جزوه جلسه بیست و هشتم داده ساختارها و الگوریتم

۷ دی ۱۴۰۰

فهرست مطالب

۱ تحلیل سرشکن

در طول ترم، در مباحثی مانند:

۱. آرایه پویا

۲. جدول درهم سازی پویا

۳. هرم فیبوناچی

۴. مجموعه های مجزا

با تحلیل سرشکن سروکار داشتیم.
تحلیل سرشکن یعنی یک زمان فرضی برای عملیات در نظر می گیریم که برابر با زمانهای واقعی نیستند اما مجموع این زمانها از اول تا آخر به هم ربط دارد.
در اصل، در تحلیل سرشکن، هزینه زیاد یک عملیات را بین عملیات دیگر پخش یا سرشکن می کنیم.
در ادامه، روش های تحلیل سرشکن را بررسی می کنیم.

۱.۱ روش تجمیعی یا انبوهه

هزینه سرشکن هر عملیات، برابر است با مجموع هزینه ها تقسیم بر تعداد عملیات.
اگر بخواهیم کمی پیچیده تر به این حالت نگاه کنیم، می توانیم یک درخت AVL را در نظر بگیریم که هزینه سرشکن درج در آن $O(logn)$ و هزیته سرشکن حذف برابر با 0 است!
می دانیم هزینه واقعی هر دو عملیات برابر با $clogn$ می باشد. درج را در $2clogn$ و حذف را در 0 می توان طبق تعریف بالا انجام داد. به بیان دیگر با فرض خالی بودن درخت در ابتدا، به هنگام درج هر عنصر در درخت، هزینه حذف آن را نیز می پردازیم.
با توجه به مطالب ذکر شده، استفاده از این متد زمانی پیشنهاد می شود که تعداد عملیات تعریف شده برای داده ساختار فقط یکی باشد و با زیاد شدن عملیات، باگ های این متد پیدا می شوند.
به منظور پوشش ایرادات روش قبلی، روش حسابداری معرفی شده است.

۲.۱ روش حسابداری

برای هر عمل، علاوه بر هزینه اصلی خود، یا هزینه بیشتری برای پس انداز در نظر می گیریم و یا هزینه کمتری در نظر می گیریم و از پس انداز استفاده می کنیم. توجه شود که پس انداز نباید منفی شود.

برای مثال درخت AVL ذکر شده در بخش قبل، اگر برای هر درج علاوه بر زمان $clogn$ خودش، زمان $clogn$ را نیز پس انداز کنیم، کافی است اثبات کنیم هزینه حذف را می توان کاملا از پس انداز ها خرج کرد. این ادعا نیز اثبات می شود؛ زیرا هر درج، یک حذف معادل دارد.
همچنین در آرایه پویا می توان هزینه هر درج معمولی بدون نیاز به افزایش اندازه آرایه را چند $O(1)$ در نظر گرفت و هزینه کپی و بزرگ کردن آرایه را از پس اندازها خرج کرد. همچنین باید توجه کرد جمع پس اندازها هزینه کپی و بزرگ کردن آرایه را بدهد. بدین منظور زمان هر درج معمولی را ۳ یا ۵ برابر زمان معمول ($O(1)$) در نظر می گیریم.

۳.۱ روش شارژ کردن

در این روش مقداری از هزینه هر عمل را روی اعمال قبلی شارژ می کنیم. در این حالت، هزینه سرشکن هر عمل برابر است با هزینه واقعی بعلاوه هزینه ای که در آینده رو آن شارژ خواهد شد منهای هزینه ای که روی بقیه شارژ می کند (هزینه را دیگران بپردازند).
برای مثال در آرایه پویا:

- هنگام دو برابر کردن آرایه هزینه واقعی خودش زیاد است و هزینه ای که روی بقیه شارژ می کند نیز زیاد است به همین دلیل هزینه سرشکن آن کم می شود.
- در بقیه موارد نیز هزینه واقعی و هزینه ای که در آینده روی آن شارژ خواهد شد هر دو کم هستند و هزینه سرشکن نیز کم است.

حال وجود چندین عملیات (برای مثال درج و حذف در آرایه پویا) را می توان با این روش تحلیل کرد.

۱.۳.۱ شمارشگر بیتی یا **Binary Counter**

در یک شمارشگر n-بیتی، فرض کنید در state زیر هستیم:
$XX..XX0111...111$
در کلاک بعدی باید در state زیر باشیم:
$XX..XX1000...000$
اگر تعداد یک های حالت اول و صفر های حالت دوم در سمت راست اعداد را m تا فرض کنیم، می توانیم هزینه بیت $m+1$ام را روی بقیه شارژ کنیم تا هزینه شمارش در هر کلاک برابر با $O(1)$ باشد. بدین منظور هزینه هر شارژ را 1 در نظر می گیریم (هزینه اضافی برای هر عملیات یک واحد است). پس برای حالت بالا هزینه شمارش برابر است با:
$2 + m + 1 - (m - 1) = 2$

برای بقیه حالت ها نیز داریم:

$1 + 1 + 0 = 2$

۴.۱ تابع پتاسنیل (قوی ترین روش تحلیل سرشکن)

از تایع پتانسیل برای نامنفی کردن یالهای منفی در الگوریتم دکسترا و همچنین در مجموعه های مجزا استفاده کرده ایم.
تابع پتانسیل Φ تابعی است از وضعیت داده ساختار به اعداد نامنفی.
در این روش هزینه سرشکن برابر است با هزینه واقعی بعلاوه تغییرات تابع پتانسیل ($\Phi_1 - \Phi_0 = \Delta\Phi$ که Φ_1 برابر با مقدار جدید تایع پتانسیل و Φ_0 برابر با مقدار قدیم آن است).
تعریف تایع پتاسنیل نیز ساده نیست.
همچنین برای هر i، مقدار Φ_i باید نامنفی باشد.
اگر مجموع هزینه های واقعی را با $\sum_{i=1}^{n} c_i$ نشان دهیم و مجموع هزینه های سرشکن را با $\sum_{i=1}^{n} c_i + (\Phi_i - \Phi_{i-1})$ نشان دهیم با فرض بزرگتر یا مساوی بودن مجموع هزینه های سرشکن از مجموع هزینه های واقعی، داریم:
$\sum_{i=1}^{n} c_i + (\Phi_i - \Phi_{i-1}) = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0 \geq \sum_{i=1}^{n} c_i \implies \Phi_n - \Phi_0 \geq 0 \implies \Phi_n \geq \Phi_0$
که با فرض نامنفی بودن Φ_0، برای هر n، Φ_n نامنفی است.
با بررسی توابع پتانسیل پیشنهادی زیر برای شمارشگر بیتی، بهترین تابع پتانسیل برابر است با تعداد کل یک ها در هر لحظه.

- تعداد صفر های ابتدا
- تعداد صفر های کل
- تعداد یک های ابتدا
- تعداد یک های کل

۱.۴.۱ مثال هایی از توابع پتانسیل سرشکن

- شمارشگر بیتی: تعداد کل یک های شمارشگر
- درج عنصر iام در آرایه پویا: $2i - 2^{\lceil log(i) \rceil}$
- مجموعه های مجزا: $\sum \Phi(x)$ for all x
  تابع Φ نیز برای x هایی که ریشه نباشند و یا رنک آنها بزرگتر از صفر باشد به

شکل:

$$(\alpha(x) - level(x)).rank(x) - iter(x)$$

و برای بقیه x ها به شکل:

$$\alpha(x).rank(x)$$

تعریف می شود.