جزوه جلسه ششم داده ساختارها و الگوریتم

۱۱ مهر ۱۴۰۰

فهرست مطالب

۱ پیاده سازی داده ساختار های Stack، Queue و Deque

۱. استفاده از لیست پیوندی:
اینترفیس های ذکر شده برای داده ساختار های بالا به راحتی توسط لیست پیوندی قابل پیاده سازی هستند اما در مواردی مانند دسترسی به عنصر iام داده ها در زمان ثابت $O(1)$ امکان پذیر نیست.
۲. آرایه با سایز متغیر:
تغییر طول آرایه به منزله allocate کردن یک فضای جدید و انتقال آرایه قبلی به آرایه جدید است که این کار هزینه زمانی زیادی دارد. از طرفی سایز یک آرایه به هر میزان نمیتواند بزرگ باشد و مطلوب این است که برای یک آرایه به طول n فضای مصرفی برابر $O(n)$ باشد.
حال شیوه پیاده سازی یک داده ساختار با اینترفیس های ذکر شده (در جلسه قبل) را ذکر میکنیم. داده ساختار های vector در زبان c++، list در python و array list در java اینگونه پیاده سازی شده اند:
- به جای آرایه ای به طول n، آرایه ای به طول $\Theta(n)$ میگیریم.
- هر وقت آرایه پر شد، آرایه جدید به طول ۲ برابر آرایه قبل میگیریم و آرایه قبل را در ابتدای آرایه جدید کپی میکنیم.
پس با تفاسیر فوق اگر سایز آرایه قبل از درج توانی از ۲ باشد، درج عضو جدید در زمان $O(n)$ و درغیر اینصورت در زمان $O(1)$ انجام میشود. برای تحلیل زمانی عملیات درج نیز داریم: (اگر فرض کنیم $O(1) = 1$)

$$1 + 2 + 4 + 1 + 8 + 1 + 1 + 1 + 16 + 1 + 1 + ... \geq 5n = \Theta(n)$$

با تحلیل سرشکن میتوان نتیجه گرفت:

$$\Theta(n)/n = O(1)$$

پس میتوان ادها کرد هزینه درج بصورت سرشکن برابر $O(1)$ است.
- برای اینکه حافظه مصرفی $O(n)$ باقی بماند، وقتی اعضای آرایه به اندازه $1/4$ طول آن شدند، آرایه جدید به طول نصف آرایه میگیریم و اعضا را به ارایه جدید منتقل میکنیم.
بدیهی ایست که اگر مقدار یک چهارم، برابر یک دوم میشد، هنگامی که نصف آرایه پر بود، درج یک عضو و سپس حذف آن به دفعات زیاد، هزینه زمانی بالایی در پی داشت.
مجددا مانند تحلیل سرشکن برای درج میتوان نشان داد هزینه زمانی حذف عضو به صورت سرشکن برابر $O(1)$ میباشد

۲ واسط های مجموعه ای

یک واسط مجموعه ای، مجموعه ای مانند S را نگهداری میکند که هر عضو(شئ) از مجموعه یک کلید نیز دارد.

۱.۲ اینترفیس مجموعه‌ای ساده یا ایستا

۱. find-by-key(key): شی با کلید key را درصورت وجود برمیگرداند.
۲. iter(): اشیا را با تریتبی دلخواه برمیگرداند.

۲.۲ اینترفیس مجموعه‌ای پویا

تمام عملیات واسط مجموعه‌ای ایستا بعلاوه:
۱. insert(key, value): شئ value با کلید key را وارد مجموعه میکند. اگر شئ با کلید key وجود داشت، حذف میشود.
۲. delete-by-key(key): شئ با کلید key را در صورت وجود پاک میکند.

۳.۲ اینترفیس مجموعه‌ای اشیا مرتب

تمام عملیات واسط مجموعه ای ایستا بعلاوه:
۱. find-next(key): شئ x عضو S را با حداقل کلید بزرگتر از key را برمیگرداند.
۲. find-prev(key): شئ x عضو S با حداکثر کلید کوچکتر از key را برمیگرداند.
۳. find-next$(-\infty)$ = find-min()
۴. find-prev(∞) = find-max()
۵. ordered-iter(): مجموعه S را به ترتیب کلیدها برمیگرداند.

۴.۲ اینترفیس مجموعه‌ای پویا و مرتب

تمام عملیات واسط ایستای مرتب و پویا بعلاوه:
۱. delete-min(): حذف شئ x با کوچکترین کلید
۲. delete-max(): حذف شئ x با بزرگترین کلید

۳ واسط صف اولویت یا Priority Queue

در این داده ساختار اشیا به ترتیب اولویت در صف قرار میگیرند و داده با بیشترین اولویت در دسترس است. عملیات تعریف شده در این واسط به شرح زیر هستند:

۱. len(): تعداد اشیا را برمیگرداند.
۲. insert(key, value): شئ value را با کلید key وارد صف میکند.
۳. find-max()
۴. delete-max()

اگر صف اولویت را با آرایه عادی پیاده سازی کنیم، در اینصورت مرتبه زمانی عملیات به ترتیب زمان $O(1)$، $O(1)$(بصورت سرشکن)، $O(n)$ و $O(n)$ است.
اما اگر این پیاده سازی با آرایه مرتب انجام شود به ترتیب داریم: $O(1)$، $O(n)$، $O(1)$ و $O(1)$(بصورت سرشکن).