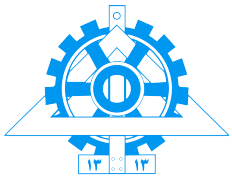

باسمه تعالی

دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر

ساختمان‌داده‌ها و الگوریتم‌ها

پاسخ تمرین دوم - داده‌ساختارهای پایه

علی ممتحن، محمد امانلو

تاریخ تحویل: ۱۴۰۳/۰۱/۲۶

۱. اعجاب لینکد لیست ها — ۱۰ نمره

یک لینکد لیست یک طرفه برای اعداد ۴ بایتی در اختیار داریم. این لینکد لیست دارای یک بخش val برای ذخیره کردن عدد ۴ بایتی و یک بخش next شامل اشاره‌گری به نود دیگر می‌باشد. فرض کنید حافظه‌ای که اشاره‌گر next دارد برابر ۸ بایت است. حال با استفاده از ساختمان این لینکد لیست، یک لینکد لیست دوطرفه برای اعداد ۴ بایتی طراحی کنید.

پاسخ :

می‌توانیم با استفاده از دو متغیر val یک اشاره‌گر را ذخیره کنیم. ۳ نود از این نوع لینکد لیست را بعنوان یک نود لینکد لیست دوطرفه در نظر می‌گیریم. به این صورت که این ۳ نود هر کدام به یکدیگر اشاره می‌کنند (نود اول به نود دوم و نود دوم به نود سوم اشاره می‌کند) و سومین نود، به نود اول سازنده‌ی نود بعدی لینکد لیست دوطرفه اشاره می‌کند. همچنین ۳ حافظه val داریم که با استفاده از ۲ تای آن‌ها می‌توانیم اشاره‌گری به نود اول سازنده عنصر قبلی لینکد لیست دوطرفه را ذخیره کنیم. از حافظه val سوم هم برای ذخیره مقدار مربوط به عدد ۴ بایتی لینکد لیست دوطرفه استفاده می‌کنیم.

اگر حافظه مورد نیاز برای اشاره‌گر لینکد لیست برابر k بایت باشد هم می‌توان از همین ایده، منتها با استفاده از تعداد نود بیشتر بهره برد.

۲. تقسیم عادلانه — ۱۵ نمره

تعدادی کودک به شدت گرسنه در یک ردیف ایستاده‌اند که مقدار گرسنگی این کودکان در یک آرایه داده شده است. می‌خواهیم کمترین تعداد شکلات ممکن را بین این کودکان به گونه‌ای تقسیم کنیم که هر کودک حداقل یک شکلات بدست بیاورد و اگر کودکی از کودک کناری خود گرسنه‌تر باشد، از او بیشتر شکلات بگیرد. الگوریتمی در مرتبه $O(n)$ ارائه دهید تا این کار را برای ما انجام دهد.

پاسخ :

به کودک اول یک شکلات داده و سپس از کودک دوم شروع کرده و هر بار گرسنگی کودک فعلی را با قبلی مقایسه می‌کنیم. اگر گرسنه‌تر بود، به اندازه یکی بیشتر از کودک قبلی به آن شکلات می‌دهیم. اگر برابر یا کمتر بود، یک شکلات به کودک جدید می‌دهیم و این کار را ادامه می‌دهیم تا به آخرین کودک برسیم. حالا از اخر به اول حرکت می‌کنیم و به هر کودکی که می‌رسیم، آن را با کودک جلوتر از خودش مقایسه می‌کنیم و مانند مرحله قبل اگر گرسنه‌تر بود، به اندازه یکی بیشتر از کودک جلویی به او شکلات می‌دهیم و در غیر این صورت از او عبور کرده و به سراغ کودک بعد می‌رویم. با رسیدن به کودک اول، الگوریتم پایان می‌پذیرد و شکلات‌ها مطابق خواست مسئله تقسیم شده‌اند.

۳. بزرگترین عدد ۱۵ نمره

فرض کنید دنباله‌ای از اعداد ۱ تا ۹ به طول N داریم. می‌خواهیم زیردنباله‌ای به طول K انتخاب کنیم به طوری که با کنار هم قرار دادن ارقام داخل این زیردنباله به همان ترتیبی که در دنباله اصلی قرار دارند، بزرگترین عدد ممکن را بسازیم. الگوریتمی از مرتبه O(n) ارائه دهید که این عدد را بیابد.

$$\{۲, ۴, ۵, ۲, ۴, ۹, ۱\} \quad K = ۳ \rightarrow ۵۹۱$$

پاسخ :

پشته‌ای به سایز K در نظر می‌گیریم. روی دنباله پیمایش می‌کنیم و هر بار عنصر فعلی دنباله را با سر استک مقایسه می‌کنیم. اگر استک خالی باشد، عنصر را در استک پوش می‌کنیم و در غیر این صورت، اگر از سر استک کوچکتر یا مساوی بود، در صورتی که استک فضای خالی داشته باشد آن را داخل استک پوش میکنیم و سراغ عنصر بعدی در دنباله می‌رویم؛ اما اگر از سر استک بزرگتر بود، در صورتی که تعداد عناصر باقی‌مانده در دنباله از مقدار حافظه خالی استک بیشتر یا مساوی باشد، سر استک را پاپ می‌کنیم. سپس دوباره مقایسه گفته شده را بین سر استک جدید و عنصر فعلی دنباله انجام می‌دهیم. در نهایت، عناصر موجود در استک جواب مسئله خواهند بود.

۴. بیشترین‌ها ۲۰ نمره

یک آرایه به طول N داریم و می‌خواهیم بیشترین عدد تمام بازه های به طول k از این آرایه را بیابیم. الگوریتمی در پیچیدگی $O(N)$ برای این کار ارائه دهید.

$$array = \{۱, ۲, ۴, ۳, ۶, ۵, ۷\} \quad k = ۳$$

$$max\{۱, ۲, ۴\} = ۴, \quad max\{۲, ۴, ۳\} = ۴, \quad max\{۴, ۳, ۶\} = ۶, \quad max\{۳, ۶, ۵\} = ۶, \quad max\{۶, ۵, ۷\} = ۷$$

پاسخ :

از یک صف دوطرفه به طول k استفاده می‌کنیم.

ابتدا تلاش می‌کنیم تا k عنصر ابتدایی آرایه را وارد صف کنیم؛ اما به این صورت که قبل از اضافه کردن هر عنصر به صف، آن عنصر را با عنصر موجود در ته صف مقایسه کرده و اگر عنصر ته صف از عنصر فعلی کوچکتر بود، عنصر ته صف را خارج کرده و آنقدر این کار را انجام می‌دهیم تا عناصر باقی‌مانده در صف بزرگتر از عنصر فعلی باشند. سپس عنصر فعلی را به ته صف وارد می‌کنیم.

سپس در یک حلقه از k تا انتهای آرایه، مراحل زیر را انجام می‌دهیم:

۱. عنصر سر صف را به عنوان بزرگترین عدد بازه قبلی چاپ می‌کنیم.

۲. در صورتی که عنصر سر صف، خارج از بازه k تایی فعلی باشد، آنرا از صف حذف می‌کنیم.

۳. عنصر بعدی آرایه را به ته صف وارد می‌کنیم؛ اما به این صورت که قبل از اضافه کردن عنصر به صف، آن عنصر را با عنصر موجود در ته صف مقایسه کرده و اگر عنصر ته صف از عنصر فعلی کوچکتر بود، عنصر ته صف را خارج کرده و آنقدر این کار را انجام می‌دهیم تا عناصر باقی‌مانده در صف بزرگتر از عنصر فعلی باشند. سپس عنصر فعلی را به ته صف وارد می‌کنیم.

در انتها پس از خارج شدن از حلقه، عنصر سر صف (که بزرگترین عدد بازه آخر است) را چاپ می‌کنیم.

۵. لولک و بولک ۲۰ نمره

در پیک نوروزی از لولک و بولک خواسته شده است تا باینری تمام اعداد ۱ تا n را با استفاده از صف بنویسند. از آن جا که این دو عزیز به نظم اهمیت بسیاری می‌دهند تصمیم گرفته اند این کار را به ترتیب انجام دهند؛ یعنی از ۱ شروع کرده و در آخر n را بنویسند. اما مثل جیگر در گل گیر کرده و حالا از شما میخواهند به آنان کمک کنید. برای آن ها شبه کدی بنویسید که کار بالا را در O(n) انجام دهد.

پاسخ:

فرض کنید در حال حاضر رشته باینری مربوط به عدد i ام در سر صف باشد، با استفاده از آن رشته باینری عدد $2i$ و رشته باینری عدد $2i+1$ را در صف پوش می‌کنیم. از یک شروع کرده و به ازای هر عدد که در سر صف قرار می‌گیرد این کار را انجام می‌دهیم.

```
void printBinary1toN(int n)
{
    // create an empty queue and enqueue 1
    queue<string> q;
    q.push("1");

    // run `n` times
    int i = 1;
    while (i++ <= n)
    {
        // append 0 and 1 to the front element of the queue and
        // enqueue both strings
        q.push(q.front() + "0");
        q.push(q.front() + "1");

        // dequeue front element after printing it
        cout << q.front() << ' ';
        q.pop();
    }
}
```

۶. خزعبلات ۲۰ نمره

به یک پرانتزگذاری، پرانتزگذاری ناخزعبل می گوییم اگر آن پرانتزگذاری درست باشد و در آن هیچ جفت پرانتزی که با حذف آن‌ها تغییری در محاسبات کل عبارت ایجاد نشود، وجود نداشته باشد. برای مثال (x)(x) یا ((x)(x))x پرانتز گذاری ناخزعبل هستند. اما برای مثال (x) به دلیل اینکه با حذف پرانتز تغییری در نحوه عملیات ایجاد نمی‌شود، ((x)(x)) به دلیل اینکه پرانتز اول بسته نشده است، ()(x) به دلیل اینکه داخل جفت پرانتز دوم هیچ عبارتی وجود ندارد و (((x)(x))) چون در آن دومین پرانتزی که باز می‌شود بی‌فایده است و حذف آن تاثیری ندارد، خزعبل هستند. دقت کنید که x میتواند هر عبارت جبری مانند ۲+۳*۴+۵ باشد.

الگوریتمی در مرتبه O(n) ارائه دهید که مشخص کند یک پرانتزگذاری خزعبل هست یا ناخزعبل.

پاسخ:

برای اینکه یک پرانتز گذاری ناخزعبل باشد باید دو شرط داشته باشد.

اولا هر پرانتزی که باز می‌شود و بسته می‌شود باید داخش حداقل یک عبارت (مانند x) وجود داشته باشد.

دوما باید قبل یا بعد تکه‌ای که داخل پرانتز قرار گرفته است، یعنی خارج قطعه پرانتزگذاری شده و داخل پرانتز بیرونی، یک عبارت موجود باشد. وگرنه این جفت پرانتز که باز کرده و سپس بسته‌ایم، بیهوده بوده است. برای مثال در (x) قبل و بعد پرانتز هیچ عبارت دیگری نیست پس این پرانتز بیهوده است.

برای حل این مسئله یک استک در نظر می‌گیریم. حال هر کدام از اعداد زیر اگر در سر استک باشند، معانی زیر را دارند:

عدد ۰: یعنی آخرین پرانتزی که باز مانده است هیچ کدام از ۲ شرط بالا را ندارد.

عدد ۱: یعنی آخرین پرانتزی که باز مانده است شرط ۱ را دارد.

عدد ۲ :یعنی آخرین پرانتزی که باز مانده است شرط ۲ را دارد.

عدد ۳: یعنی آخرین پرانتز باز مانده شرط ۱ و ۲ را دارد.

در پرانتزگذاری از سمت چپ به راست شروع به پیمایش می‌کنیم. در صورتی که به پرانتز باز رسیدیم اگر قبل از آن یک پرانتز بسته یا x بوده باشد، یعنی شرط دوم را دارا است و عدد ۲ را در استک پوش می‌کنیم. در غیر این صورت عدد ۰ را در استک پوش می‌کنیم. وقتی به x برسیم یعنی پرانتزهایی که داخل آن‌ها قرار داریم شرط اول را خواهند داشت. برای اینکه این را مشخص کنیم، به عددی که تاپ استک قرار دارد نگاه می‌کنیم و آنرا در صورتی که برابر ۰ بود به ۱ و در صورتی که برابر ۲ بود به ۳ تغییر می‌دهیم. (۳ به معنای داشتن هر دو شرط خواهد بود). هرگاه به کاراکتر پرانتز بسته رسیدیم، در صورتی که تاپ استک برابر ۱ یا ۳ نبود یعنی شرط اول را نداشته‌ایم و کلا پرانتزگذاری خزعبل خواهد بود. در غیر این صورت اگر برابر ۱ بود، یعنی هنوز شرط دوم را نداریم و با نگاه کردن به کاراکتر بعدی اگر "x" یا ")" بود یعنی این شرط را خواهیم داشت. حال عدد را از سر استک پاپ می‌کنیم و شرط اول را روی تاپ استک مجددا اعمال می‌کنیم (یعنی اگر ۰ بود به ۱ و اگر ۲ بود آن را به ۳ تبدیل می‌کنیم). در آخر اگر استک خالی باشد یعنی پرانتزگذاری ناخزعبل بوده و اگر در طول الگوریتم خواستیم از استک پاپ کنیم و استک خالی بود، این پرانتزگذاری خزعبل بوده است.