Hyperloglog 9910220 سوال ۷ میان ترم دی اس استاد صدیقین - اعمال محمدی -

راه حلی می خواهیم که با صرف حافظه $O(1)$ مشخص کند تعداد اعداد متمایز یک مجموعه ای از اعداد را

اگر فرض کنیم که اعداد ورودی ما در بازه $[1, n]$ هستند و در بین همه اعداد ما، ۲ تا عدد متمایز وجود

دارد، درست می آوریم که: اعداد هش، باینری اند (شامل ۰ و ۱) ⟵ در تابع هش ایده آل، دو عدد ورودی یکسان به یک خانه هش می شوند

فرض می کنیم h یک تابع هش جهانی ایده آل است که $h(x): [1, n] \to [0, m]$

پس در تمام هش ها اگر طول بزرگترین تعداد صفرهای متوالی را پیدا کنیم، که در سمت چپ عدد ظاهر شده اند، ⟸ احتمال وجود یک صفر در سمت چپ برابر است با $\frac{1}{2}$

احتمال وجود دو صفر متوالی در سمت چپ برابر است با $\frac{1}{4}$

⋮

احتمال وجود ... در سمت چپ عدد: $\frac{1}{2}\log m$

با تقریب خوبی تعداد اعداد متمایز پیدا می شد که

طول رشته $=m$

تابع هش ایده آل داریم پس اعداد ورودی یکسان مشکلی ایجاد نمی کنند و متمایز شناخته نمی شوند.

تابع هش ایده آل داریم پس اتفاقی مثل ۱۰ تا صفر متوالی در سمت چپ عدد رخ نمی دهد و احتمال آن خیلی پایین است $\left(\frac{1}{2^{10}}\right)$ ⟵

روش ما به این شکل است که اگر طول بزرگترین تعداد صفرهای متوالی، q باشد،

حدوداً 2^q عدد متمایز داریم چون از بین m هش موجود ما، $\frac{1}{2^q}\cdot n$ هش داریم که q صفر متوالی در سمت چپ

آن وجود دارد. ⟸ $Pr(x) = \frac{\frac{n}{2^q}}{n} = \frac{1}{2^q}$ می باشد که تخمین خوبیست در حالت ایده آل!

اما تخمین ما دارای ۲ مشکل و ایراد است ⟸ ۱- اگر حالتی پرت به وجود آید، مثلاً اگر ۵ هش داشته

باشیم که در یکی از آن، ۷ صفر متوالی داریم، نتیجه تخمین خیلی خطا خواهد بود.

۲- تخمین ما صرفاً شامل اعدادی از توان ۲ می باشد که این الگوریتم عددی بین این اعداد خروجی نخواهد داد.

$2^n \Rightarrow n \in \mathbb{N}$ ⟵

برای مشکل اول دو راه حل ارائه می دهیم ⟸ یکی اینکه جواب نهایی را میانگین تخمین های توابع هش قرار دهیم که

با مستقل بودن هش ها، مشکل ایجاد می شود و ایده آل نیست.

یکی هم اینکه هش ها را با استفاده از چندبیت سمت چپ هش ها دسته بندی کنیم و برای هر دسته دوباره تخمین زده و جمع کنیم.

مثلاً برای ۳ بیت ابتدایی سمت چپ، ۸ حالت روبرو پیش می آید:

$000 \to 2^{l_0}$ $011 \to 2^{l_3}$ $110 \to 2^{l_6}$

$001 \to 2^{l_1}$ $100 \to 2^{l_4}$ $111 \to 2^{l_7}$

$010 \to 2^{l_2}$ $101 \to 2^{l_5}$

⟸ تخمین نهایی: $2^{l_0}+2^{l_1}+2^{l_2}+2^{l_3}+2^{l_4}+2^{l_5}+2^{l_6}+2^{l_7}$

حتی برای بهینه تر کردن روشمان و برای اینکه جواب پرت باز هم تأثیرش کمتر شود می توانیم از l_i ها $i \in \mathbb{N}$ میانگین بگیریم.

فرض می کنیم p دسته داریم، $p = 2^k$ پس:

$p\left(\frac{l_0+l_1+\cdots l_v+\cdots+l_{p-1}}{p}\right) =$ تخمین نهایی

در اینجا برای تابع درهم سازی، به $O(\log n)$ حافظه و برای ذخیره سازی اعداد به $O(p \log \log n)$ حافظه

نیاز داریم که با $O(1)$ قابل مقایسه است. روشی که توضیح دادیم، Hyperloglog نام دارد.

خطای روش هایپرلاگ لاگ به این صورت است که در حالت کلی تخمین به دست آمده بیشتر از $2^{\text{تعداد اعداد متمایز}}$ است.

برای کم کردن این خطا، ضریب α_p در آن ضرب می شود پس تخمین نهایی برابر است با:

$\alpha_p \cdot p \cdot 2^{\frac{1}{p}\sum_{i=0}^{p-1} l_i}$: تخمین نهایی

$$\left(\alpha_m = \left(p \int_0^\infty e^{-q} q^{-\frac{1}{p}} dq \; \frac{2^{-\frac{1}{p}}}{\ln 2}\right)^{-p}\right)$$

p دسته داریم