‌کنند. زیرا تعداد کل آنها از مقدار مرزی تجاوز نمی‌کند و در نتیجه، باید در والد ادغام شوند. همانطور که پیشتر ذکر شد، بررسی خاصیت تلخیص مقدار ریشه ضروری نیست، زیرا در صورت داشتن شمارش‌های غیرصفر در ساختار فشرده گنجانده می‌شود. از این رو، نسخه نهایی داده‌ساختار فشرده به شرح زیر است:



بنابراین، این مثال به داده‌ساختار فشرده با ذخیره فقط پنج سطل با شمارش‌های غیرصفر نیاز دارد.

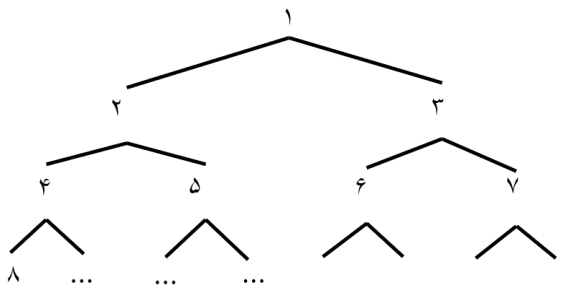
به دلیل پیمایش از پایین به بالا، بررسی خاصیت تلخیص و تصمیم در مورد ادغام سطل‌ها فقط یک بار در طول رویه صورت می‌گیرد، در نتیجه لازم نیست که همه سطل‌های تلخیص چ فشرده پس از فشرده‌سازی خاصیت تلخیص را برآورده کنند. برای مثال، تغییراتی مانند ادغام در والد در سطل‌های سطوح بالای درخت می‌تواند باعث شود سطل‌های قبلا گنجانده شده محدودیت‌های خاصیت

تلخیص (۳) را نقض کنند. با این حال، در عمل، دقت الگوریتم را کاهش نمی‌یابد و در بدترین حالت، داده‌ساختار با بهینگی کمتری ایجاد می‌شود که حافظه بیشتری نسبت به میانگین و انتظار نظری مصرف می‌کند. سخن کوتاه، الگوریتم تلخیص چ برای مجموعه داده ورودی دلخواه به صورت زیر است.

الگوریتم ۵- الگوریتم تلخیص چ
ورودی: مجموعه داده D با مقادیری از بازهٔ [0, N - 1]
ورودی: ضریب فشرده‌سازی σ
خروجی: داده‌ساختار تلخیص چ فشرده
$Q - DIGEST \leftarrow BinaryPartitionTree(D, [0, N - 1])$
برگرداندن $Compress(Q - DIGEST, N, \sigma)$

می‌توان بهینه‌سازی بیشتری در نمایش داده‌ساختار لحاظ کرد. سطل‌ها در درخت دودوئی را از چپ به راست، و از بالا به پایین شماره‌گذاری کرد.

تصویر ۱- شماره‌گذاری سطل‌ها



با شماره‌گذاری همهٔ سطل‌ها ν، بازیابی بازهٔ متناظر $[\nu_{min}, \nu_{max}]$ با دانستن نمایهٔ i به راحتی صورت می‌پذیرد.

الگوریتم ۶- بازیابی محدوده سطل $\left[\nu_{\text{بیش}}, \nu_{\text{کم}}\right]$
ورودی: نمایهٔ سطل i
خروجی: بازهٔ سطل
سطح $\leftarrow \lfloor log(i) \rfloor$
$n \leftarrow 2^{\text{سطح}}$ // تعداد سطل‌ها در سطح
$m \leftarrow i - n$ // موقعیت سطل در سطح
برگرداندن $\left[\frac{(N+1)m}{n}\right], \left[\frac{(N+1)(m+1)}{n}\right] - 1$

سخن کوتاه، از داده ساختار تلخیص چ نمایشی خطی از آرایه‌ای از سطل‌ها ایجاد می‌شود که هر سطل صرفا دوتائی شماره و شمارنده متناظر است. برای مثال، تلخیص چ فشردهٔ مثال ۷ نمایش خطی $\langle (1,4), (8,6), (9,2), (10,3), (11,5) \rangle$ را دارد.

با اتحاد مجموعه سطل‌های ذخیره شده و جمع شمارنده‌های سطل‌های متناظر و اجرای دوبارهٔ الگوریتم فشرده‌سازی، دو تلخیص چ با ضریب فشرده‌سازی σ و محدوده‌های مقدار یکسان را می‌توان ادغام کرد، درنتیجه امکان پردازش جریان‌های داده بزرگ را به صورت توزیع شده فراهم می‌کند.
الگوریتم تلخیص چ در یافتن پاسخ پرس‌وجوی چندک و یافتن چندک-q کاربرد دارد. در ابتدا، دنبالهٔ مرتب S را از طریق مرتب‌سازی صعودی سطل‌ها بر اساس مقادیر ν_{max} متناظر به دست می‌آید. سپس، دنبالهٔ S از ابتدا پویش و شمارنده‌های سطل‌ها به ترتیب

پیمایش اضافه می‌شوند. به محض، بزرگتر شدن مقدار مجموعه از qn در سطل ν^*، سطل مذکور به عنوان ν^*_{max} و تخمین چندک-q گزارش می‌شود.

الگوریتم ۷- پاسخ به پرس‌وجوهای چندک با تلخیص چ
ورودی: داده‌ساختار تلخیص چ
ورودی: مقدار $q \in [0,1]$
خروجی: چندک-q
$S \leftarrow sort(Q - DIGEST)$
$rank \leftarrow 0$
برای $(\nu, count) \in S$ انجام بده
$rank \leftarrow rank + count$
اگر $rank \geq q \cdot n$ آنگاه
$v_{max} = v$
برگرداندن v_{max}

حداقل qn سطل وجود دارد که مقادیر حداکثر آنها کمتر از ν^*_{max} است، بنابراین رتبه سطل ν^* حداقل qn است.
اگر مقادیر کمتر از ν^*_{max} در اجداد سطل ν^* وجود داشته باشند، آن‌گاه در الگوریتم ۷ شمارش نمی‌شوند و در نتیجه امکان خطا در محاسبه تقریب-ϵ چندک-q وجود دارد. چنین خطایی با ϵn محدود می‌شود و الگوریتم رتبه‌ای را در بازه $[qn, (q+\epsilon)n]$ گزارش می‌دهد. بنابراین، هرگز مقدار دقیق چندک-q کمتر از حد تخمین زده نمی‌شود.

مثال ۸- پرس‌وجوی چندک با تلخیص چ- پرس‌وجوی چندک را برای محاسبه چندک ۰.۶۵ از داده‌ساختار ایجادی در مثال ۷ انجام می‌دهیم که نمایش خطی آن به شکل زیر است:

$$\langle (1,4), (8,6), (9,2), (10,3), (11,5) \rangle$$

دنباله مرتب شده سطل‌ها بر اساس بیشینه بازه هر راس برابر فهرست زیر است:

$$S = \langle (8,6), (9,2), (10,3), (11,5), (1,4) \rangle$$

با توجه به الگوریتم، با رفتن از ابتدا، شمارنده‌های سطل‌ها را جمع می‌کنیم تا زمانی که مجموع از $n = 13 \times ۰.۶۵$ بزرگتر شود. در ساختار فعلی، از مقدار مرزی در سطل (۱۱،۵) که مربوط به مقدار برگ ۳ است، فراتر می‌رویم. بنابراین، تخمین تلخیص چ از چندک ۰.۶۵ (یا صدک شصت و پنجم) مجموعه داده مثال ۷ مقدار ۳ است.

به روشی مشابه می‌توان به پرس‌وجوی چندک معکوس پرداخت. دنباله مرتب شده S از سطل‌ها می‌سازیم و از ابتدا در حالی که مجموع جاری شمارنده‌های سطل‌های مشاهده شده را نگه‌داری می‌کنیم، آن را پیمایش می‌کنیم. رتبه مقدار داده شده x را می‌توان به عنوان مجموع شمارش‌های سطل‌ها ν که برای آنها $x > \nu_{max}$ است، تخمین زد.

الگوریتم ۸- پاسخ به پرس‌وجوهای چندک معکوس با تلخیص چ
ورودی: مقدار x
ورودی: داده‌ساختار تلخیص چ
خروجی: رتبه عنصر
$S \leftarrow sort(Q - DIGEST)$
رتبه $\leftarrow 0$

برای $(v, count) \in S$ انجام بده
اگر $x > v_{max}$ آنگاه
رتبه $\leftarrow$ رتبه $+ count$
برگرداندن رتبه

مانند پرس‌وجوی چندک در بالا، رتبه به دست آمده در اینجا در بازه $[rank(x), rank(x) + \epsilon \cdot n]$ قرار دارد. همانطور که قبلا اشاره شد، برای پاسخ به پرس‌وجوی بازه، دو پرس‌وجوی چندک معکوس برای یافتن رتبه‌ها و تفاوت بین مرزهای محدوده a و b اعمال می‌شود. حداکثر خطا برای پرس‌وجوی محدوده در تلخیص چ را می‌توان به صورت $2\epsilon \times n$ تخمین زد.

الگوریتم ۹- پاسخ به پرس‌وجوهای بازه با تلخیص چ
ورودی: بازهٔ [a, b]
ورودی: داده‌ساختار تلخیص چ
خروجی: تعداد مقادیر در محدوده
$r_a \leftarrow InverseQuantileQuery(a, Q - DIGEST)$
$r_b \leftarrow InverseQuantileQuery(b, Q - DIGEST)$
برگرداندن $r_b - r_a$

ویژگی‌ها

الگوریتم تلخیص چ الگوریتمی با اتلاف است و اطلاعات مربوط به مقادیر با بسامد پایین را فشرده و اطلاعات مربوط به مقادیر با بسامد بالا را به دقت حفظ می‌کند. بنابراین، طرح تقریب با کیفیت بالا را هنگامی ارائه می‌کند که تغییرات زیادی در بسامد مقادیر مختلف وجود نسبت به هم وجود داشته باشد. الگوریتم تلخیص چ اطلاعاتی در مورد توزیع مقادیر مقادیر را معلوم می‌کند اما اطلاعاتی در مورد جای آنها در اختیار نمی‌نهد. ضریب فشرده‌سازی σ تعادل بین دقت الگوریتم و حافظه مورد نیاز برای ذخیره داده‌ساختار تلخیص چ را کنترل می‌کند. بنابراین، برای بازهٔ داده شده $[0, N-1]$، انتظار حداکثر $30 \cdot \sigma$ سطل ذخیره شده وجود دارد و خطا در محاسبهٔ تقریب-ϵ چندک-q با کران بالای زیر محدود می‌شود:

$$\epsilon \leq \frac{\log N}{\sigma}$$

تلخیص چ با داده‌ها سازگار است و سطل‌هایی با وزن تقریبا مساوی می‌سازد. در مقابل هیستوگرام معمول، تلخیص چ امکان همپوشانی سطل‌ها را می‌دهد که پاسخ به پرس‌وجوهای اجماع (به عنوان مثال، مقادیر پربسامد) را ممکن می‌سازد.

از مشکلات اصلی در کاربرد عملی الگوریتم تلخیص چ این است که فقط می‌تواند مقادیر عدد صحیح را مدیریت کند، نیاز به دانستن بازه مجموعه داده را از قبل دارد و برای چندک‌های دو سر طیف دارای خطائی قابل توجهی است.

تلخیص د

یکی از روش‌های جانشین برای انباشت دقیق آمار مبتنی بر رتبه به صورت آنلاین، تلخیص د t-digest نام دارد که تد دانینگ و اوتمار ارتل در سال ۱۳۹۳ پیشنهاد کردند. الگوریتم تلخیص د امکان تخمین چندک‌ها را در جریان‌های نامحدود با تمرکز بر مقادیر بزرگ یا دو سر طیف مانند چندک ۰.۹۹ فراهم می‌کند.

تلخیص د جریان داده ورودی D را در خوشه‌هایی با اندازه متغیر $\{C_i\}_{i=1}^{m}$ خلاصه می‌کند و در نتیجه اجازه می‌دهد دقت خوبی در محاسبه چندک در حین پردازش حجم زیادی از داده حفظ کند. هر چنین خوشه C_i زیرمجموعه‌ای از مقادیر ورودی را نشان می‌دهد و اندازه آن به گونه‌ای است که اطمینان حاصل شود برای تخمین چندک‌ها با درون‌یابی خیلی بزرگ نباشد، اما برای جلوگیری از داشتن خوشه‌های زیاد خیلی کوچک نباشد.

هر خوشه C_i دارای مرکز جرم c_i یا نقطه داده‌ای در مرکز خوشه است که میانگین مقادیر ورودی است، و تعداد مقادیر مذکور با c_i^{count} نمایش داده می‌شود. داده‌ساختار تلخیص د آرایه‌ای از چنین مراکز جرم وزنی $\{(c_1, c_1^{count}), (c_2, c_2^{count}), \dots, (c_m, c_m^{count})\}$ است که به ترتیب صعودی مرتب شده‌اند. از این دنباله مرتب شده می‌توانیم حداکثر مقدار چندک را که با هر مرکز جرم c_i مطابقت دارد تخمین زد:

$$q(c_i) = \frac{1}{n}\sum_{j<i} c_j^{count} + \frac{1}{n} \times c_i^{count} \qquad (۴)$$

که در آن $n = \sum_{j=1}^{m} c_j^{count}$ تعداد کل مقادیر نمایه شده در داده‌ساختار است. بنابراین، با توجه به معادلهٔ (۴)، هر خوشه C_i در داده‌ساختار تلخیص د مسئول بازهٔ خاصی از مقادیر چندک $[q(c_{i-1}), q(c_i)]$ است که طول آن به اندازه خوشه، یا تعداد عناصری که به این خوشه تعلق دارند، بستگی دارد. تعیین اندازه صحیح خوشه تأثیر مستقیمی بر دقت دارد و در الگوریتم تلخیص د با تابع مقیاس غیرنزولی مشخص می‌شود. چنین تابعی $k = k(q, \sigma)$ فشرده‌سازی مطلوب σ را در نظر می‌گیرد و مقادیر چندک q را بر اساس اینکه چقدر از انتها مانند $q = 0$ و $q = 1$ دور هستند، به طور متفاوت مقیاس می‌دهد. انتخاب مناسب تابع مقیاس بسیار مهم است و توابع با مقیاس‌بندی متفاوت از لحاظ دقت وجود دارد. برای مثال، یکی از توابع رایج استفاده شده عبارت است از

$$k(q, \sigma) = \frac{\sigma}{2\pi} arcsin\,(2q - 1) \qquad (۵)$$

که در آن پارامتر فشرده‌سازی $\sigma > 1$ (مقادیر بزرگتر منجر به فشرده‌سازی کمتر است). با توجه به تابع مقیاس انتخاب شده $k = k(q, \sigma)$ برای هر خوشه C_i با مرکز جرم c_i اندازه k (نمایش داده شده با $K(c_i)$)را برابر طول مقیاس‌گذاری شده محدوده چندک خوشه به صورت زیر تعریف می‌کنیم:

$$K(c_i) = k(q(c_i), \sigma) - k(q(c_{i-1}), \sigma), i = 2 \dots m \qquad (۶)$$

که در آن $K(c_1) = k(q(c_1), \sigma)$. برای محدود کردن تعداد مقادیر موجود در هر خوشه، به‌گونه‌ای که این تعداد به بازه چندکی متناظر با آن خوشه وابسته باشد، می‌توان اندازه خوشه را با پارامتر k کنترل کرد. با استفاده از توابع مقیاس غیرخطی، خوشه‌هایی با اندازه‌های غیریکنواخت ایجاد می‌شوند. در این حالت، خوشه‌های متناظر با چندک‌های میانی بزرگ‌تر هستند، در حالی که هرچه به دو انتهای توزیع نزدیک‌تر شویم، اندازه خوشه‌ها کوچک‌تر می‌شود و حتی ممکن است به خوشه‌هایی برسیم که صرفا یک مقدار را در خود جای می‌دهند. افزون بر این، الگوریتم تلخیص د به‌گونه‌ای طراحی شده است که ساختار داده تلخیص د کاملاً ادغام‌شده ایجاد کند. به عبارت دیگر، همه خوشه‌های C_j، برای $j = 1, \dots, m$، ویژگی تلخیص را برآورده می‌کنند.

$$\begin{cases} K(c_i) \leq 1, & \text{به جز خوشه‌های تکی} \\ K(c_i) + K(c_{i+1}) > 1 \end{cases} \qquad (۷)$$

این ویژگی نه‌تنها اندازه هر خوشه را به k محدود می‌کند، بلکه تضمین می‌کند که هیچ دو خوشه مجاوری را نتوان بدون نقض این محدودیت با یکدیگر ادغام کرد. در عمل، برای برقرار ماندن محدودیت‌های ویژگی تلخیص (۷)، نیازی نیست پس از هر تغییر، مقدار k برای همه خوشه‌ها دوباره محاسبه شود. در عوض، چون همه مراکز جرم در تلخیص د به‌صورت مرتب نگهداری می‌شوند، می‌توان تعداد مقادیر موجود در خوشه C_i را با تعیین کران برای بیشترین مقدار چندکِ برآوردشده آن خوشه محدود کرد.

$$q_{limit} = k^{-1}(k(q(c_i), \sigma) + 1, \sigma) \qquad (۸)$$

که برای تابع مقیاس (۵) به شکل زیر است:

$$q_{limit} = \frac{1}{2}\left[1 + sin\left(\arcsin(2 \cdot q(c_i) - 1) + \frac{2\pi}{\sigma}\right)\right] \quad (۹)$$

با در اختیار داشتن رابطه (۸) برای محدود کردن تعداد مقادیر در هر خوشه در تلخیص د، الگوریتم ادغام تلخیص د را می‌توان مطابق الگوریتم ۹ صورت‌بندی کرد که از نظر عملکرد مشابه رویه خوشه‌بندی معمولی است. برای خلاصه‌سازی دنباله ورودی شامل نقاط داده وزنی $\{(x_1, x_1^{count}), (x_2, x_2^{count}), \dots, (x_b, x_b^{count})\}$، این نقاط به همراه همه مراکز جرم موجود در ساختار داده تلخیص د مرتب می‌شوند. سپس با یک بار پیمایش دنباله حاصل X، تلاش می‌شود مراکز جرم تا جایی که ویژگی تلخیص نقض نشود، به‌صورت متوالی با یکدیگر ادغام شوند.

فرایند با چپ‌ترین مرکز جرم آغاز می‌شود. خوشه متناظر با آن به عنوان خوشه نامزد انتخاب می‌شود و مقدار مرزی q_{limit} با استفاده از رابطه (۸) محاسبه می‌شود. در ادامه، همه مراکز جرم موجود در X به ترتیب پردازش می‌شوند، مقدار چندک تقریبی هر یک برآورده می‌شود و با مقدار مرزی مقایسه می‌گردد. اگر افزودن مرکز جرم در انتظار باعث تجاوز از مقدار مرزی نشود، آن مرکز جرم در خوشه نامزد ادغام می‌شود و پردازش با مرکز جرم بعدی ادامه می‌یابد. در غیر این صورت، این وضعیت نشان‌دهنده آن است که خوشه نامزد به حداکثر ظرفیت مجاز خود رسیده و دیگر امکان افزودن مقادیر جدید به آن وجود ندارد. در این حالت، خوشه نامزد در ساختار داده تلخیص د ذخیره می‌شود، سپس، خوشهٔ نامزد جدیدی با مرکز جرم در انتظار تشکیل می‌شود و مقدار مرزی چندک q_{limit} دوباره محاسبه می‌شود. در پایان این فرایند، یک ساختار داده تلخیص د کاملاً ادغام‌شده حاصل می‌شود.

الگوریتم ۱۰- ادغام مقادیر در تلخیص ت

ورودی: بافر B با مقادیر $\{(x_1, x_1^{count}), (x_2, x_2^{count}), \dots (x_b, x_b^{count})\}$

ورودی: داده‌ساختار تلخیص د

ورودی: پارامتر فشرده‌سازی $\sigma > 1$، تابع مقیاس k

خروجی: داده‌ساختار تلخیص د با بافر ادغام شده

$X \leftarrow sort(T - DIGEST \cup B)$

$T - DIGEST \leftarrow \emptyset$

$m \leftarrow count(X), n \leftarrow \sum_{x_i \in X} x_i^{count}$

$c \leftarrow x_1, q_c \leftarrow 0$

$q_{limit} \leftarrow k^{-1}(k(q_c, \sigma) + 1, \sigma)$

برای i از 2 تا m انجام بده

$\hat{q} \leftarrow q_c + \frac{1}{n} c^{count} + \frac{1}{n} \cdot x_i^{count}$

اگر $\hat{q} \leq q_{limit}$ آنگاه

$c^{count} \leftarrow c^{count} + x_i^{count}$

$c \leftarrow c + x_i^{count} \cdot \frac{x_i - c}{c^{count} + x_i^{count}}$

continue

$T - DIGEST \leftarrow T - DIGEST \cup \{(c, c^{count})\}$

$q_c \leftarrow q_c + \frac{1}{n} c^{count}$

$q_{limit} \leftarrow k^{-1}(k(q_c, \sigma) + 1, \sigma)$

$c \leftarrow x_i$

$T - DIGEST \leftarrow T - DIGEST \cup \{(c, c^{count})\}$// آخرین خوشه معمولا تکی

برگرداندن T-DIGEST

همان‌گونه که مشاهده می‌شود، با تشکیل هر خوشه نامزد جدید، مقدار مرزی q_{limit} باید دوباره محاسبه شود. این محاسبه، بر اساس روابط (۸) و (۹)، مستلزم ارزیابی تابع مقیاس و معکوس آن است که از نظر محاسباتی پرهزینه به شمار می‌رود. با این حال، از آنجا

که تعداد خوشه‌ها در عمل معمولاً زیاد نیست، این هزینه قابل مدیریت است. افزون بر این، روش‌های مختلفی برای کاهش هزینه این محاسبات پیشنهاد شده‌اند که از جمله آن‌ها می‌توان به استفاده از تقریب‌های کارآمد برای اجزای تابع مقیاس و نیز برآورد تقریبی حداکثر تعداد عناصری که می‌توان در هر خوشه خلاصه کرد، اشاره کرد.
الگوریتم کامل نمایه‌گذاری جریان پیوسته داده بر این اساس است که داده‌های ورودی ابتدا در بافرهایی با اندازه ثابت جمع‌آوری شوند و سپس این بافرها به‌طور پیوسته در ساختار داده تلخیص د ادغام شوند.

الگوریتم ۱۱- اندیس کردن جریان داده با تلخیص ت
ورودی: جریان داده $D = \{x_1, x_2, \dots\}$
ورودی: اندازه بافر b، پارامتر فشرده‌سازی $\sigma > 1$، تابع مقیاس k
ورودی: داده‌ساختار تلخیص د
$T - DIGEST \leftarrow \emptyset$
تازمانی که D انجام بده
$B \leftarrow \{(x_1, 1), (x_2, 1), \dots, (x_b, 1)\}$
$T - DIGEST \leftarrow Merge(T - DIGEST, B, \sigma, k)$
برگرداندن T-DIGEST

باید توجه باشد که هزینه اجرای الگوریتم بافر-و-ادغام ۱۱ به دو بخش توزیع می‌شود. بخش نخست، درج مکرر مقادیر ورودی در بافر است و بخش دوم، فراخوانی‌های گهگاه الگوریتم ۱۰ برای ادغام محتویات بافر در ساختار داده است. از آنجا که عملیات درج در بافر بسیار کم‌هزینه است، بخش عمده هزینه اجرا به مرتب‌سازی داده‌ها و فراخوانی‌های تابع مقیاس در الگوریتم ادغام مربوط می‌شود. با این حال، از آنجا که هر بار اجرای الگوریتم ادغام پس از چندین عملیات درج انجام می‌شود، این هزینه میان تعداد زیادی از درج‌ها سرشکن شده و در نتیجه، هزینه متوسط هر درج پایین باقی می‌ماند.

مثال ۹- نمایه‌سازی جریان داده با تلخیص د: مجموعه داده با n = 20 عدد صحیح از مثال ۶ را در نظر می‌گیریم:
{۰,۰,۳,۴,۱,۶,۰,۵,۲,۰,۳,۳,۲,۳,۰,۲,۵,۰,۳,۱}
پارامتر فشرده‌سازی $\sigma = 5$، اندازه بافر b = 10 و تابع مقیاس (۵) را در نظر می‌گیریم. بافر B را با ده مقدار اول از ورودی پر می‌کنیم:
$$B = \langle (0,1), (0,1), (3,1), (4,1), (1,1), (6,1), (0,1), (5,1), (2,1), (0,1) \rangle$$
با توجه به الگوریتم ۱۱، باید مقادیر را از بافر و مراکز جرمی که در تلخیص د هستند، بپیوندیم. با این حال، چون داده‌ساختار تلخیص د خالی است، فهرست نامزدهای مراکز جرم X فقط شامل n = 10 مقدار از B است که آنها را به ترتیب صعودی مرتب می‌کنیم:
$$X = \langle (0,1), (0,1), (0,1), (0,1), (1,1), (2,1), (3,1), (4,1), (5,1), (6,1) \rangle,$$
اولین خوشه نامزد را با خواندن چپ‌ترین مرکز جرم (۰,۱) انتخاب می‌کنیم و مقدار مرزی چندک آن q_{limit} را با استفاده از معادله (۹) محاسبه می‌کنیم:
$$q_{limit} = \frac{1}{2}\left[1 + sin\left(arcsin(-1) + \frac{2\pi}{5}\right)\right] = 0.34549$$
سپس، مقدار بعدی را از X می‌گیریم که دوباره (۰,۱) است و ارزیابی می‌کنیم که آیا می‌توان آن را بدون نقض خاصیت تلخیص در خوشه نامزد ادغام کرد. در مورد ما، برای تخمین حداکثر مقدار چندک $\hat{q}$ این خوشه ادغام شده نیاز دارد:

$$\hat{q} = \frac{۱ + ۱}{۱۰} = ۰.۲,$$

چون مقدار آن از مقدار مرزی چندک $\hat{q}$ کمتر است، می‌توانیم آزادانه مقدار (۰,۱) را در خوشه نامزد فعلی ادغام کنیم، که اکنون دارای $c^{count} = 2$ مقدار است اما، از آنجایی که مقادیر یکسان هستند، مرکز جرم ثابت می‌ماند c = 0.

به طور مشابه، می‌توانیم مقدار بعدی (۰،۱) را در خوشه نامزد ادغام کنیم که فقط تعداد مقادیر خلاصه شده را به $c^{count} = 3$ تغییر می‌دهد.

سپس، چهارمین مقدار را از X می‌گیریم که (۰،۱) است و مانند روش بالا برای بررسی اینکه آیا می‌تواند در خوشه نامزد نیز ادغام شود، عمل می‌کنیم. با این حال، حداکثر چندک تخمینی $\hat{q}$ چنین خوشه ادغام شده به صورت زیر خواهد بود:

$$\hat{q} = \frac{۳ + ۱}{۱۰} = ۰.۴,$$

که از مقدار مرزی فعلی $q_{limit} = 0.34549$ فراتر می‌رود. بنابراین، تلاش‌های خود را برای جذب سایر مراکز جرم توسط خوشه نامزد متوقف می‌کنیم، آن را در داده‌ساختار تلخیص د ذخیره می‌کنیم

$$T - DIGEST = \langle (0,3) \rangle,$$

و حداکثر مقدار چندک آن را به عنوان $q = \frac{c^{count}}{n} = 0.3$ به خاطر می‌سپاریم.

از این مرحله، خوشه نامزد جدید از مقدار (0,1) تشکیل می‌شود، به‌طوری‌که c = 0 و $c^{count} = 1$ است. سپس مقدار مرزی چندک متناظر با این خوشه محاسبه می‌شود.

$$q_{limit} = \frac{1}{2}\left[1 + sin\left(arcsin\,(2 \times 0.3 - 1) + \frac{2\pi}{5}\right)\right] = 0.874025$$

سپس، مقدار (۱،۱) را می‌گیریم و اگر آن را در خوشه نامزد فعلی ادغام کنیم، حداکثر چندک تخمینی $\hat{q}$ خواهد بود:

$$\hat{q} = 0.3 + \frac{۱ + ۱}{۱۰} = ۰.۵,$$

که از مقدار مرزی فعلی تجاوز نمی‌کند. بنابراین، مقدار (۱،۱) را در خوشه فعلی ادغام می‌کنیم که تعداد آن به $c^{count} = 2$ افزایش می‌یابد و مرکز جرم می‌شود:

$$c = 0 + \frac{1 - 0}{2} = 0.5$$

به طور مشابه، بقیه مقادیر را از X پردازش می‌کنیم و داده‌ساختار تلخیص د به موارد زیر رشد می‌کند:

$$T - DIGEST = \langle (0,3), (2,5), (5,1), (6,1) \rangle$$

با ادامه پردازش مجموعه داده، بافر جدید پر می‌کنیم:

$$B = \langle (3,1), (3,1), (2,1), (3,1), (0,1), (2,1), (5,1), (0,1), (3,1), (1,1) \rangle,$$

آن را با داده‌ساختار تلخیص د پیوند می‌دهیم و دنباله حاصل X را به ترتیب صعودی مراکز جرم مرتب می‌کنیم:

$$X = \langle (0,3), (0,1), (0,1), (1,1), (2,5), (2,1), (2,1), (3,1), (3,1), (3,1), (3,1), (5,1), (5,1), (6,1) \rangle$$

با داشتن آن، داده‌ساختار تلخیص د را تخلیه می‌کنیم و از چپ‌ترین مقادیر شروع می‌کنیم و سعی می‌کنیم مقادیر را به طور متوالی ادغام کنیم و خوشه‌ها را در تلخیص د ذخیره کنیم تا دیگر نتوان آنها را ادغام کرد. در پایان، داده‌ساختار تلخیص د حاصل از m = 5 خوشه تشکیل شده است و نمای زیر را دارد:

$$T - DIGEST = \langle (0.1667,6), (2.36364,11), (5,1), (5,1), (6,1) \rangle$$

به دلیل از دست رفتن اطلاعات دقیق مقادیر خوشه‌بندی‌شده، به‌جز در خوشه‌های تکی که مرکز جرم آن‌ها با مقدار اولیه یکسان است، ساختار داده تلخیص د نمایشی با اتلاف از جریان داده فراهم می‌کند. بنابراین، برآورد چندک‌ها و پاسخ به پرس‌وجوی چندک مستلزم درون‌یابی بر پایه توزیع خوشه‌های ایجادشده توسط تابع مقیاس است.

الگوریتم ۱۲- پاسخ به پرس‌وجوهای چندک با تلخیص ت
ورودی: داده‌ساختار تلخیص د با m خوشه
ورودی: مقدار $q \in [0,1]$
خروجی: q - چندک
$n \leftarrow \sum_{j=1}^{m} c_j^{count}$
اگر $n \cdot q < 1$ آنگاه
برگرداندن c_1
اگر $n \cdot q > n - \frac{1}{2} c_m^{count}$ آنگاه
برگرداندن c_m
// یافتن i به طوری‌که $\exists\, i \in [1,m): q(c_i) + \frac{1}{2n} c_{i+1}^{count} > q$،
// ⟸ چندک جستجو شده جایی بین c_i و c_{i+1}
اگر $c_i^{count} == 1$ و $q(c_i) > q$ آنگاه
برگرداندن c_i
اگر $c_{i+1}^{count} == 1$ و $q(c_{i+1}) + \frac{1}{n} \leq q$ آنگاه
برگرداندن c_{i+1}

$\Delta_{\text{چپ}} \leftarrow (c_i^{count} == 1)?\, 1 : 0$
$\Delta_{\text{راست}} \leftarrow (c_{i+1}^{count} == 1)?\, 1 : 0$
$w_{\text{چپ}} \leftarrow n \cdot q - n \cdot q(c_i) + \frac{c_i^{count} - \Delta_{\text{چپ}}}{2}$
$w_{\text{راست}} \leftarrow n \cdot q(c_{i+1}) - n \cdot q + \frac{c_{i+1}^{count} - \Delta_{\text{راست}}}{2}$
برگرداندن $\frac{c_i \cdot w_{\text{راست}} + c_{i+1} \cdot w_{\text{چپ}}}{w_{\text{راست}} + w_{\text{چپ}}}$

بنابراین، برای یافتن چند q از ساختار داده تلخیص د، ابتدا رتبه متناظر با آن، یعنی $n \times q$، محاسبه می‌شود که در آن n تعداد کل مقادیر خلاصه‌شده در ساختار داده تلخیص د است. اگر این رتبه کمتر از یک باشد، مرکز جرم c_1 به عنوان مقدار چندک گزارش می‌شود. همچنین، اگر این رتبه در نیمه دوم آخرین خوشه یا فراتر از آن قرار گیرد، c_m که بزرگ‌ترین مقدار موجود در تلخیص د است، به عنوان پاسخ بازگردانده می‌شود. در غیر این صورت، دو خوشه C_i و C_{i+1} جستجو می‌شوند، به‌گونه‌ای که مقدار چندک مورد نظر (q) میان برآوردهای چندک این دو خوشه، که با استفاده از رابطه (۵.۴) محاسبه می‌شوند، قرار گیرد. اگر خوشه سمت چپ، یعنی C_i، خوشه تکی باشد و بیشترین مقدار چندک متناظر با آن از q فراتر رود، مرکز جرم c_i به عنوان مقدار چندک گزارش می‌شود، زیرا این مرکز جرم همان مقداری است که واقعاً در خوشه ذخیره شده است. به همین ترتیب، اگر خوشه سمت راست، یعنی C_{i+1}، خوشه تکی باشد و کمترین مقدار چندک برآوردشده آن از q کمتر باشد، این خوشه مسئول چندک مورد نظر در نظر گرفته می‌شود و مرکز جرم c_{i+1} به عنوان بهترین برآورد گزارش می‌شود. در غیر این صورت، سهم هر یک از دو خوشه در مقدار چندک مورد نظر محاسبه می‌شود و با استفاده از این سهم‌ها، وزن هر خوشه تعیین می‌شود. در نهایت، با محاسبه میانگین وزنی مراکز جرم دو خوشه، مقدار چندک به روش درون‌یابی برآورد و به عنوان پاسخ گزارش می‌شود.

عملکرد الگوریتم برآورد چندک به تابع مقیاس انتخاب‌شده وابسته است. استفاده از توابع مقیاسی که در دو انتهای توزیع تعداد بیشتری خوشه تکی ایجاد می‌کنند، می‌تواند دقت برآورد چندک‌های حدی را بهبود دهد. همچنین توصیه می‌شود در طول فرایند نمایه‌گذاری، کمینه و بیشینه مقادیر نیز ذخیره شوند تا در مرحله درون‌یابی برای افزایش دقت برآورد مورد استفاده قرار گیرند.

مثال ۱۰- پرس‌وجوی چندک با تلخیص د: پرس‌وجوی چندک را برای محاسبه چندک ۰.۶۵ از داده‌ساختار تلخیص د ساخته شده در مثال ۵.۹ انجام می‌دهیم:

$$T - DIGEST = \langle (0.1667,6), (2.36364,11), (5,1), (5,1), (6,1) \rangle$$

تعداد کل مقادیر در تلخیص د مجموع شمارش‌های همه خوشه‌ها است $n = 20$ است.

رتبه چندک جستجو شده x برابر با $nq = 20 \times 0.65 = 13$ است، که نه کمتر از یک و نه خیلی نزدیک به n است، بنابراین جستجو برای دو خوشه متوالی C_i و C_{i+1} را آغاز می‌کنیم که مراکز جرم آنها چندک جستجو شده را احاطه خواهند کرد. در داده‌ساختار تلخیص د فعلی، اینها C_2 و C_3 هستند، زیرا

$$q(c_2) + \frac{1}{2} \cdot 20\, c_3^{count} = 6 + 1120 + 140 = 0.875 > 0.65,$$

همانطور که در الگوریتم ۱۲ لازم است. خوشه C_3 تکی است ($c_3^{count} = 1$)، باید حداکثر مقدار چندک آن را با مقدار چندک جستجو شده q مقایسه کنیم تا بررسی کنیم که آیا مرکز جرم آن می‌تواند بهترین تناسب باشد یا خیر. بنابراین، محاسبه می‌کنیم

$$q(c_3) + 120 = 6 + 11 + 120 + 120 = 0.95,$$

که به طور قابل توجهی از مقدار $q = 0.65$ فراتر می‌رود و نتیجه می‌گیریم که چندک واقعی در جایی بین مراکز جرم c_3 و c_2 قرار دارد، اما، خوشه راست دارای تصحیح سوگیری $\Delta_{\text{راست}} = 1$ تکی است. بنابراین، چندک جستجو شده را می‌توان با درون‌یابی با استفاده از میانگین وزنی مراکز جرم تخمین زد، که در آن وزن‌ها هستند

$$w_{\text{چپ}} = 20 \cdot 0.65 - 20 \cdot q(c_2) + \frac{c_2^{count} - 0}{2} = 13 - 20 \cdot \frac{6 + 11}{20} + \frac{11}{2} = 1.5,$$

$$w_{\text{راست}} = 20 \cdot q(c_3) - 20 \cdot 0.65 - \frac{c_3^{count} - 1}{2} = 20 \cdot \frac{6 + 11 + 1}{20} - 13 - \frac{1 - 1}{2} = 5$$

در نهایت، چندک ۰.۶۵ تخمینی برای مجموعه داده مثال ۹ است:

$$x = \frac{c_2 \cdot w_{\text{راست}} + c_3 \cdot w_{\text{چپ}}}{w_{\text{چپ}} + w_{\text{راست}}} = \frac{2.36364 \cdot 5 + 5 \cdot 1.5}{1.5 + 5} = 2.97,$$

که بسیار نزدیک به مقدار دقیق ۳ برای آن مجموعه داده است.

مشابه پرس‌وجوی چندک، ساختار داده تلخیص د را می‌توان برای پاسخ به پرس‌وجوی چندک معکوس و برآورد رتبه مقدار مشخص x نیز به کار گرفت. این فرایند با مقایسه مقدار x با کمینه و بیشینه مراکز جرم موجود در تلخیص د آغاز می‌شود که به‌ترتیب متناظر با چپ‌ترین و راست‌ترین خوشه‌ها هستند. اگر x خارج از این بازه قرار داشته باشد، بسته به اینکه در کدام سمت قرار گرفته است، به‌ترتیب مقدار ۱ یا تعداد کل مقادیر n به عنوان رتبه تقریبی آن گزارش می‌شود.

اگر x در این بازه قرار داشته باشد، مراکز جرم ساختار داده برای یافتن این مقدار جستجو می‌شوند. در صورتی که یک یا چند خوشه با مرکز جرم برابر با x یافت شوند، شمارش آن‌ها با یکدیگر جمع می‌شود و رتبه x برابر با رتبه خوشه‌ای گزارش می‌شود که کوچک‌ترین اندیس را در میان آن خوشه‌ها دارد.

اگر هیچ‌یک از حالت‌های بالا برقرار نباشد، مقدار x میان مراکز جرم دو خوشه متوالی، یعنی c_i و c_{i+1}، قرار دارد و در نتیجه، رتبه آن حداقل برابر با $n \times q(c_i)$ خواهد بود. اگر هر دو خوشه تکی باشند، مراکز جرم آن‌ها دقیقاً همان مقادیر اولیه جریان داده

هستند. بنابراین، نیازی به اعمال هیچ‌گونه اصلاحی نیست و همین مقدار به عنوان رتبه برآوردشده گزارش می‌شود. اگر تنها یکی از این دو خوشه تکی باشد، رتبه با در نظر گرفتن سهم مقیاس‌بندی‌شده خوشه دیگر اصلاح می‌شود تا برآورد دقیق‌تری به دست آید. در غیر این صورت، با استفاده از اندازه هر دو خوشه، مقدار رتبه به روش درون‌یابی برآورد شده و بر اساس آن، رتبه نهایی گزارش می‌شود.

الگوریتم ۱۲- پاسخ به پرس‌وجوهای چندک معکوس با تلخیص د
ورودی: مقدار x
ورودی: داده‌ساختار تلخیص د با m خوشه
خروجی: رتبه عنصر
$n \leftarrow \sum_{j=1}^{m} c_j^{count}$
اگر $x < c_1$ آنگاه برگرداندن ۱
اگر $x > c_m$ آنگاه برگرداندن n
// بررسی مرکز جرم بودن x
اگر $\exists\, j: c_j == x$ آنگاه
$J \leftarrow \{j: c_j == x\}, i^* \leftarrow min(J)$
برگرداندن $n \cdot q(c_i^*) - c_{i^*}^{count} + \frac{1}{2}\sum_{j \in J} c_j^{count}$
// در این مرحله، می‌توانیم مطمئن باشیم که $\exists\, i \in [1, m): x \in (c_i, c_{i+1})$
رتبه $\leftarrow n \cdot q(c_i)$
اگر $c_i^{count} == 1$ و $c_{i+1}^{count} == 1$ آنگاه
برگرداندن رتبه
$\hat{x} \leftarrow \frac{x - c_i}{c_{i+1} - c_i}$
اگر $c_i^{count} == 1$ آنگاه
برگرداندن رتبه $+ \frac{\hat{x}}{2} \cdot c_{i+1}^{count}$
اگر $c_{i+1}^{count} == 1$ آنگاه
برگرداندن رتبه $- \frac{1 - \hat{x}}{2} \cdot c_i^{count}$
برگرداندن رتبه $+ \frac{\hat{x}}{2} \cdot c_{i+1}^{count} - \frac{1 - \hat{x}}{2} \cdot c_i^{count}$

همانطور که قبلا اشاره شد، برای پاسخ به پرس‌وجوی محدوده، انجام دو پرس‌وجوی چندک معکوس و یافتن تفاوت بین رتبه‌های مرزهای محدوده کافی است.

ویژگی‌ها

تعادل بین اندازه ساختار داده تلخیص د، که با پارامتر فشرده‌سازی σ کنترل می‌شود، و سرعت و دقت برآورد چندک‌ها در ساختار داده مذکور دارای اهمیت است. به بیان دیگر، با انتخاب مقدار کوچک‌تر برای σ و استفاده از بافر بزرگ‌تر b، می‌توان سرعت پردازش بیشتری را با مصرف ثابت حافظه به دست آورد. در مقابل، اگر هدف دستیابی به بیشترین دقت باشد، استفاده از مقدار بزرگ‌تر σ که فشرده‌سازی کمتری را اعمال می‌کند، همراه با بافر بزرگ‌تر، برای مثال با اندازه‌ای در حدود 10σ، توصیه می‌شود. از سوی دیگر، در شرایطی که محدودیت حافظه اهمیت بیشتری دارد، استفاده از یک بافر کوچک‌تر و مقادیر کوچک‌تر برای پارامتر فشرده‌سازی σ مناسب‌تر خواهد بود. همان‌گونه که پیشنهاددهندگان الگوریتم تلخیص د نشان داده‌اند، در صورت استفاده از تابع مقیاس (۵)، تعداد

خوشه‌های m در ساختار داده تلخیص د که ویژگی تلخیص (۷) را برآورده می‌کنند و دست‌کم $n \geq \sigma/2$ مقدار را نمایه کرده‌اند، در بازه زیر قرار می‌گیرد:

$$\left\lfloor \frac{\sigma}{2} \right\rfloor \leq m \leq \lfloor \sigma \rfloor$$

مثال ۱۱- تخمین فضای مورد نیاز: حداقل n = 1000 مقدار را با پارامتر فشرده‌سازی $\sigma = 100$ نمایه کنیم. بنابراین، با توجه به رابطه (۱۰)، ۵۰ تا ۱۰۰ خوشه در تلخیص د کاملا ادغام شده مورد انتظار خواهد بود.

در ساختار داده تلخیص د، هر خوشه با دو مؤلفه مرکز جرم و تعداد مقادیر نمایه‌شده در آن خوشه نمایش داده می‌شود. در نتیجه، اگر برای شمارنده‌ها از اعداد صحیح ۳۲ بیتی و برای مقدار مرکز جرم از اعداد ممیز شناور ۶۴ بیتی با دقت دوگانه استفاده شود، هر خوشه در مجموع به ۱۲ بایت حافظه نیاز خواهد داشت. بنابراین، کل ساختار داده با حدود ۱٫۲ کیلوبایت حافظه قابل ذخیره‌سازی است.

برای دستیابی به دقت بالا، معمولاً اندازه بافر را حدود ده برابر پارامتر فشرده‌سازی در نظر می‌گیرند. در نتیجه، با انتخاب $b = 10 \times \sigma = 1000$، می‌توان برای مقادیر موجود در بافر از شمارنده‌های کوچک‌تر ۱۶ بیتی و اعداد ممیز شناور ۶۴ بیتی با دقت دوگانه استفاده کرد. این پیکربندی در زمان اجرا حدود ۱۰ کیلوبایت حافظه اضافی نیاز دارد.

الگوریتم تلخیص د، دقت برآورد چندک q را با خطایی حفظ می‌کند که متناسب با $q(1-q)$ است. برخلاف بسیاری از الگوریتم‌های دیگر که تنها یک کران ثابت برای خطای مطلق ارائه می‌کنند، در تلخیص د خطای نسبی لحاظ می‌شود. این ویژگی سبب می‌شود دقت برآورد چندک‌های حدی به‌طور قابل توجهی افزایش یابد و الگوریتم در برابر خطاهای بزرگ در این نواحی مقاوم باشد. افزون بر این، یکی از مزیت‌های مهم تلخیص د نسبت به تلخیص چ آن است که از مقادیر ممیز شناور نیز پشتیبانی می‌کند، در حالی که تلخیص چ، همان‌گونه که پیش‌تر بیان شد، تنها برای داده‌های صحیح قابل استفاده است.

دو ساختار داده تلخیص د را می‌توان به‌سادگی با استفاده از همان الگوریتم ادغام با یکدیگر ترکیب کرد. البته، ساختار حاصل دقیقاً با تلخیص د ایجادشده از پردازش مستقیم کل جریان داده یکسان نخواهد بود. با وجود این، نتایج تجربی نشان می‌دهند که ساختار ادغام‌شده همچنان برآوردهای دقیقی در اختیار می‌نهد. در نتیجه، می‌توان تلخیص‌های د ایجادشده برای بخش‌های مختلف جریان داده را به‌صورت موازی تولید و سپس با یکدیگر ادغام کرد و از ساختار نهایی برای پاسخ به پرس‌وجوهای مبتنی بر رتبه استفاده کرد. این ویژگی، تلخیص د را به الگوریتمی مناسب برای پردازش موازی، چارچوب‌های MapReduce و کاربردهای استخراج جریان در سامانه‌های داده بزرگ تبدیل کرده است.

امروزه الگوریتم تلخیص د استفاده‌ای گسترده دارد. برای نمونه، این الگوریتم در محاسبه صدک‌ها در Elasticsearch به کار گرفته شده است و پیاده‌سازی آن در stream-lib و Apache Mahout نیز در دسترس است.