ساختمان داده و الگوریتم ها

تمرین ششم - تحلیل سرشکن

مجید فریدفر، فاطمه کرمی

تاریخ تحویل: ۱۴۰۲/۱۰/۲۴

۱. استک k تایی — ۱۰ نمره

فرض کنید روی یک ساختمان داده استک که اندازه آن هرگز از k بیشتر نمی‌شود تعدادی عملیات انجام می‌دهیم و پس از هر k عملیات برای $Backup$ گرفتن از استک یک کپی از کل عناصر آن می‌گیریم. با اختصاص دادن هزینه سرشکن مناسب به هر عملیات استک (شامل $Push$، Pop و $Copy$) نشان دهید که انجام n عملیات روی استک زمان اجرای $O(n)$ دارد.

پاسخ :

به ازای هر عملیات $Push$ و Pop به اندازه ۲دلار و به ازای هر عملیات $Copy$ به اندازه ۰ دلار حساب را شارژ می‌کنیم. هر بار که عملیات $Push$ را انجام دادیم ۱ دلار برای انجام آن هزینه می‌کنیم و ۱ دلار باقیمانده را در عنصری که $Push$ کردیم نگه می‌داریم. همچنین هر بار که عملیات Pop را انجام دادیم ۱ دلار برای انجام آن هزینه کرده و ۱ دلار باقیمانده را در خود استک نگه می‌داریم. می‌دانیم اندازه استک هرگز از k بیشتر نمی‌شود پس حداکثر هزینه برای هر بار $Copy$ کردن استک k دلار است. همچنین بین هر دو بار که عملیات $Copy$ را استفاده می‌کنیم، k عملیات $Push$ یا Pop انجام شده است پس به مقدار k دلار در عناصر داخل استک و خود استک ذخیره کرده‌ایم. حال برای انجام عملیات $Copy$ از همین k دلار ذخیره شده استفاده می‌کنیم. پس هزینه سرشکن هر عملیات $O(1)$ است و زمان اجرای n عملیات نیز $O(n)$ خواهد بود.

۲. صلوات counter پیشرفته — ۱۵ نمره

شایان که عاشق درس مدار منطقی است، برای خرید تعدادی ترانزیستور و گیت به خیابان پشت‌شهرداری رفته. هنگامی که به دنبال لوازم مورد نیازش می‌گشت در یکی از مغازه‌ها وسیله‌ای دید که برایش خیلی جالب بود: یک صلوات counter پیشرفته! این دستگاه روش جالبی برای نمایش اعداد دارد. برخلاف بقیه‌ی counter های موجود در بازار که از نمایش باینری اعداد برای محاسباتشان استفاده می‌کنند، در این شمارنده از نمایش فیبوناچی آن‌ها استفاده شده (دقت کنید که هر عددی را می‌توان به صورت مجموعی از جملات دنباله‌ی فیبوناچی نوشت). به عبارت دیگر به جای نگه‌داری آرایه‌ای از bit ها، از آرایه‌ای از fit ها استفاده می‌شود که اگر فیت i ام، صفر باشد به این معنی است که جمله‌ی i ام دنباله‌ی فیبوناچی در ساختن عدد n ضریب صفر دارد، اگر یک باشد یعنی این ضریب یک است و جمله‌ی i ام دنباله در ساختن عدد n به کار رفته است. برای مثال دنباله‌ی ۱۰۱۱۱۰ عدد ۱۴ $= ۱ + ۲ + ۳ + ۸ = F_۲ + F_۳ + F_۴ + F_۶$ را نشان می‌دهد. به شایان کمک کنید هزینه‌ی سرشکن عمل increment در این صلوات counter را به دست بیاورد. از روش accounting استفاده کنید.

پاسخ :

این مسئله را به روش accounting حل می‌کنیم. در ابتدا لازم است گزاره‌ی زیر را اثبات کنیم.

* در هر بار انجام عملیات increment دقیقا یک فیت مقدارش از ۰ به ۱ تغییر می‌کند.

اثبات: فرض کنید در حال حاضر صلوات counter عدد n را نشان می‌دهد که آرایه‌ی $F(n)$ نمایش فیبوناچی آن است. سمت راست‌ترین رقم ۰ را در این آرایه در نظر بگیرید. اگر این رقم در جایگاه ۰ ام یا ۱ ام باشد، کافی است آن را یک کنیم (جایگاه‌های ۰ام و ۱ام، نماینده‌ی دو جمله‌ی اول دنباله‌ی فیبوناچی ($F_۰ = F_۱ = ۱$) هستند). به این ترتیب به عدد $n + ۱$ می‌رسیم. پس در این حالت درستی گزاره به وضوح مشخص است.

حالا فرض کنید ایندکس این فیت (i) ، بیشتر یا مساوی ۲ است. می‌دانیم که مقدار تمامی فیت‌های ۰ ام تا $i - ۱$ ام، یک است. طبق خاصیت دنباله‌ی فیبوناچی، می‌توانیم فیت‌های $i - ۱$ ام و $i - ۲$ ام را صفر و بیت i ام را یک کنیم. مقدار عدد هیچ تغییری نمی‌کند و n باقی می‌ماند. این کار را برای بیت‌های $i - ۳$ ام و $i - ۴$ ام هم می‌توانیم انجام دهیم. با ادامه‌ی این روند، اگر مقدار i زوج باشد، در نهایت،

مقدار فیت ۰ ام و ۱ ام صفر می‌شود و اگر فرد باشد، مقدار بیت ۱ ام. حالا کافی است اولین فیتی که مقدارش ۰ است را ۱ کنیم (اگر i زوج باشد، مقدار فیت ۰ ام و اگر فرد باشد، مقدار فیت ۱ ام). با توجه به این نکته که هر دو فیت ۰ ام و ۱ ام، نمایش‌دهنده‌ی جملات اول دنباله‌ی فیبوناچی‌اند (۱)، با انجام این کار عدد n را تبدیل به عدد $n+1$ می‌کنیم. هم‌چنین واضح است فقط بیت i ام، مقدارش از ۰ به ۱ تغییر کرده، که درستی گزاره‌ی گفته شده را نشان می‌دهد.

حالا کافی است در هر بار انجام عملیات increment ، ۲ تا در حساب ذخیره کنیم. یکی برای فیت ۰ ای که مقدارش ۱ شده، و یکی هم برای ادامه‌ی کار. زمانی که می‌خواهیم مقدار این بیت را از ۱ به ۰ تغییر دهیم.

به وضوح هزینه‌ی سرشکن این عملیات $o(1)$ است ($\frac{2n}{n} = 2$)

۳. چشم بهم بزنی، هاشمی نیست! ۱۵ نمره

صادق مدتی است از ماتریکس خارج شده. دنیا به مکان ترسناکی تبدیل شده است. ساختمان‌ها و خیابان‌های ویران‌شده توسط ربات‌ها، طبیعتِ نابود شده، آسمانی که دیگر رنگ خاکستری به خود گرفته و بدتر از همه، مزارع کِشت انسان... تحمل این همه ویرانی برای صادق خیلی سخت است. برای همین به سرعت شروع به برنامه‌ریزی نقشه‌هایی کرده است تا دنیا را از ربات‌ها پس بگیرد. مورفیوس پس از دیدن پشتکار او، تصمیم گرفت حقوقش را از بقیه‌ی اعضای تیم بیش‌تر کند. اما صادق که خیلی متواضع است، قبول نکرد. مورفیوس، بعد از یک جلسه‌ی طولانی، موفق شد او را متقاعد کند که مدل دریافتی زیر را بپذیرد، که البته خیلی کم‌تر از مقدار پیشنهادی اولیه است:

صادق هر روز، ۱۰۰۰ دلار دریافت می‌کند (مثل بقیه‌ی اعضای تیم)، اما اول ژانویه‌ی هر سال، یک عیدی هم می‌گیرد که برابر با تعداد کل روزهایی است که از ابتدای شروع به کارش تا آخر آن سال کار کرده.

مدتی به همین منوال گذشت، تا این که صادق در ضیافت هالوین، بهمن هاشمی را ملاقات کرد که سال‌های زیادی است از ماتریکس خارج شده -حتی بعضی‌ها می‌گویند اصلا وارد آن نشده! پس از صحبت با او که اطلاعات خیلی زیادی دارد، به موضوع عجیبی پی برد که به شدت ذهن او را درگیر کرد. آقای هاشمی گفت، بعد از انقلاب ربات‌ها در روز سال نوی ۲۰۰۰، به دلایلی شتاب زمین به نحوی تغییر کرد که هر سال سرعت گردش زمین به دور خورشید نصف می‌شود! خوشبختانه او موفق شده برای یک سال جلوی این روند را بگیرد و انتظار می‌رود سال ۲۰۰۱ هم مثل ۲۰۰۰، ۳۶۵ روزه باشد، اما از ۲۰۰۲ به بعد تعداد روزهای سال به صورت نمایی زیاد خواهد شد!

با فرض این که زمانی که صادق از ماتریکس خارج شد دقیقا به اول ژانویه‌ی ۲۰۰۰ برگشت، به صورت سرشکن درآمد روزانه‌ی صادق را از ابتدای خروجش از ماتریکس تا روز n ام از روش aggregate محاسبه کنید.

پاسخ :

در ابتدا لازم است دقت کنید که سال اول و دوم، ۳۶۵ روزه‌اند. سال سوم ۲×۳۶۵ روز (چون سرعت نصف شده، زمان یک دور چرخش کامل به دور خورشید دو برابر می‌شود)، سال چهارم ۴×۳۶۵ و... . پس می‌توان به نتیجه‌ی زیر رسید که اگر c_i درآمد روز i ام باشد. داریم:

$$c_i = \begin{cases} ۱۰۰۰ + i & \text{اگر } i \text{ به فرم } ۳۶۵ * ۲^k \text{ است} \\ ۱۰۰۰ & \text{در غیر این صورت} \end{cases}$$

با استفاده از روش aggregate درآمد n روز اول، برابر است با

$$\begin{aligned} \sum_{i=۱}^{n} c_i &= n \times ۱۰۰۰ + \sum_{j=۰}^{\lfloor \log_۲ \frac{n}{۳۶۵} \rfloor} ۲^j \times ۳۶۵ \\ &= n \times ۱۰۰۰ + ۳۶۵(۲^{\lfloor \log_۲ \frac{n}{۳۶۵} \rfloor + ۱} - ۱) \\ &< n \times ۱۰۰۰ + ۳۶۵ \left(\frac{n}{۳۶۵} \times ۲ - ۱ \right) \\ &= n \times ۱۰۰۰ + ۲ \times n \\ &< n \times ۱۰۰۲ \end{aligned}$$

پس از میانگین گیری به این نتیجه می‌رسیم که $\frac{\sum_{i=۱}^{n} c_i}{n} = ۱۰۰۲$. یا به عبارتی به صورت سرشکن‌شده، c_i از اردر $o(1)$ است. (عدد ثابت)

۴. ساختمان داده‌ی به درد نخور ۱۵ نمره

می‌خواهیم یک ساختمان داده S با اعداد حقیقی و عملیات‌های زیر پیاده‌سازی کنیم:

۱. عملیات $Insert(S, x)$ عنصر x را به S اضافه می‌کند.

۲.عملیات $Delete(S)$، $\lceil \frac{|S|}{۲} \rceil$ تا از بزرگترین عناصر در S را پاک می‌کند.

یک پیاده‌سازی پیشنهاد دهید که هزینه سرشکن هر دو عملیات $O(۱)$ بشود. (راهنمایی: می‌توان در زمان $O(n)$ میانه آرایه‌ای با سایز n را پیدا کرد.)

پاسخ :

ساختمان داده S را با یک آرایه مرتب نشده می‌سازیم. درج کردن در این آرایه زمان $O(۱)$ می‌گیرد و حذف از این آرایه در بدترین حالت زمان $O(|S|)$ خواهد گرفت به این صورت که میانه را در زمان $O(|S|)$ پیدا کرده و سپس با پیمایش آرایه $\lceil |S|/۲ \rceil$ تا از عناصر که از میانه بزرگ‌تر یا مساوی آن هستند را حذف می‌کنیم که در کل زمان $O(|S|)$ می‌گیرد. فرض کنیم ثابت a وجود دارد به طوری که عملیات حذف حداکثر $O(a|S|)$ زمان بگیرد. از آنالیز $Potential$ استفاده می‌کنیم تا هزینه سرشکن هر عملیات را بدست آوریم. تابع $Potential$ به صورت $phi_i = an_i$ است در حالتی که n_i اندازه آرایه قبل از عملیات i ام باشد. اگر عملیات i درج باشد $c_i = ۱$ و $n_i = n_{i-۱} + ۱$ خواهد بود. بنابراین $c_hat_i = ۱ + (phi_i - phi_{i-۱}) = ۱ + a$ که یک عدد ثابت است خواهد بود. در صورتی که عملیات i حذف باشد $c_i <= an_i$ و $n_i <= n_{i-۱}/۲$ است. پس $c_hat_i <= an_i + a(n_{i-۱}/۲ - n_{i-۱}) <= ۰$ خواهد بود. در نتیجه هزینه سرشکن هردو عملیات حداکثر $a + ۱$ است.

۵. DFS پیشرفته ۲۰ نمره

پیچیدگی زمانی فراخوانی تابع زیر را محاسبه کنید.

```
def DFS(v):
    sub[v] = [v]
    for u in children[v]:
        DFS(u)
        for x in sub[v]:
            for y in sub[u]:
                print("Hello\n")
        sub[v] += sub[u]
```

پاسخ :

عملیاتی که این تابع انجام می‌دهد معادل این است که در یک درخت به ازای هر راس i، بین هر دو راس j و k که در دو زیردرخت متفاوت راس i قرار دارند و راس i اولین جد مشترک‌شان است یک یال بگذاریم. این عملیات به ازای هر دو راس متفاوت درخت در نهایت یک یال بین آن‌ها می‌گذارد که در مجموع هزینه انجام آن $O(n^۲)$ است.

۶. زندان مرکزی گاتهام بزرگ ۲۵ نمره

به دنبال سرماخوردگی بروس وین، صابر مدتی است که مسئولیت دستگیری تبهکاران شهر گاتهام را به عهده گرفته است. او که از ابتدا به دلیل کُند بودن عملکرد پلیس با تحویل مجرمان مشکل داشت (بارها جوکر از همین موضوع برای فرار استفاده کرده بود و موفق شده بود!) تصمیم گرفته خودش زندانی را احداث کند تا شروران را بعد از دستگیری در آن نگه دارد. زندان او به تعدادی بخش تقسیم‌بندی شده که بخش

i ام، 2^i تا سلول دارد (بخش ۰ام، بخش ۱ام، بخش ۲ام و...). با توجه به محدودیت زندان‌بان، او مجبور است زندانیان را به نحوی بین این بخش‌ها پخش کند، که هیچ بخش نیمه‌خالی‌ای وجود نداشته باشد (هر بخش یا کاملا خالی است و زندان‌بان ندارد، یا کاملا پر است). هم‌چنین برای این که از طریق دستگاه بت‌ویو بتواند بهتر اطلاعات آنان را آنالیز کند، زندانیان هر بخش را به ترتیب حروف الفبا در سلول‌ها قرار می‌دهد. (بین زندانیان دو بخش متفاوت، لزومی ندارد ترتیبی وجود داشته باشد) اثبات کنید صابر از بروس وین بت‌من بهتری است (نشان دهید هزینه‌ی سرشکن بازداشت یک تبهکار جدید با روش صابر از اردر $O(\log n)$ است. در حالی که می‌دانیم بروس وین این کار را با اردر $O(n)$ انجام می‌دهد!)

فرض کنید محدودیتی روی تعداد بخش‌ها نداریم.

پاسخ :

در ابتدا دقت کنید، با توجه به این موضوع که هیچ بخش نیمه‌خالی نداریم، پس اگر کلا n تا زندانی داشته باشیم، و $a_{m-1}a_{m-2}\dots a_1a_0$ نمایش باینری این عدد باشد، سلول‌های بخش i ام $(i < m)$ پر است اگر a_i برابر یک باشد. در غیر این صورت این بخش خالی است.

برای افزودن یک زندانی جدید، کافی است اولین بخش خالی را پیدا کنیم. فرض کنید i شماره‌ی این بخش باشد. می‌دانیم که تمام i-۱ بخش قبلی، پر هستند. زندانی جدید را در بخش i ام قرار می‌دهیم. سپس آن را با بخش ۰ ام merge می‌کنیم و دوباره در بخش i ام قرار می‌دهیم. حالا دو زندانی در این بخش هستند. یک بار دیگر merge را با بخش ۱ ام انجام می‌دهیم. این کار را برای تمام بخش‌های ۰ ام تا i-۱ ام به ترتیب انجام می‌دهیم تا در نهایت تمامی زندانی‌های این بخش‌ها به همراه زندانی جدید به صورت مرتب شده در بخش i ام قرار بگیرند. می‌دانیم هزینه‌ی انجام عملیات مرج برای دو لیست k تایی برابر است با $k \times 2$. پس هزینه‌ی انجام عملیات افزودن زندانی جدید برابر است با

$$T(n) = \Theta(2^0 \times 2 + 2^1 \times 2 + \dots + 2^{i-1} \times 2) = \Theta(2^{i+1})$$

برای به دست آوردن هزینه‌ی سرشکن این عملیات، فرض کنید با شروع از یک زندان خالی می‌خواهیم n تا زندانی به ترتیب به آن اضافه کنیم. از روش aggregate استفاده می‌کنیم. فرض کنید هزینه‌ی خالی شدن یک بخش با شماره‌ی i برابر است با هزینه‌ی مرج 2^i زندانی آن بخش با 2^i زندانی‌ای که در یک بخش با شماره‌ی بزرگ‌تر قرار دارند (مثلا شماره‌ی .j فرض کنید طبق توضیح گفته شده در بالا، در حال پر کردن این بخش با merge کردن تمام بخش‌های قبلی‌اش هستیم). پس عملا برای پر شدن یک بخش هزینه‌ای نمی‌دهیم. چون هزینه‌ی کل عملیات در مراحل قبلی به وسیله‌ی مرج‌هایی که انجام شده محاسبه شده است. پس به طور کلی در بدترین حالت، برای تغییر وضعیت یک بخش 2×2^i تا هزینه می‌کنیم.

می‌دانیم بخش ۰ ام، n بار تغییر وضعیت می‌دهد. بخش ۱ ام، $\frac{n}{2}$ بار، بخش ۲ ام، $\frac{n}{4}$ بار و...

در نتیجه هزینه‌ی کل عملیات برابر است با

$$T(n) = \Theta(n \times 2^1 + \frac{n}{2}2^2 + \dots + \frac{n}{2^{\log n}}2^{\log n+1}) = \Theta(\log n \times 2n) = \Theta(n \log n)$$

پس هزینه‌ی سرشکن برابر است با

$$\Theta(\log n)$$