Subject: DS تمرین پایانی

Year Month Day

۸۱۰۱۰۱۵۰۲ — محمد طاها جابری

• سوال ①: چون به اندازه‌ی ساختمان داده کمتر از k بسته نمی‌شود و تعداد به طور معمول برای Push کردن و Pop کردن هزینه‌ی ۱ را در نظر گرفت.

حال چون عمل کپی کردن را داریم باید به نوعی هزینه‌ی کپی کردن را از هزینه‌ی Push و Pop کردن بدست آوریم. به همین جهت می‌دانیم که هزینه‌ی کپی کردن بعد از هر k عملیاتی که انجام می‌دهیم به تعداد اعضای ساختمان داده خواهد بود. حال اگر برای Push و Pop کردن هر کدام یک واحد اضافه که در نظر بگیریم یعنی هزینه‌ی Amotize Cost را برابر با ۲ برای هر کدام در نظر بگیریم

و هم چنین هزینه‌ی Amotize کپی کردن را هم برابر با صفر در نظر بگیریم در این صورت

بعد از هر k عملیاتی که انجام می‌دهیم به اندازه‌ی k ذخیره داریم و چون تعداد اعضای آن ساختمان داده حداکثر k خواهد بود بنابراین حتماً در بدترین حالت هزینه‌ی کافی را در روش Accounting خواهیم داشت.

	Amotize Cost	real Cost
Push	۲	1
Pop	۲	1
Copy	۰	k

• هر کدام یک واحد ذخیره می‌کنند.

* می‌توانستیم به طور مثال برای Push کردن هزینه‌ی Copy کردن را آماده کنیم اما مشکلی که وجود داشت این بود که شاید مثلاً k بار یکی در میان یک عنصر را حذف و عنصر دیگری را اضافه می‌کردیم. در این صورت نمی‌توانستیم عملیات backup گرفتن را انجام دهیم چون هزینه‌ی کافی را در صندوق نداشتیم.

NoteBook Sarv

سوال ۲ : می دانیم که هر عددی را می‌توان به صورت مجموعی از اعداد فیبوناچی نوشت؛

پس الگوریتم را این گونه می‌نویسیم. می‌دانیم که در اعداد فیبوناچی جمع هر دو عدد برابر با عدد بعدی خواهد شد.
۸=۳+۵ پس یعنی اگر هر بیت را معادل با یک عدد فیبوناچی در نظر بگیریم در این صورت
هر ۲ تا بیت کناری معادل با یک بیت کناری سمت چپی خواهد شد.
مثلاً اگر بیت ۳ و ۵ که کنار هم هستند را روشن کنیم می‌توانیم هر دو را خاموش کرده و به جای آن
بیت شماره‌ی ۸ را روشن کنیم. پس می‌توان نتیجه گرفت که هیچ دو بیت ۱ ای در کنار هم نباید ① باشند.

مثلاً مثال خود سوال را به این صورت هم می‌توان نوشت؛

10 1110

110010

$\boxed{1000010}$ → شکل صحیح که

حال می‌توان الگوریتم را این گونه تعریف کرد
که ما برای اضافه کردن یک عدد باید یکی از بیت
های یک را از صفر به یک تبدیل کنیم (لازم به ذکر است که هر دو تا بیت کنار هم نمی‌توانند یک باشند)
حال اگر دو بیت کنار هم که در ابتدا هر دو تا یک بودند آن ها را صفر کرده و بیت بعدی آن ها را
یک می‌کنیم که بیت بعدی آن ها هم حتماً صفر بوده است مطابق با فرض
پس در کل در هر عمل تعدادی بیت از یک به صفر تبدیل می‌شوند و یک بیت از صفر به یک تغییر پیدا
خواهد کرد. حال اگر به روش accounting بخواهیم حساب کنیم.
برای هر کدام از تبدیل‌ها از صفر به یک شدن اگر ② هزینه کنیم. برای هر بار تبدیل شدن از یک به صفر
هزینه. در این صورت برای هر بار increment کردن تنها یک هزینه‌ی به اندازه‌ی ۲ داریم.
پس در کل هزینه‌ی یک بار Count کردن معادل با هزینه‌ی یک بیت از صفر به یک
است که ② می‌باشد.

Subject:

Year Month Day

سوال ② : می‌توان این گونه نشان داد چون که از اول ژانویه شروع کرده بود : سال ۲۰۰۰

۲۰۰۰ [۳۶۵] $\rightarrow 1000n + 0$

۲۰۰۱ [۳۶۵] $1000n + 365$

۲۰۰۲ [۷۳۰] $1000n + 730 + 365$

۲۰۰۳ [۱۴۶۰] $1000n + 1460 + 730 + 365$

⋮

○ و به همین صورت ادامه پیدا خواهد کرد.

حال باید به روش Aggregate هزینه را حساب کنیم:

هزینه کلی تا روز n ام می‌توان باید که در چه سالی باشد. به طور مثال اگر که سال ۲۰۰۳ باشد به این صورت

$$1000n + 365a(1+2+4)$$

$$= 1000n + 365a(7) = 1000n + 365(8-1)$$ خواهد بود.

پس می‌توان از آن اگر استفاده کرد دریافت که تقریبا در هر سالی که باشد به اندازه تعداد روزهای آن سال تقریبا دو برابر است. و با توجه به تابع آن سال دریافت کرده است (با ارزش سال‌های پیش) به اندازه‌ی حدود ② برابر مقدار روزهای این سال است.

یعنی حال $(1+2+4+\dots+2^i) = 2^{i+1}-1$ که باید در ۳۶۵ ضرب شود. برای سال i ام

$$365(1+2+\dots+2^i) = (2^{i+1}-1)365$$

یعنی به طور حدودی ② برابر پول به دست می‌آورد. پس می‌توان این گونه هزینه را حساب کرد: چون سقف بالا را تقلید کرده‌ایم

$$365a\left(2^{\left\lceil \log_2 \frac{n}{365} \right\rceil}-1\right) + \underbrace{1000n}_{\text{مقدار دریافتی پول روزانه}} \le 365\left(2^{\log_2 \frac{n}{365}+1}-1\right)+1000n$$

$$\le 365\left(2\left(\frac{n}{365}\right)-1\right)+1000n \le 365\left(\frac{2n}{365}\right)+1000n = 1002n$$

پس به صورت Amotized $\boxed{1002n}$ در هر روز درآمد دارد.

SarvNoteBook

سوال ④ : برای این سوال می‌توانیم از لینک لیست استفاده کنیم.

چون حذف کردن در لینک لیست از $O(1)$ انجام می‌شود. برای بدست آوردن آرایه که باید هر دو اینک بست به صفت دهیم. پس راحت‌تر هستیم که از لینک لیست استفاده کنیم.
حال ابتدا باید هزینه‌ی پاک کردن k تا عنصر بزرگتر را محاسبه کنیم که برای این کار ابتدا باید این لینک لیست را پیمایش کنیم و عنصر میانه را پیدا کنیم. می‌دانیم که این کار از $O(n)$ طول خواهد کشید و عنصر میانه را پیدا کنیم چون می‌خواهیم عناصر بزرگتر از میانه را در این لینک لیست حذف کنیم. پس از آن یک بار دیگر پیمایش می‌کنیم و عناصر بزرگتر را حذف می‌کنیم. در کل حذف کردن عناصر از $O(n)$ طول خواهد کشید.

حذف کردن ← $O(n)$

حال باید برای اضافه کردن عنصر هزینه‌های هزینه کنیم که هزینه‌ی حذف کردن را جبران کند. برای این کار اگر kn هزینه‌ی حذف کردن باشد یعنی اگر لازم باشد k بار آرایه را پیمایش کنیم برای هر عنصری که اضافه می‌شود باید $2k$ اضافه‌تر هزینه از آن بگیریم. یعنی $2k+1$ هزینه‌ی Amortize خواهد بود. $2k$ را باید بگیریم چون هر بار طول لینک لیست نصف خواهد شد. و برای این که هزینه‌ها جبران شود به ازای هر بار پیمایش برای هر کدام از عناصر موقع اضافه شدن ② تا هزینه اضافه‌تر دریافت می‌کنیم.
حال اگر $k=1$ را در نظر بگیریم هزینه‌ی Amortize برای اضافه کردن برابر با ۳ و همین طور هزینه‌ی Amortize برای حذف کردن برابر با ۰ در نظر گرفته می‌شود.

اگر به روش Aggregate بخواهیم باید بدترین حالت را در نظر بگیریم و برای هر کدام از insert ها یک هزینه داریم و برای هر کدام از delete کردن ها هم n هزینه داریم بسته به طول آرایه
پس برای یک لینک لیست از ابتدا تا انتها این ترتیب را خواهیم داشت:
چون هر بار طول لینک لیست نصف می‌شود ← k را باید در نظر گرفت

$$\underbrace{1+1+1+\dots+1}_{\text{insert کردن } n \text{ تا}} + n + \frac{n}{2}+\frac{n}{4}+\dots+\frac{n}{2^{\log_2 n}}$$

$$\frac{n+n+n(1+\frac{1}{2}+\dots)}{n} = \frac{3n}{n} = ③$$

که با این روش هم ۳ به دست آمد.

$n+\log^2 n$ چون جمع تنها $\log_2 n$ بار می‌توان حذف کرد.

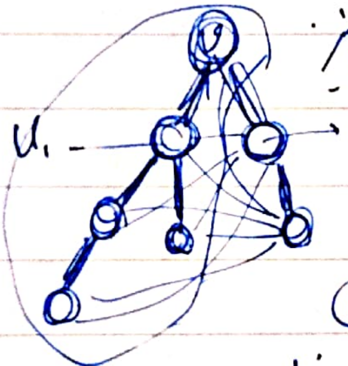
Subject

Year Month Day

سوال ۵ : اثبات می‌شود که در این DFS خاص که باید بدون یک درخت یا یک جنگل حتماً زده شود (چون اگر گراف غیر درخت باشه در آن صورت بی نهایت بار درخت‌ش تکرار می‌شود) به تعداد یال های یک گراف کامل هزینه خواهد داشت.

یعنی $n(n-1)$ ← اگر n تعداد راس ها باشه :

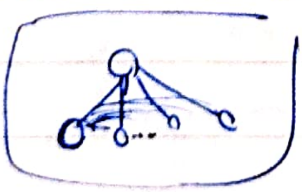
برای اثبات اگر یک راس u را در نظر بگیریم. بعد از DFS زدن کامل روی درخت تا راس u یعنی به طور مثال اگر ۲ راس فرزند u_1 و u_2 را داشته باشه. اول خود راس u (آنلاک در آن) باید در تعداد زیر درخت های راس u_1 ضرب شود و بعد راس u با زیر درخت u_1 ادغام و دوباره با زیر درخت راس u_2 ضرب شود. می‌توان ضرب کردن را این گونه تصور کرد که یال از هر راس به تمام راس های دیگر یک یال رسم می‌کنیم.



و همین گونه اگر برای دیگر راس ها ادامه بدیم به یک گراف کامل خواهیم رسید. که تعداد یال های آن هزینه‌ی مورد نظر ما است پس در مجموع $O(n(n-1))$ هزینه خواهیم داشت. که همان $O(n^2)$ خواهد بود.

در این سوال تعداد print ها رو معادل با یک یال در نظر گرفتیم و چون عناصر می‌گیرد ۲ بخش که با هم اشتراکی از تمام ندارند و همه باید به هم متصل بشوند (ضرب شوند) می‌توانیم به عنوان یک درخت کامل از گراف در نظر بگیریم.

در حالت پایه هم می‌توان یک راس با تعدادی فرزند بزرگ را در نظر گرفت که دو به دو هم به هم یال نیاز خواهند داشت.

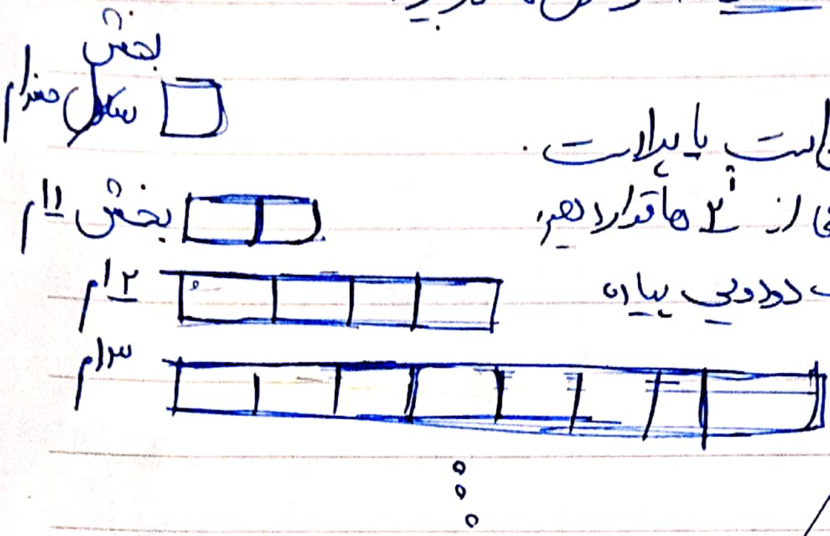


Subject:

Year Month Day

سوال ④ : در صورت سوال گفته شده که تعدادی بخش داریم و هر بخش 2^i تا سلول دارد
هر زندانی را باید به صورت sort شده در سلول ها قرار بدهند.

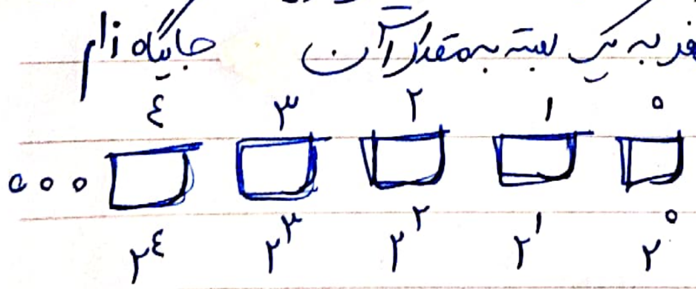
سلول ها به این شکل می باشند



هر بخش یا خالی است یا پر است.

پس ما باید زندانی ها را به صورت مجموعی از 2^i ها قرار دهیم
به نحوی باید مجموع n را به صورت دودویی پیاده سازی کنیم که عدد iام معادل با بخش iام می شود؛

پس با اضافه کردن هر زندانی
باید از بیت سمت راست شروع کرد
تا زمانی که به بیت 0 نرسیدیم همه را صفر کنیم و آخرین بیت سمت راست که صفر است را یک کنیم. به نحوی باید تمام آن زندانی ها را خارج کنیم و با آن یک زندانی اضافه شده همه را به آن بخشی که از صفر به یک تغییر کرده است ببریم. حال چون هر کدام از بخش ها sort شده اند پس باید برای منتقل کردن آن ها آن ها را merge کنیم.
و از بخش یعنی مقدار iام که این کار 2^i طول خواهد کشید و هزینه خواهد داشت.
پس یعنی برای تبدیل شدن هر بیت صفر به یک بسته به مقدار آن 2^i هزینه خواهد داشت.



- جایگاه ها دودویی
- میزان هزینه ها

حال می دانیم که در n بار اضافه کردن زندانی ها ما به اندازه‌ی n بار بیت صفر را به اندازه‌ی $\frac{n}{2}$ بار بیت 1ام و ... تغییر خواهیم داشت پس این گونه می توان نوشت؛

$$\overbrace{n \times 2^0 + \frac{n}{2} \times 2^1 + \frac{n}{4} \times 2^2 + \dots + \frac{n}{\log_2 n} \times 2^{\log_2 n}}^{\log_2 n \text{ بار}} = n(\overbrace{1+1+1+\dots+1}^{\log_2 n}) = n\log_2 n$$

حال اگر تقسیم بر n کنیم — $\frac{n\log_2 n}{n} = \log n$

که یعنی هزینه‌ی سرشکن شده برابر با $\log n$ است.

NoteBook Sary