به نام خدا

الگوریتم Timsort

مهدی حق‌وردی

فهرست مطالب

مقدمه و معرفی

مقدمه و معرفی

- در دنیای علوم کامپیوتر، مرتب‌سازی یک عملیات اساسی با کاربردهای بی‌شمار است.
- در میان انبوهی از الگوریتم‌های مرتب‌سازی، یکی از الگوریتم‌ها به دلیل کارایی، تطبیق‌پذیری و طراحی زیبا متمایز شده است: الگوریتم تیم‌سورت[1].
- این الگوریتم که توسط تیم پیترز[2] برای زبان برنامه نویسی پایتون[3] توسعه یافته است، به سنگ بنای پیاده‌سازی مرتب‌سازی در زبان‌ها و محیط‌های مختلف برنامه‌نویسی تبدیل شده است.
- ترکیب منحصر به فرد مرتب‌سازی ادغامی[4] و مرتب‌سازی درجی[5] به همراه بهینه‌سازی‌های مخصوص روی هر الگوریتم و بهینه‌سازی‌های تطبیقی، تیم‌سورت را به یکی از پیچیده‌ترین و کاربری‌ترین الگوریتم‌های مرتب‌سازی موجود تبدیل کرده است.

---

[1] Timsort
[2] Tim Peters
[3] Python programming language
[4] Merge sort
[5] Insertion sort

تاریخچه

تاریخچه

- الگوریتم تیم‌سورت، در سال ۲۰۰۲ توسعه یافت.
- تیم پیترز این الگوریتم را اینگونه توصیف می‌کند:

"A non-recursive adaptive stable natural mergesort / binary insertion sort hybrid algorithm"

- این الگوریتم از Python 2.3 تا حدود بیست سال، الگوریتم استاندارد مرتب‌سازی در پایتون بود و از نسخه‌ی 3.11.1 به دلیل تغییراتی که در سیاست‌های ادغام آن بوجود آمد، الگورتیمی به اسم Powersort بر پایه‌ی تیم‌سورت، جایگزین آن شد.
- الگوریتم تیم‌سورت در Java SE 7، Android، GNU Octave، V8، Swift و Rust پیاده‌سازی شده است.

چراها

- چرا non-recursive؟
چون طبق گفته‌ی تیم پیترز: «به طور خلاصه، روتین اصلی یک بار از سمت چپ تا راست، آرایه را طی، Runها[2] را شناسایی و هوشمندانه آنها را با هم ادغام می‌کند.»

- چرا adaptive؟
چون این الگوریتم با توجه به طول و ترتیب‌های از قبل موجود در آرایه، و همچنین بر اساس اندازه‌ی Runهای پیدا شده، تصمیماتی می‌گیرد تا از الگوریتم بهتری برای آن موقعیت استفاده کند.

- چرا stable؟
چون این الگوریتم، ترتیب عناصر یکسان در آرایه‌ی اولیه را حفظ می‌کند. برای مثال اگر لیستی از این اسامی داشته باشیم: $[\text{peach}, \text{straw}, \text{apple}, \text{spork}]$ و آنرا بخواهیم بر اساس حرف اول کلمات مرتب کنیم، چنین چیزی می‌گیریم: $[\text{apple}, \text{peach}, \text{straw}, \text{spork}]$ اگر دقت کنید در لیست اولیه، straw قبل از spork آمده بود و در لیست مرتب شده هم همین ترتیب حفظ شد. به این نگهداری ترتیب پایداری الگوریتم مرتب‌سازی می‌گویند.

[2] در ادامه مفهوم Run توضیح داده می‌شود.

چراها

- چرا hybrid؟
  چون این الگوریتم از ترکیب دو الگوریتم merge sort و binary insertion sort برای مرتب سازی استفاده می‌کند.

چرا تیم‌سورت؟

چرا تیم‌سورت؟

- پیچیدگی زمانی الگوریتم تیم‌سورت با الگوریتم‌های Merge sort، Quick sort و Heap sort برابری می‌کند و برابر $O(nlgn)$ است.
- اما این تحلیل کلی یک سری جزئیات راجع به پیچیدگی زمانی الگوریتم را پنهان می‌کند که آن پیچیدگی یک constant factor اثرگذار در میزان پیچیدگی الگوریتم است. $(c_f.nlgn)$
- برای مثال در الگوریتم Quick sort انتخاب مقدار left، right و pivot تاثیرگذار است و در nهای کوچک سرعت را پایین می‌آورد.
- در الگوریتم Merge sort هم ما فضایی به اندازه‌ی $n + m$ برای ادغام کردن آرایه‌ها آن هم به صورت بازگشتی و تعداد زیاد نیاز دارد. همچنین این الگوریتم یک الگوریتم بازگشتی‌ست و درخت بازگشتی و یک system stack برای اجرا نیاز دارد.
- بخاطر جابجایی‌هایی در الگوریتم Heap sort انجام می‌شود، Locality of Reference در آن نقض شده و پیشبینی‌های پردازنده برای کش کردن داده‌ها را تضعیف می‌کند.

پس اگر بتوانیم این constant factor را کاهش دهیم
می‌توانیم سرعت بیشتری از $O(nlgn)$ بگیریم.

مرتب سازی درجی دودویی

- پیچیدگی زمانی insertion sort برابر با $O(n^2)$ است و constant factor آن بسیار بسیار پایین است چون اولا inplace عمل می‌کند (پس نیازی به فضای اضافه ندارد) و ثانیا فقط بین عناصر آرایه پیمایش انجام می‌دهد (پس Locality of Reference هم در آن بسیار خوب است و پردازنده می‌تواند داده‌ها را کش کند.)
- در تحلیل‌های انجام شده روی الگوریتم‌ها، این الگوریتم روی تعداد ورودی ۶۴ و پایین‌تر از الگوریتم‌های دیگر مرتب سازی سریع‌تر عمل می‌کند.
- الگوریتم binary insertion sort بجای جستجوی خطی در آرایه (با پیچیدگی $O(n)$) در آن جستجوی دودویی انجام داده و در زمان لوگاریتمی ($O(\lg n)$) مکان صحیح آیتم را پیدا می‌کند (علت استفاده از این الگوریتم در ادامه روشن خواهد شد.)