جزوه جلسه پنجم داده ساختارها و الگوریتم

۱۱ مهر ۱۴۰۰

فهرست مطالب

۱ مقایسه الگوریتم های مرتب سازی(درجی و ادغامی)

۱. زمان اجرا در بدترین حالت و حالت میانگین:
مرتب سازی ادغامی در زمان $O(nlogn)$ آرایه به طول n را مرتب میکند در حالی که همان آرایه در زمان $O(n^2)$ توسط الگوریتم درجی مرتب میشود.

۲. زمان اجرا در بهترین حالت:
در بهترین حالت (یک آرایه مرتب به عنوان ورودی به الگوریتم داده شود) الگوریتم ادغامی یک آرایه را در زمان $O(nlogn)$ مرتب میکند که این زمان برای الگوریتم درجی برابر $O(n)$ است.

۳. حافظه مصرفی (حافظه اضافی به جز حافظه مخصوص آرایه):
مرتب سازی درجی به صورت in-place انجام میشود و به همین دلیل حافظه اضافی آن برای ایندکس های مخصوص پیمایش آرایه و swap عناصر است. با این تفاسیر حافظه مورد نیاز از مرتبه $O(1)$ است. اما مرتب سازی ادغامی نیازمند یک حافظه اضافی به دلیل تقسیم آرایه به زیر آرایه های کوچک و همچنین ادغام آنهاست. از این رو حافظه مورد نیاز در این الگوریتم $O(n)$ است. توجه شود که حافظه از مرتبه $O(n)$ کافی است و در هر مرحله از الگوریتم که به صرت بازگشتی اجرا میشود نیاز به حافظه جدید نداریم. برای کم کردن حافظه میتوان از الگوریتم های کمکی دیگر مانند External Sort استفاده کرد اما اینکار موجب افزایش زمان اجرای مرتب سازی به اندازه ۲ یا ۳ برابر میشود و حافظه مصرفی از $O(n)$ به $O(1)$ میرسد.

۲ الگوریتم های مرتب سازی مورد استفاده در زبان های برنامه سازی

۱. C/C++
برای داده های پایه(Primitive Data) و اشیا(Objects) از الگوریتم مرتب سازی Quick Sort استفاده میشود.

۲. Python
در این زبان برای هر دوی داده های پایه و اشیا از الگوریتم مرتب سازی Tim Sort استفاده میشود.

۳. Java
این زبان برای مرتب سازی داده های Primitive از الگوریتم Quick Sort و برای Objectها

از الگوریتم Tim Sort استفاده میکند.

۱.۲ Tim Sort

این الگوریتم در سال ۲۰۰۲ توسط Tim Peter در فضای صنعتی و توسط زبان Python پیاده سازی شده است. یک آرایه طی مراحل زیر توسط این الگوریتم که خود ترکیبی از الگوریتم های مرتب سازی درجی دودویی و ادغامی است Sort میشود:

۱. قسمت های صعودی و نزولی در آرایه پیدا میشوند.
۲. قسمت های نزولی برعکس شده و صعودی میشوند.
۳. قسمت های صعودی باهم ادغام میشوند.
۴. اگر مجموع طول ۲ قسمت ادغام شده کمتر از ۶۴ باشد با الگوریتم درجی و در غیر اینصورت با الگوریتم ادغامی مرتب میشوند.

۳ داده ساختارها یا Data Structures

دو مفهوم ابتدایی در مورد ساختمان های داده شرح داده شده است:

۱. Interface/Abstract Data Type: این اصطلاح که با نام واسط نیز شناخته میشود مربوط به مشخصات داده ساختار و ویژگی های آن است؛ یعنی چه داده هایی نگه داری میشوند و چه عملیاتی روی آنها انجام میشود. به عبارت دیگر این اصطلاح همان صورت مسئله است.
۲. Data Structure: این اصطلاح مربوط به پیاده سازی داده ساختار با ویژگی های واسط است. در این قسمت درمورد چگونگی نگه داری داده ها و الگوریتم هایی که روی داده ها پیاده میشوند صحبت میشود. این اصطلاح همان راه حل مسئله است.
هر داده ساختار یه واسط مخصوص به خود را دارد که با توجه به ویژگی های همان واسط، داده ساختار موردنظر پیاده سازی میشود.

اینترفیس های مورد بحث در این درس به دو بخش تقسیم میشوند:
۱. واسط های دنباله ای: این واسط ها برای داده ساختار هایی که ترتیب آنها مهم است تعریف میشود.
۲. واسط های مجموعه ای: این واسط بر خلاف واسط های دنباله ای، برای داده هایی است که ترتیب ذخیره سازی آنها مهم نیست.
واسط های دنباله ای، خود به دو دسته ۱.واسط های دنباله ای ایستا و ۲.واسط های دنباله ای پویا تقسیم میشوند.

۱.۳ واسط های دنباله ای ایستا

این واسط ها برای نگه داری دنباله ای به طول n به کار میرود که عملیات زیر نیز در آن تعریف شده است:
len(): طول دنباله را برمیگرداند که عدد ثابت n است.
seq-iter(): کل دنباله را برمیگرداند
right()/left(): عضو اول/آخر را برمیگرداند
at(i): عنصر iام را برمیگرداند
set-at(i, x): عنصر x را در جایگاه iام قرار میدهد

۲.۳ واسط های دنباله ای پویا

این واسط ها همانند واسط های ایستا هستند با این تفاوت که تعداد اعضای دنباله ای که میخواهیم آنرا ذخیره کنیم فیکس نیست. تمام عملیات موجود برای واسط ایستا برای این واسط نیز تعریف میشود بعلاوه:
insert-at(i, x): عنصر x را در جایگاه iام وارد میکند.
insert-right(x)/insert-left(x): عنصر x را به ابتدا/انتهای دنباله وارد میکند.
delete-at(i)/delete-right()/delete-left(): عنصر اول/آخر/iام را حذف میکند.
تمامی عملیات فوق در یک آرایه به راحتی انجام میشود اما در یک لیست پیوندی بعضی عملیات به سادگی آرایه نیست.

لیست پیوندی: یک لیست پیوندی شامل تعدادی شی است که هر شی به شی بعدی خود در حافظه اشاره میکند. در لیست پیوندی دسترسی به عضو اول یا آخر مانند آرایه در زمان ثابت $O(1)$ انجام میشود اما عملیاتی مانند at(i) در زمان ثابت انجام نمیشوند، زیرا دسترسی مستقیم به آنها موجود نیست و باید از عضو اول یا آخر به آن رسید.

۳.۳ برخی واسط های مهم

۱.۳.۳ واسط **Stack**

اشیا به ترتیب وارد stack میشوند و روی هم قرار گرفته و در هر لحظه تنها به بالاترین عضو دسترسی داریم(Last In First Out). عملیاتی که در این واسط قابل انجام است عبارتند از:

۱. top(): معادل عمل right() در آرایه است و اجازه دسترسی به بالاترین عضو را میدهد.
۲. push(): معادل عمل insert-right() در آرایه است که یک عضو جدید را در بالای استک قرار میدهد.
۳. pop(): معادل عمل delete-right() در آرایه است که بالاترین عضو را از استک حذف میکند.

۲.۳.۳ واسط **Queue** یا صف یکطرفه

عناصر ورودی به ترتیب ورود، از صف خارج میشوند(First In First Out). عملیات تعریف شده در صف یک طرفه:

۱. enqueue(): معادل عمل insert-left() در آرایه است که یک عنصر به صف اضافه میکند.
۲. dequeue(): معادل عمل delete-right() در آرایه است که قدیمی ترین عضو را خارج میکند.

۳.۳.۳ واسط **Deque** یا صف دوطرفه

اشیا از هر دو طرف میتوانند وارد و یا خارج شوند و محدودیتی از این بابت وجود ندارد. عملیاتی که برای Deque تعریف شده اند:(همانند عملیات آرایه ها)

۱. delete-right()
۲. delete-left()
۳. insert-right()
۴. insert-left()