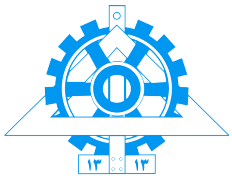
باسمه تعالی

دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر



ساختمان‌داده‌ها و الگوریتم‌ها

تمرین دوم - داده‌ساختارهای پایه

شایان کاشفی، صادق ابوفاضلی

تاریخ تحویل: ۰۲/۸/۱۹

۱. پشته خودشناس — ۲۰ نمره

الف) با استفاده از استک و با کمک حافظه اضافی $O(n)$ ، ورژن جدیدی از استک ارائه دهید که علاوه بر عملیات‌های درج و حذف، عملیات خروجی دادن عنصر کمینه را در مرتبه زمانی $O(1)$ انجام دهد. نحوه انجام عملیات درج، حذف و خروجی دادن عنصر کمینه را در این داده ساختار جدید توضیح دهید.

ب) سوال بالا را این بار با کمک حافظه اضافی $O(1)$ حل کنید. نحوه انجام عملیات درج، حذف و خروجی دادن عنصر کمینه را در این داده ساختار جدید توضیح دهید.

پاسخ :

الف) برای اینکار میتوان هنگام درج در استک ، به جای اینکه فقط عدد مورد نظر را اضافه کنیم ، زوج مرتب (عدد، مینیمم فعلی استک) را در پشته درج کرد. در این صورت در هر زمانی ، زوجی که سر استک است ، عنصر کمینه اعداد داخلی استک را نیز نشان میدهد. در نهایت عملیات ها به صورت زیر خواهد بود.

درج:

عدد مورد نظر را به همراه مینیمم عدد درحال درج و عدد کمینه فعلی استک را در استک درج میکنیم.

حذف:

زوج سر استک را حذف میکنم.

پیدا کردن کمینه:

عدد مربوط به عنصر مینیمم در زوج مرتب سر استک را خروجی میدهیم.

ب) یک متغیر به نام $MinValue$ در نظر می‌گیریم که در آن عنصر کمینه فعلی استک را نگه میداریم. چالش اصلی ما در این بخش، مدیریت کردن این متغیر در زمانی است که عنصر کمینه از داخل استک pop میشود و ما باید عنصر کمینه از میان عناصر باقی مانده‌ی استک را جایگزین آن کنیم. برای این کار، در زمانی که عنصر کمینه جدیدی مانند x به داخل استک $push$ میشود، بجای x ، مقدار $2x - MinValue$ را وارد استک میکنیم و عملیات های مربوط به استک را به صورت زیر پیاده سازی میکنیم:

درج:

اگر استک خالی باشد، عدد مورد نظر (x) را در سر استک قرار میدهیم و $MinValue$ را برابر با x قرار میدهیم.

اگر استک خالی نباشد، دو حالت رخ میدهد:

۱. عدد x بزرگتر یا مساوی $MinValue$ باشد، در این صورت مقدار $MinValue$ تغییری نخواهد کرد پس x را به آسانی در سر استک قرار میدهیم.

۲. عدد x کوچکتر از $MinValue$ باشد که در این صورت همانطور که گفته شد مقدار $2x - MinValue$ را در سر استک قرار داده و سپس مقدار $MinValue$ را آپدیت کرده و برابر با x قرار میدهیم.

حذف: (عنصر سر استک را y در نظر میگیریم)

دو حالت وجود دارد:

۱. عنصری که در سر استک وجود دارد بزرگتر یا مساوی $MinValue$ باشد، در این صورت مقدار کمینه یعنی $MinValue$ ثابت میماند و مقدار y به راحتی pop میشود.

۲. عنصری که در سر استک وجود دارد کمتر از $MinValue$ باشد، که نشان میدهد در هنگام درج بجای خود عنصر ،x مقدار $2x - MinValue_{previous}$ در استک درج شده است. (زیرا میدانیم در حال حاضر $MinValue$ کمترین مقدار است اما عنصری کوچکتر از آن در سر استک وجود دارد که طبیعتا حاصل $2x - MinValue_{previous}$ بوده است. از طرفی میدانیم که اینکار زمانی رخ میدهد که عنصری کوچکتر از کمینه ی موجود در آن زمان به استک وارد شده است، پس x در اصل $MinValue_{current}$ است و y که در سر استک وجود دارد برابر با $2x - MinValue_{previous}$ میباشد. در نتیجه مقدار $MinValue_{previous}$ را میتوانیم از بصورت $2MinValue_{current} - y$ بدست آوریم و جایگزین $MinValue$ فعلی کنیم.

همچنین مقدار x را که برابر با $MinValue_{current}$ است را بعنوان عنصر pop شده خروجی میدهیم.

پیدا کردن کمینه:

برای این عملیات تنها کافی است مقدار $MinValue$ را خروجی دهیم.

۲. درهم سازی ۱۵ نمره

یک آرایه به طول n در اختیار داریم و میخواهیم تعدادی استک را با استفاده از این آرایه پیاده‌سازی کنیم.

الف) راه حلی بهینه ارائه دهید که بتوان ۲ استک را در این آرایه پیاده سازی کرد. راه حل شما باید هم از لحاظ زمان و هم از لحاظ حافظه بهینه باشد. (هردو از مرتبه $O(1)$ باشند)

ب) راه حلی ارائه دهید که بتوان ۳ استک را در این آرایه پیاده سازی کرد به نحوی که از لحاظ حافظه بهینه باشد.

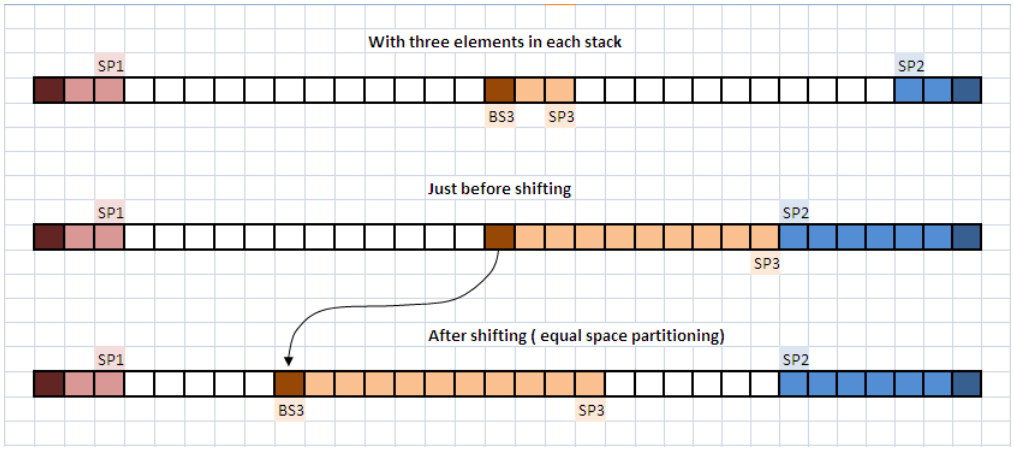
ج) راه حلی ارائه دهید که بتوان ۳ استک را در این آرایه پیاده سازی کرد به نحوی که از لحاظ زمانی بهینه باشد.

پاسخ :

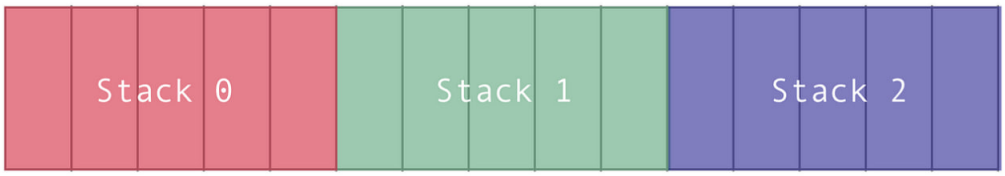
الف) برای اینکه از حافظه موجود به طور کامل استفاده کنیم. آرایه را به طور ثابت بین دو استک تقسیم نمیکنیم و اندازه شان را شناور در نظر میگیریم. به این صورت که شروع آرایه را به استک اول و پایان آرایه را به استک دوم اختصاص میدهیم به گونه ای که استک اول به سمت انتهای آرایه و استک دوم به سمت شروع آرایه رشد میکند. برای پیاده‌سازی این ساختار میتوان دو پوینتر ایجاد کرد که سر استک اول و دوم را نگه میدارند و عملیات های درج و حذف با جابجایی پوینترها و ذخیره اطلاعات در آرایه انجام میشود. با اینکار نیاز به دو پوینتر به عنوان حافظه اضافی داریم که از $O(1)$ میباشد و عملیات های درج و حذف هم فقط نیاز به جابجایی پوینتر دارد که عملیاتی از $O(1)$ است.

ب) برای اینکار میتوان همانند قسمت الف سه استک با سایزهای متغییر در نظر گرفت به این صورت استک اول از ابتدای آرایه به سمت انتهای آرایه و استک دوم از انتهای آرایه به سمت ابتدای آرایه و استک سوم از وسط آرایه به سمت انتهای آرایه رشد کند. حال برای استفاده بهینه از حافظه موجود ، هنگامی که استک اول یا دوم به استک سوم برخورد میکردند ، اعضای استک سوم را به گونه ای در آرایه جابجا میکنیم که حافظه موجود به طور مساوی بین استک ها تقسیم شود. اینکار باعث میشود در کل هزینه زمانی زیادی برای جابجایی استک سوم بپردازیم اما از طرفی از حافظه موجود به طرز بهینه استفاده کرده ایم.

Two Stacks using one single array



ج) میتوان با تقسیم کردن آرایه به سه بخش با اندازه ثابت و نگهداری داده‌های مربوط به استک در این بخش ها به علاوه پوینترهایی که سر استک ها را نشان میدهند ، سه استک را با پیچیدگی زمانی $O(1)$ برای تمام عملیات های مربوط به استک، پیاده سازی کرد.



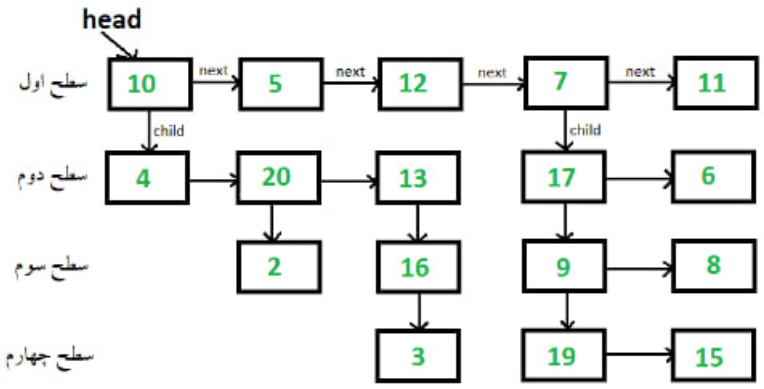
۳. عمق اول
۱۰ نمره

یک لیست پیوندی داریم که هر گره آن، علاوه بر اشاره‌گر به عنصر بعدی ،(next) یک اشاره‌گر فرزند (child) دارد که ممکن است به سر یک لیست جداگانه اشاره کند. هر کدام از لیست‌های فرزند نیز ممکن است یک یا چند لیست فرزند برای خود داشته باشند (مانند شکل زیر). سرِ لیست اول از سطح اول به شما داده شده است. لیست‌های پیوندی چند سطحی را به نحوی به یک لیست پیوندی تک‌سطح ساده (فقط شاملِ اشاره‌گر next) تبدیل کنید که در لیست پیوندی نهایی، تمام گره‌های سطح اول قبل از گره‌های سطح دوم، و تمام گره‌های مربوط به سطح دوم قبل از گره‌های سطح سوم بیایند و به همین ترتیب. همچنین در بین دو لیست فرزند که در یک سطح قرار دارند، اولویت با لیستی است که گره پدر آن به سر لیست نزدیکتر است.

مثال:

لیست نهایی:

۱۰ <- ۵ <- ۱۲ <- ۷ <- ۱۱ <- ۴ <- ۲۰ <- ۱۳ <- ۱۷ <- ۶ <- ۲ <- ۱۶ <- ۹ <- ۸ <- ۳ <- ۱۹ <- ۱۵



پاسخ :

از عنصر اول شروع کرده و تا انتهای لیست را پیمایش کرده و به همان ترتیب خروجی میدهیم. همچنین در حین پیمایش لیست، فرزند عنصرهایی که دارای فرزند هستند را به انتهای یک صف وارد میکنیم. با اینکار پس از پیموده شدن سطح اول ، عناصر اول تمام لیست های سطح دوم را در صف داریم. سپس تا زمانی که صف خالی نشده است، عنصر سر صف را برداشته و عملیات پیمایش را روی آن انجام داده و فرزندان جدید را نیز به انتهای صف اضافه میکنیم. با اینکار ترتیب عناصر در سطخ های مختلف حفظ شده و عنصری که سطح بالاتری داشته باشد، قبل از سطرهای بعدی پیمایش میشوند.

۴. در جستجوی خود
۱۰ نمره

یک لیست پیوندی با اندازه متناهی در اختیار داریم. ممکن است عضو آخر لیستمان به یکی از اعضای قبلی این لیست اشاره کند و باعث ایجاد دور شود. الگوریتمی از مرتبه زمانی چندجمله‌ای نسبت به اندازه لیست پیوندی ارائه دهید که با استفاده از حافظه $O(1)$ تشخیص دهد که آیا این لیست دور دارد یا نه؟

پاسخ :

برای یافتن دور ابتدا دو پوینتر تعریف میکنیم که هر دو به عضو اول لیست پیوندی اشاره میکنند. حال پوینتر اول را با قدم های به طول ۱ و پوینتر دوم رو با قدم هایی به طول ۲ جلو می بریم. اگر پوینتر دوم مجددا به پوینتر اول رسید یعنی اینکه پوینتر دوم از طریق دوری که در لیست پیوندی موجود است ، مجددا به پوینتر اول رسیده است. در غیر این صورت پوینتر دوم پس از طی کردن کل لیست پیوندی به انتهای لیست رسیده و لیست پیوندی ، بدون دور تشخیص داده میشود. توجه کنید که این الگوریتم در کل از مرتبه زمانی چند جمله ای نسبت به اندازه لیست پیوندی است زیرا پوینتر اول هر عضو این لیست پیوندی را حداکثر یکبار مشاهده میکند.

۵. سال بالایی
۲۰ نمره

در روزهایی که سلف دانشگاه جوجه‌کباب میدهد، صفی دراز به طول n در سلف ایجاد میشود. دانشجویان از ورودی های مختلف و با سن های متفاوت هستند و ما سن هرکس را میدانیم. از آنجایی که احترام به بزرگتر واجب است، میخواهیم کاری کنیم که هرکس در صف، سن اولین فرد عقب‌تر از خودش در صف که از او بزرگتر است را بداند تا مکان خود را در صف به اون تعارف کند. الگوریتمی ارائه دهید که برای هر یک از افراد داخل صف، این مقدار (سن اولین فرد عقب‌تر از خودش در صف) را خروجی دهد. (اگر فردی با چنین ویژگی وجود نداشت، ۱- خروجی دهد) پیچیدگی زمانی الگوریتم شما باید $O(n)$ باشد

مثال: لیست ورودی:

ته صف ۲۴, ۱۸, ۲۲, ۲۰, ۱۹ سر صف

لیست خروجی:

۲۰, ۲۲, ۲۴, ۲۴, −۱

پاسخ :

نکته ۱: اگر فرد y سال بالایی فرد x باشد، هیچکدام از افراد بین x و y نمیتوانند سال بالایی افراد سمت چپ x باشند.

با توجه به نکته بالا میتوانیم افراد بین یک فرد و سال بالایی اش را، از بین کاندیداهای سال‌بالایی بودن افراد سمت چپ این فرد حذف کنیم. حال برای حل این سوال از ته صف شروع کرده و به سمت سر صف حرکت میکنیم و سعی میکنیم سال‌بالایی هر فرد را پیدا کنیم. برای این کار یک پشته از افرادی که کاندید سال‌بالایی بودن هستند را در هر مرحله نگه‌داشته و بروز میکنیم. به این صورت که وقتی به دنبال سال‌بالایی یک فرد هستیم ، از سر پشته شروع به جستجو کرده و در صورتی که سن این فرد از سن فردی که سر پشته است کمتر بود ، سال‌بالایی این فرد را یافته ایم و خود این فرد را نیز به کاندیداهای افراد بعدی اضافه میکنیم(با اضافه کردن سن این فرد به سر پشته) و در صورتی که سن این فرد از کسی که سر پشته است بیشتر بود ، فرد سر پشته را از پشته حذف کرده و در باقی پشته بدنبال سال‌بالایی میگردیم (اگر پشته خالی بود جواب −۱ است و فقط فرد فعلی را به پشته اضافه میکنیم). از آنجایی که هر وقت به پشته عددی اضافه میکنیم از عدد سر پشته کوچکتر است ، همیشه در پشته، افراد کاندید به صورت صعودی وجود دارند و میتوان به دنبال اولین فرد بزرگتر از فرد فعلی گشت و کسانی که دیگر نمیتوانند کاندید باشند را از پشته حذف کرد. از آنجایی که هر فرد دقیقا یکبار به پشته اضافه شده و دقیقا یکبار از پشته حذف میشود و در کل n نفر داریم، پس این الگوریتم در نهایت از $O(n)$ خواهد بود.

۶. سیم پیچ

۲۵ نمره

یک دستگاه توسط یک سیم + و یک سیم - به یک دوشاخه متصل شده است. اما متاسفانه سیم ها در هم پیچیده شده اند. سیم ها در امتداد کف از دیوار (در سمت چپ) به دستگاه (در سمت راست) کشیده می شوند. دیوار و دستگاه هر دو دارای دو ورودی در یک سطح هستند که سیم ها به ترتیب به آنها وصل می شوند. اگر یک یا چند مکان وجود داشته باشد که یک سیم از روی یک سیم دیگر عبور باشد، سیم‌ها در هم پیچیده می‌شوند. به عنوان مثال، تصویر زیر دارای چهار مکان از این قبیل است (نمای بالا):

ما یک رشته داریم که نشان میدهد به ترتیب کدام سیم از روی دیگری عبور کرده است(برای مثال بالا -++-). همچنین میدانیم که در سمت چپ، سیم + همیشه به ورودی بالایی وصل می شود (همانطور که در تصویر مشاهده می شود). ما دوست داریم که سیم‌ها را بدون جدا کردن و بدون حرکت دادن دستگاه باز کنیم به صورتی که دیگر تقاطع نداشته باشند. الگوریتمی از $O(n)$ ارائه کنید که تعیین کند آیا امکان انجام آن وجود دارد یا خیر. یک سیم را می توان آزادانه حرکت داد و روی زمین کشید، اما نمیتوان آن را برش داد. برای درک بهتر مشکل لطفا نکات مربوط به نمونه های آزمایشی را مطالعه کنید.

$++ : YES$

$+- : NO$

$- : NO$

پاسخ :

اگر در قسمتی از این سیم پیچ ، دو علامت هم جنس داشته باشیم ، میتوانیم تقاطع های آن بخش را باز کرده و تعداد تقاطع ها را کاهش دهیم. برای مثال میتوان از $+-++--$ به $+--+--$ رسید و همینطور به این کار ادامه داد. اگر در نهایت همه تقاطع ها رفع شده باشد به هدف رسیده ایم و اگرنه به سیم پیچی رسیده ایم که در آن سیم های مثبت و منفی بطور یکی در میان از روی یکدیگر عبور کرده و امکان باز کردن گره ها وجود ندارد. برای پیاده‌سازی این الگوریتم میتوانیم از یک پشته استفاده کنیم. به این صورت که از سمت چپ رشته شروع به پیمایش کرده و به ازای هر کاراکتر عملیات زیر را انجام میدهیم.

اگر کارکتر فعلی با کاراکتری که در بالای پشته است یکسان باشد ، این دو با هم ساده شده و کاراکتر بالای پشته زا خارج میکنیم در غیر این صورت کاراکتر فعلی به پشته اضافه میشود. در نهایت در صورت خالی بودن پشته جواب مثبت است و در غیر این صورت جواب منفی است. از آنجایی که عملیات های درج و حذف از پشته از $O(1)$ است و در کل n کاراکتر داریم، پس در نهایت پیچیدگی زمانی این الگوریتم از $O(n)$ میباشد.