ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۷ـ۹۸

دانشکده‌ی مهندسی کامپیوتر

گردآورندگان: آراد محمدی ، کیارش بنی هاشم

پاسخ‌نامه تمرین عملی دوم

مسئله‌ی ۱. دیوار

باقی های مانده های ارتفاع قسمت های مختلف دیوار را بر ۲ در نظر می گیریم و درون یک stack پوش می کنیم. هر گاه عضوی که درون stack پوش کردیم با عضو روی stack برابر بود هر دو عنصر را پاپ می کنیم(در واقع میتوان با تعدادی بلوک به صورت عمودی این دو قسمت را هم ارتفاع کرد) در انتها اگر تعداد عناصری که در stack باقی مانده بود کمتر مساوی ۱ بود جواب مسأله برابر با YES است و در غیر این صورت برابر با NO (چون می توان با قرار دادن تعدادی بلوک به صورت افقی این قسمت هایی که در مرحله قبل با هم دیگر هم ارتفاع شده اند را با یکدیگر هم ارتفاع کرد) اثبات درستی این الگوریتم هم ب استقرا به سادگی انجام می شود.

مسئله‌ی ۲. نابجایی تابعی

ابتدا دو آرایه B و C را به این صورت می سازیم . از اول آرایه شروع می کنیم و B_i را برابر تعداد a_i هایی که تا این لحظه آمده اند قرار می دهیم(این کار را به کمک یک آرایه کمکی که در هر لحظه خانه ی i ام آن تعداد بار هایی که a_i آمده را نشان می دهد به راحتی می توان به دست آورد) بعد از اتمام این کار a_i برابر $f(۱, i, a_i)$ می شود. در مرحله بعدی همین کار را از انتهای دنباله انجام می دهیم یعنی C_j را برابر تعداد a_j هایی که تا این لحظه آمده اند قرار می دهیم (با شروع از عنصر آخر دنباله) بعد از این مرحله a_j برابر $f(j, n, a_j)$ می شود. حالا مسأله به این تبدیل می شود که تعداد اندیس های i و j را بیاید که i از j کمتر است ولی B_i بیشتر از C_j است . برای حل این سوال مانند الگوریتم مرج سورت هر دو آرایه را به دو تکه تقسیم می کنیم . و بعد جواب را در تیکه راست و چپ به صورت بازگشتی حساب می کنیم و جوری تابع را پیاده سازی می کنیم که دو تکه در انتها به صورت مرتب شده باشند . در مرحله بعد تعداد اندیس ها را در قسمت اول آرایه B و قسمت دوم آرایه C حساب می کنیم و در انتها مانند الگوریتم مرج دو تکه آرایه ها را به صورت مرتب شده ادغام می کنیم .

مسئله‌ی ۳. یه مشت بازه

ابتدا دقت کنید که تابع پیچیده مساله همان XOR است.
یک بار لیست را پیمایش می کنیم و هنگام بررسی عنصر i ام سعی می کنیم تمامی بازه هایی که به عنصر i ام ختم می شوند را بررسی کنیم. اما از آن جایی که برای هر بازه تنها دو عنصر کمینه

مهم هستند، تنها بازه هایی که این عنصر در آن ها عنصر کمینه یا کمینه دوم است مهم هستند.
یک گزینه این است که برای محاسبه مقادیر فوق در هر مرحله اندیس j را برابر i قرار دهیم و آن قدر کم کنیم تا عنصر i جزو دو عنصر کمینه $[i,j]$ نباشد. اما این راه حل زمان زیادی می برد.
دقت کنید که که اگر عنصر $a[j]$ از عنصر $a[j-1]$ کمتر باشد آنگاه اگر بازه $[j,i]$ را بررسی کنیم، نیازی نیست که بازه $[j-1,i]$ را نیز بررسی کنیم زیرا اگر عنصر i ام جزو دو عنصر کمینه نباشد که اصلا این بازه مهم نیست و اگر هم باشد، عنصر $a[j-1]$ جزو دو عنصر کمینه نیست.
با توجه به شهود فوق، یک پشته از عناصر در نظر می گیریم که در آن تنها عناصری نگه داشته می شوند که بعد از آن ها عنصر کوچکتری نیامده است. این عناصر به ترتیب مشاهده شدن به پشته اضافه می شوند و در نتیجه دنباله ای صعودی تشکیل می دهند (عنصر روی پشته بیشنه است).
در هر مرحله با مشاهده عنصر جدید مانند x تا زمانی که عنصر روی پشته از آن بزرگتر است، عنصر روی پشته را حذف (pop) کرده و تابع را برای زوج (عنصری که حذف کردیم، x) محاسبه می کنیم و اگر از بیشینه مقادیری که قبلا محاسبه شده اند بیشتر بود، بیشینه را به روز می کنیم.
پس از این که عنصر رویی از عنصر ما کوچکتر شد، یک بار دیگر این تابع را برای زوج (عنصر رویی، x) محاسبه می کنیم تا حالت هایی که در آن ها x عنصر دوم کمینه است را نیز بررسی کرده باشیم. بدیهی است که در مرحله اگر پشته خالی باشد، کاری نمی کنیم.
پس از انجام محاسبات فوق x را در پشته درج می کنیم و برای عناصر بعدی این عمل را تکرار می کنیم.