پیچیدگی محاسبه: پیچیدگی محاسبات در مسائل داده بزرگ

۵ تیر ۱۳۹۴

دکتر فروغمند

علی ستاری جاوید
شماره دانشجویی: ۹۳۲۰۱۸۴۶

چکیده

در کلاس، در اغلب موارد مسائل در کلاس P به عنوان مسائلی که به راحتی حل می‌شوند دسته‌بندی شدند. اما در عمل، بسیاری از مسائلی که نیاز به حل شدن دارند نیاز به پردازش حجم بسیار عظیمی از داده‌ها دارند که حتی اگر الگوریتم حل آنها از کلاس n^2 باشد هم، از توان محاسباتی فعلی ما خارج می‌شوند. به عنوان مثال، می‌توان به پخش یک فیلم اشاره کرد. فیلم‌های با کیفیت FullHD امروزه به یکی از موارد عادی زندگی ما تبدیل شده‌اند. هر ثانیه از فیلم‌هایی با این کیفیت شامل بیش از 350MB داده است. به این معنی که برای پخش این گونه فیلم‌ها، این حجم عظیم از داده باید به نمایشگر در هر ثانیه ارسال شود. این در حالی است که یک پردازنده‌ی معمولی، امروزه می‌تواند حداکثر چیزی در حدود 10^9 پردازش در هر ثانیه انجام دهند، یعنی فقط ۳ برابر آنچه برای پخش یک فیلم لازم است.

یک مثال دیگر می‌توان به گرافیک کامپیوتری، و یا بازی‌های رایانه‌اش اشاره کرد. در بسیاری از بازی‌هایی که امروزه منتشر می‌شوند، در یک صحنه ممکن است بیش از ۲۰۰ هزار مثلث کشیده شود، که برای کشیدن هر مثلث، باید تعداد بسیار زیادی محاسبات ماتریسی انجام شود. طبعا اگر الگوریتم‌های این عملیات از مراتبی بیشتر از $O(n($ باشند، با توان محاسباتی در دسترس عموم مردم، نمی‌توان این پردازش را با سرعتی انجام داد که بازی قابل انجام باشد.

در این گزارش می‌خواهیم به این مسئله بپردازیم که این گونه مسائل، چه شرایطی دارند و چگونه می‌توان آنها را حل نمود

مسائل خطی

شاید اولین راه حلی که به نظر برسد، محدود کردن راه حل‌ها، به الگوریتم‌های خطی است. به عبارت دیگر، روش‌های حلی که با افزایش حجم ورودی‌ها، حجم محاسبات به صورت حداکثر خطی رشد می‌کند. با توجه به قضیه‌ی سلسه مراتب چند جمله‌ای می‌دانیم همه‌ی مسائل را نمی‌توان با راه حل‌های خطی حل نمود. اما این پرسش باقی است که این محدودیت چقدر توان محاسباتی ما را کاهش می‌هد و به عبارت دیگر، آیا مسائل مورد نیازیمان را می‌توانیم با وجود این محدودیت حل کنیم؟

حل k معادله k مجهول

یک دستگاه خطی، متشکل از k معادله و k مجهول، به صورت یک ماتریس $k * k$ نمایش داده می‌شود. ضرب این چنین ماتریس‌هایی از مرتبه‌ی زمانی $O(k^۳)$ زمان نیاز دارد. همچنین ضرب این چنین ماتریس‌هایی در یک بردار k بعدی، حداقل به $O(k^۲)$ عمل ضرب و جمع نیاز دارد. اما در مساله‌ی گرافیک کامپیوتری که قبلا مطرح شد، k معمولا یک عدد ثابت و بسیار کوچک مانند ۳ یا ۴ می‌باشد. آنچه این مساله را سخت می‌کند، تعداد بسیار زیاد ضرب‌های ماتریسی‌ای که باید انجام شود. به عبارت دیگر، در این مساله، ما نیاز به انجام n عمل ضرب از نوع ضرب بین دو ماتریس $۴ * ۴$ داریم، که به وضوح راه حل این مساله از مرتبه‌ی خطی است.

یافتن k امین بزرگترین عدد

اگر یک آرایه از اعداد در اختیار داشته باشیم، یافتن میانه‌ی این آرایه، در بسیاری زمینه‌ها از جمله تحلیل‌های آماری کاربرد دارد. هر چند مرتب کردن یک آرایه، زمانی از مرتبه‌ی غیر خطی نیاز دارد، اما می‌توانیم، عنصر مطعلق یک اندیس دلخواه از آرایه‌ی مرتب شده را، بدون اینکه واقعا آرایه را مرتب کنیم بیابیم. یکی از معروف‌ترین روش‌هایی که برای این کار وجود دارد الگوریتم "میانه‌ی میانه" ها نام دارد. در این روش، ابتدا، کل آرایه به دسته‌هایی ۵ تایی تقسیم می‌شوند و هر دسته جداگانه مرتب می‌شود. سپس با استفاده از میانه هر کدام از این دسته‌ها، میانه‌ی کلی آرایه پیدا می‌شود. در زمان پیدا شدن میانه‌ی کلی، اعضای آرایه به گونه‌ای تقسیم می‌شوند که تمام اعداد کوچکتر از میانه در نیمه‌ی اول قرار بگیرند و تمام اعداد بزرگتر از میانه در نیمه‌ی دوم. سپس به صورت بازگشتی، بسته به اینکه k بزرگتر از نیمی از اعداد است یا خیر، در نیمه‌ی بزرگتر و یا نیمه‌ی کوچکتر به دنبال عنصر k ام می‌گردیم.
در تحلیل پیچیدگی این روش، کافی است چند نکته را دقت نماییم:

۱. در هر مرحله، از دسته‌های ۵ تایی، نیمی دسته‌ها هستند که میانه‌شان کمتر از مقدار میانه‌ی کلی است. و نیم دیگر این دسته‌ها میانه‌هایی بیشتر از میانه کلی دارند.

۲. در میان دسته‌هایی که میانه‌ای کمتر از میانه کلی دارند، حداقل ۳ عضو آنها نمی‌تواند میانه‌ی کلی باشد. دو عضو کوچکتر و میانه آن دسته. با فرض اینکه دسته ای که شامل میانه‌ی کلی است را در نظر نگیریم. همچنین در نیمه‌ی دیگر ۳ عضو بزرگتر نمی‌توانند میانه کلی باشند. در نتیجه با فرض مرتب شدن دسته ها به ترتیب فقط میانه شان، می‌توانیم مطمئن باشیم که $\frac{۶}{۱۰}$ از اعداد نمیتوانند میانه باشند.

۳. میتوان میانه‌ی میانه‌ها را به صورت بازگشتی یافت و این کار دقیقا $\frac{۱}{۵}$ زمان کلی الگوریتم طول خواهد کشید.

۴. پس از یافتن میانه‌ی میانه‌ها، می‌توان گروه‌ها را به نیمه‌ی بزرگتر و نیمه‌ی کوچکتر تقسیم کرد و با این کار طبق استدلال قبل $\frac{۶}{۱۰}$ اعداد را از مجموعه‌ی اعدادی که احتمال دارد میانه باشند حذف کرد.

با توجه به نکات بالا، حد زمان مورد نیاز برای محاسبه‌ی میانه‌ی یک آرایه از مرتبه‌خطی خواهد بود و در نتیجه یافتن k امین عدد نیز میتواند در زمان خطی انجام شود.

محدودیت ها

با اینکه تا امروز توانسته‌ایم اکثر مسائلی که در مسائلی را که با آنها مواجه شده‌ایم با الگوریتم‌ها خطی حل کنیم، ولی با توجه به قضیه‌ی سلسله مراتب چند جمله‌ای می‌دانیم تمام مسائل را نمی‌توان به این صورت محاسبه نمود. به عنوان نمونه مرتب سازی یک آرایه، یکی از مسائلی است که علی‌رغم کاربرد گسترده‌اش در انواع محاسبات، نه تنها راه حلی با روش‌های خطی برایش پیدا نکرده‌ایم بلکه اثبات هم شده است که راه حلی چند جمله‌ای ندارد.
مرتب‌سازی یک آرایه از اجسام دلخواه را می‌توان به عنوان یک الگوریتم در نظر گرفت، که در صورت نیاز می‌تواند دو عضو دلخواه را انتخاب کرده، و توسط یک جعبه‌ی سیاه آن دو را مقایسه نماید. البته فرض بر این است که مقایسه‌ی دو عنصر به محاسبه‌ی زیادی نیاز ندارد و در یک لحظه قابل انجام است. حال فرض کنید الگوریتم به غیر از عملگر مقایسه هیچ روش دیگری برای مرتب کردن یک آرایه نداشته باشد. این الگوریتم باید بتواند با استفاده از فقط عملگر محاسبه، از بین $n!$ ترتیب مختلفی که می‌شود یک آرایه را چید، یکی را انتخاب کند. مقایسه‌ی دو عضو، تنها یک جواب بله، یا خیر به دنبال خواهد داشت که یک الگوریتم، در بهترین حالت می‌تواند با استفاده از این پاسخ نیمی از حالات ممکن را حذف نماید. در نتیجه الگوریتم برای اتخاب یک حالت از بین $n!$ حالت باید حداقل $log(n!) \leq O(nlog(n))$ مقایسه را انجام دهد. در نتیجه هیچ روش مرتب سازی‌ای نیست که بتواند با کمتر از این تعداد مقایسه، ترتیب صحیح را بیابد.
از طرفی، حتی الگوریتم‌های خطی نیز در بعضی موارد برای انجام محاسبات سرعت کافی را ندارند. به عنوان مثال در یک جستجوی اینترنتی، موتور جستجو باید از بین میلیاردها میلیارد صفحات موجود بر روی اینترنت، منابع مطابق با عبارت جستجوی شما را بیابد. این در حالی است که طبق آنچه گفته شد، یک پردازنده به تنهایی فقط می‌تواند یک میلیارد صفحه را بشمارد، که کاری بسیار ساده‌تر جستجو میان صدها کلمه‌ی موجود در هر صفحه است!
به عنوان یک مثال دیگر، می‌توان به یافتن رمزهای عبور اشاره کرد. رمز عبور، واژه‌ای است معمولا متشکل از بیش از ۸ حرف، که هر کدام از این حروف می‌توانند اعداد یا علامات و یا حروف کوچک و بزرگ انگلیسی باشند. بیراه نیست اگر بگوییم هر حرف بیش از ۶۴ حالت می‌تواند داشته باشد و در نتیجه یک کلمه‌ی عبور ۸ حرفی میتواند یکی از $۱۰^{۱۴}$ حالت ممکنی باشد که این حروف می‌توانند کنار هم قرار بگیرند. بدیهی است که بررسی این تعداد حالت از توان هر پردازنده‌ی در دسترس امروزی در زمان کوتاه خارج است.

الگوریتم‌های موازی

هر چند سرعت پردازنده‌ها از نظر عملی به سادگی میسر نیست، ولی می‌توان به سادگی تعداد پردازنده‌ها را به تعداد دلخواه افزایش داد. پس این سوال مطرح می‌شود که آیا می‌توان با افزایش تعداد پردازنده‌ها سرعت محاسبات را افزایش داد؟ به عنوان مثال، اگر می‌دانیم مرتب سازی آرایه حداقل به $O(nlogn)$ زمان نیاز دارد، آیا می‌توان با افزایش تعداد پردازنده‌ها سرعت را به قدری بالا برد که در زمان $O(logn)$ یک آرایه محاسبه شود؟ و یا می‌توان رمز یک فایل قفل شده را در کسری از ثانیه یافت؟

مرتب سازی به صورت موازی

روش‌های بسیاری برای مرتب سازی اعداد به صورت موازی وجود دارد. هر چند هیچ کدام از این راه حل‌ها، نتوانسته‌اند به ایده‌آل $O(logn)$ دست یابند، ولی تعدادی از آنها با اندکی بازده کمتر، افزایش سرعت چشمگیری را به همراه داشته‌اند. به عنوان مثال، می‌توان الگوریتمی به نام مرتب سازی بیتونیک نام برد که از با $O(log^{۲}n)$ عملیات، می‌تواند یک آرایه را مرتب نماید. این شیوه‌ی مرتب سازی نیز بازگشتی است و به این صورت عمل می‌کند که ابتدا آرایه را به دو قسمت تقسیم می‌نماید. سپس نیمه‌ی اول را به ترتیب سعودی، و نیمه‌ی دوم را به ترتیب نزولی مرتب می‌کند. سپس با استفاده از $\frac{n}{۲}$ پردازنده، ابتدا هر عنصر آرایه را با عنصری که در $\frac{n}{۲}$ خانه بعد وجود دارد مقایسه می‌نماید و اگر محتوای عنصر نزدیکتر به انتها کوچکتر بود، جای این دو عنصر را تعویض می‌نماید. در مرحله‌ی بعد، همین مقایسه و تعویض را، در دو نیمه‌ی ابتدایی و انتهایی به صورت جداگانه با طول گام نصف مرحله‌ی قبلی انجام می‌دهد. و به همین ترتیب طول گام را در هر مرحله نصف می‌کند تا جایی که گام‌ها به طول ۲ برسند.

پیچدگی مداری

هر چند در نگاه اول، پردازش موازی، ارتباط چندانی به پیچیدگی مداری ندارد، اما مجموعه کلاس‌هایی از مدارها وجود دارند که مستقیما معادل محسابات موازی هستند. این مجموعه کلاس‌ها به اختصار کلاس‌های خانواده‌ی NC^i نامیده می‌شوند. هر عضو از NC^i، خانواده از مدارهای بولی را شامل می‌شود که حداکثر عمقی از مرتبه‌ی $O(log^i n)$ دارند. معادل همین کلاس در بیان محاسبات موازی، خانواده از الگوریتم‌ها را تشکیل می‌دهند که با استفاده از حداکثر چندجمله‌ی پردازشگر، می‌توانند در $O(log^i n)$ محاسبه جواب دهند. در تحلیل این درسته از الگوریتم‌های یکی از نتایج جالب این است که کلاس NC^1 چیزی بیش از الگوریتم‌های با حافظه‌ی لگاریتمی را نمی‌تواند حل کند، حال آنکه کلاس NC^2 نه تنها توان محاسباتی بیش از الگوریتم‌های لگاریتمی دارد، بلکه تمام الگوریتم‌های غیر قطعی لگاریتمی را نیز شامل می‌شود.
هر چند این دسته از الگوریتم‌ها وعده از وجود حل بسیاری از کمبودهای توان پردازشی موجود را می‌دهند، اما هنوز دو پرسش اساسی در رابطه با آنها بدون حل باقی مانده است:

- آیا سلسله مراتب NC کامل است، یا به یک مرحله‌ی خاص فرو می‌ریزد؟
- و آیا تمام الگوریتم‌های چند جمله‌ای را می‌توان با استفاده از پردازش موازی، به صورت چشمگیر تسریع کرد؟ یا به عبارت دیگر، آیا $\bigcup_{i \in N} NC^i = NC = P$؟

محدودیت‌های پردازش موازی

همان‌طور که گفتیم نمی‌دانیم آیا می‌توان تمام الگوریتم‌ها را به صورت موثر موازی‌سازی کرد یا خیر. اما بیشتر محققین بر این باورند که مسائلی وجود دارند که به طور وجودی ترتیبی هستند. به این معنی که هیچ روش مناسبی وجود ندارد که بتواند به طور موثر عملیات محاسبه جواب آنها را بین هسته‌ی پردازشی تقسیم نماید. برای مطالعه‌ی این گروه از مسائل، دسته‌ای را جداگانه "چندجمله‌ی-تمام" نامگذاری کرده‌اند که عبارتی سختی تمام مسائل چند جمله‌ای را در بر می‌گیرد. از این دسته مسائل می‌توان به چند مورد زیر اشاره نمود:

- مساله‌ی محاسبه‌ی مداری: سوال این است که آیا در صورتی که یک مدار دلخواه (از عمق دلخواه)، و به همراه یک مقداردهی از راس‌های ورودی مدار داده شود، خروجی مدار در یکی از درگاه‌های مشخص شده چیست؟
- مساله‌ی برنامه‌ریزی خطی: تابعی خطی داده شده است. همچنین زیر فضایی از فضای تمام مقادیر ممکن متغیرها مشخص شده است. بیشترین مقدار تابع در این زیرفضا کدام است، و این مقدار به ازای چه مقادیر ورودی قابل دست‌یابی است؟
- ترتیب رویت رئوس، در جستجوی عمقی گراف: یک گراف، که رئوسش شماره گذاری شده‌اند به ما داده شده است. همچنین دو راس خاص در این گراف مشخص شده است. مساله این است که کدام یک از این دو راس، در جستجوی عمقی زودتر رویت میشوند؟

علاوه بر مسائل بالا، مشکلات دیگری هم در عمل وجود دارد. به عنوان مثال، مساله‌ی حل یک دستگاه چند جمله‌ای را در نظر بگیرید. هر چند این مساله را می‌توان با استفاده از پردازشگرهای موازی، با سرعت بسیار بالا حل نمود، ولی در عمل، میزان انتقال داده در میان پردازشگرها به حدی زیاد است که به عنوان عامل محدود کننده، حتی ممکن است از پردازش مستقیم بر روی یک هسته زمان بیشتری را مصرف شود.

جمع بندی

آنچه در این مقاله جمع آوری شده است، به شرح زیر خلاصه می‌شود:

- در دنیای امروز، مسئله‌هایی که باید حل شوند، در بسیاری موارد نیازمند تحلیل حجم بسیار زیادی از داده‌ها می‌باشند که در عمل باعث می‌شود الگوریتم‌های غیر خطی ناکارآمد شوند.
- اکثر مسائلی که با آنها سر و کار داریم، راه حل‌های بسیار سریع از مرتبه‌ی خطی دارند، هر چند مسائلی می‌شناسیم که مطمئن هستیم نمی‌شود آن‌ها را با الگوریتم‌های خطی حل نمود.

- محدودیت‌های فیزیکی به ما اجازه نمی‌دهند سرعت یک پردازنده‌ی منفرد را به میزان دلخواه افزایش دهیم. هر چند در بسیاری موارد می‌توانیم بار محاسباتی را در میان تعداد زیادی پردازنده تقسیم کنیم و در نتیجه سرعت محاسبه را به طور چشم گیری افزایش دهیم.
- مسائلی وجود دارند که تا کنون نتوانسته‌ایم برای حلشان راه حلی موازی ارائه دهیم، و حدس می‌زنیم مسائلی وجود داشته باشند که با افزایش تعداد پردازنده‌ها، نمی‌توان آن‌ها را با سرعتی بیش از آنچه امروزه در دست است، حل نمود.