به نام خدا

دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر
تحلیل و طراحی الگوریتم‌ها

راه‌حل تمرین کتبی سوم
موعد تحویل: یکشنبه ۴ اردیبهشت ۱۴۰۱، ساعت ۲۳:۵۹
طراح: محمدطاها فخاریان، taha.fakharian@gmail.com

در این تمرین، فقط برای سوالاتی که اثبات خواسته شده است، لازم است که راه‌حل خود را به صورت ریاضی و دقیق اثبات کنید.

۱. انتخاب حریصانه ما به این صورت خواهد بود: قوی‌ترین و ضعیف‌ترین گاو نر را در نظر بگیرید. اگر این گروه شرط مدنظر را برآورده کنند، آنها را در یک تیم قرار می‌دهیم. در غیر این صورت، ضعیف‌ترین گاو را حذف می‌کنیم. این الگوریتم را به صورت بازگشتی روی گاوهای باقی‌مانده اجرا می‌کنیم تا تمامی گروه‌ها تشکیل شوند. برای پیدا کردن گروه‌ها برحسب قدرت گاوها، آنها را برحسب قدرتشان مرتب می‌کنیم که در $O(nlogn)$ قابل انجام است. برای تشکیل دادن گروه‌ها نیز کافی است روی مجموعه مرتب شده، از اول و آخر آن شروع کنیم و این پوینترها را آپدیت کنیم که در $O(n)$ قابل انجام است. حال ثابت می‌کنیم که این روش، درست است:

لم ۱: فرض کنید s قوی‌ترین گاو و w ضعیف‌ترین گاو باشد. در صورتی که $s+w < P$ باشد، نمی‌توان w را در هیچ گروهی قرار داد.

اثبات: فرض کنید s' گاوی باشد که با w در یک گروه قرار می‌گیرد. در این صورت داریم: $s' + w \leq s + w < p$

در نتیجه این گروه نمی‌تواند گاوآهن را بکشد. از تناقض حاصل شده، درستی لم ثابت می‌شود.

لم ۲: فرض کنید s قوی‌ترین گاو و w ضعیف‌ترین گاو باشد. در صورتی که $s + w \geq P$ باشد. فرض کنید $Teams_2$ مجموعه‌ای از تیم‌هایی باشد که می‌توانند گاوآهن را بکشند و s و w باهم در یک تیم نیستند. در این صورت مجموعه‌ای از تیم‌ها مانند $Teams_1$ وجود دارد که می‌توانند گاوآهن را بکشند و s و w باهم در یک تیم هستند و داریم: $|Teams_2| \leq |Teams_1|$.

اثبات: اگر در $Teams_2$، هردوی s و w در تیم‌هایی حضور نداشته باشند، $Teams_1$ را همان $Teams_2$ در نظر می‌گیریم، با این تفاوت که تیم‌هایی که s یا w عضو آن هستند را حذف می‌کنیم و تیم (s, w) را نیز اضافه می‌کنیم. چون حداکثر یک تیم حذف شده و یک تیم هم اضافه شده، خواهیم داشت: $|Teams_2| \leq |Teams_1|$.

در غیر این صورت، فرض کنید در $Teams_2$ s با x و w با y در یک گروه قرار گرفته باشند. در این صورت داریم: $Teams_1 = Teams_2 - \{(s, x), (w, y)\} \cup \{(s, w), (x, y)\}$ در این مجموعه، s و w در یک گروه قرار گرفته‌اند و با توجه به اینکه قدرت x حداقل به اندازه w است، اندازه دو مجموعه باهم برابر بوده و در مجموعه جدید نیز تمامی گروه‌ها می‌توانند گاوآهن را بکشند. در هردو حالت لم اثبات شد.

حال با استفاده از این دو لم، درستی روش را اثبات می‌کنیم:

قضیه: هیچ مجموعه‌ای از تیم‌های مجاز مثل $Teams_2$ وجود ندارد که اندازه آن بزرگتر از مجموعه تیم‌های تولید شده توسط این روش باشد.

این قضیه را با استقرای قوی روی تعداد گاوهای نر(n) اثبات می‌کنیم. فرض کنید به ازای این الگوریتم به ازای تعداد گاوهای کمتر از n صحیح است. برای تعداد گاوهای نر برابر با n، فرض کنید مجموعه‌ای از تیم‌ها وجود دارد که اندازه آن بزرگتر از مجموعه تیم‌های تولید شده توسط این روش است. فرض کنید s قوی‌ترین و w ضعیف‌ترین گاو باشد. در صورتی که $s + w < P$، با توجه به لم اول، در هر دو مجموعه هیچ تیمی وجود ندارد که w یکی از آنها باشد. طبق استقرا، پاسخ الگوریتم حریصانه برای مجموعه گاوها - w درست است و لذا برای مجموعه گاوها هم درست خواهد بود.

حال در صورتی که $s + w \geq P$، طبق لم ۲ یک مجموعه از تیم‌ها مثل $Teams_1$ وجود دارد که s و w در یک تیم بوده و اندازه آن بزرگ‌تر مساوی $Teams_2$ است. در صورتی که مجموعه حاصل از الگوریتم را $GreedyTeams$ و مجموعه گاوها را $Oxens$ در نظر بگیریم، چون $GreedyTeams - \{(s, w)\}$ طبق استقرا یک پاسخ بهینه برای مجموعه $Oxens - \{s, w\}$ است. همچنین $Teams_1 - \{(s, w)\}$ نیز یک پاسخ قابل قبول برای مجموعه $Oxens - \{s, w\}$ است و داریم:

$|Teams_2 - \{(s,w)\}| \leq |Teams_1 - \{(s,w)\}| \leq |GreedyTeams - \{(s,w)\}| \rightarrow |Teams_2| - 1 \leq |Teams_1| - 1 \leq |GreedyTeams| - 1$
لذا پاسخ الگوریتم در این حالت نیز بهینه خواهد بود و لذا طبق استقرا، پاسخ الگوریتم به ازای هر N بهینه خواهد بود.

۲. انتخاب حریصانه به این صورت خواهد بود: به کودکی با بیشترین درجه اشتها توجه می‌کنیم: در صورتی که بزرگترین شیرینی او را راضی کرد، آن شیرینی را به او می‌دهیم و در غیر این صورت، کوچکترین شیرینی را به او می‌دهیم و این کار را به صورت بازگشتی روی باقی کودکان و شیرینی‌ها اجرا می‌کنیم. حال ثابت می‌کنیم که این روش درست است.

لم: فرض کنید g کودکی با بالاترین درجه اشتها بین کودکان بوده و s هم بزرگترین شیرینی است که g را راضی می‌کند. در این صورت یک پاسخ درست وجود دارد که در آن، s به g داده می‌شود.

اثبات: فرض کنید $Assign$ یک پاسخ درست باشد. در صورتی که در این پاسخ، s به g داده شده باشد که کار تمام است. در غیر این صورت، یا s به هیچ کودکی داده نشده یا به کودکی غیر از g داده شده است. در حالت اول، s را به g می‌دهیم و به شیرینی باقی کودکان دست نمی‌زنیم. با توجه به اینکه باقی کودکان دست نخورده هستند و g هم راضی خواهد بود، این پاسخ نیز درست خواهد بود. در حالت دوم، فرض کنید که به g شیرینی j و شیرینی s به کودک t داده شده باشد. در این حالت، شیرینی s را به g و شیرینی j را به t می‌دهیم و به شیرینی باقی کودکان دست نمی‌زنیم. با توجه به اینکه g بالاترین درجه اشتها را دارد، در صورتی که g با j راضی بوده باشد، حتما t هم با j راضی خواهد بود و همچنین می‌دانیم که g با s راضی است. لذا در صورتی که در پاسخ قبلی، هردو کودک راضی بوده باشند، در این پاسخ نیز راضی خواهند بود و در غیر این صورت، در پاسخ جدید حداقل g راضی خواهد بود که نشان می‌دهد در هردو حالت، در پاسخ جدید تعداد کودکان راضی حداقل به اندازه تعداد کودکان راضی در $Assign$ است که باعث می‌شود لم بالا اثبات شود.

لم: فرض کنید $g_1, \cdots, g_n$ درجه اشتهای کودکان از زیاد به کم باشد و $s_1, \cdots, s_m$ سایز شیرینی‌ها از بزرگ به کوچک باشد. در صورتی که $g_1 > s_1$ باشد. فرض کنید A' یک پاسخ باشد که در آن، شیرینی m به کودک 1 داده نشده است. یک پاسخ A وجود خواهد داشت که در آن، شیرینی m به کودک 1 داده شده است و تعداد کودکان راضی در A حداقل به تعداد کودکان راضی در A' خواهد بود.

اثبات: فرض کنید در A' به کودک 1 شیرینی $j < m$ داده شده باشد. برای حالت اول فرض کنید که شیرینی m به هیچ کودکی داده نشده باشد. در پاسخ A این شیرینی را به کودک 1 داده و به شیرینی باقی کودکان دست نمی‌زنیم. با توجه به شیرینی باقی کودکان تغییر نکرده و طبق نامساوی‌های گفته شده، در هردوحالت کودک 1 نمی‌تواند راضی شود، تعداد کودکان راضی در هردو حالت باهم برابر خواهد بود. در غیر این صورت، فرض کنید شیرینی m به کودک $i > 1$ داده شده باشد. در پاسخ جدید، شیرینی کودکان به جز i و 1 با A' برابر خواهد بود و شیرینی‌های دو کودک i و 1 را باهم جابه‌جا می‌کنیم. مشابه قبل، کودک 1 در هردو پاسخ راضی نخواهد بود و در صورتی که i در A' راضی بوده باشد، در پاسخ جدید شیرینی با اندازه‌ای بزرگتر از پاسخ قبل گرفته و باز هم راضی خواهد بود. لذا در هر دو حالت، لم اثبات شد.

طبق این دو لم، در هر دو حالت، پاسخی بهینه وجود خواهد داشت که به کودک با بالاترین درجه اشتها، همان شیرینی‌ای را می‌دهد که الگوریتم حریصانه می‌دهد. حال با استقرا روی تعداد کودکان درستی الگوریتم را اثبات می‌کنیم. در صورتی که هیچ کودکی نباشد که هر پاسخی درست است. فرض کنید که الگوریتم برای تعداد کودکان $n - 1$ صحیح باشد. فرض کنید برای n نفر، $Assign$ همان پاسخی باشد که در بالا ذکر شده است و به کودک با بالاترین درجه اشتها، شیرینی CG را می‌دهد. در این پاسخ، به باقی کودکان از بین باقی شیرینی‌ها، شیرینی داده می‌شود و طبق استقرا، الگوریتم حریصانه برای این کودکان درست خواهد بود. در نتیجه تعداد کودکان راضی در این پاسخ حداکثر برابر با تعداد کودکان راضی در پاسخ الگوریتم ما خواهد بود و در هر دو پاسخ، کودک با بالاترین درجه اشتها یا راضی خواهد بود یا ناراضی که باعث می‌شود تعداد کل کودکان راضی در $Assign$ حداکثر برابر با تعداد کل کودکان راضی در پاسخ حریصانه باشد. در نتیجه طبق استقرا این الگوریتم به ازای تمامی تعداد کودکان صحیح خواهد بود. یک پیاده‌سازی به این صورت خواهد بود: ۱. کودکان را برحسب اشتها و شیرینی‌ها را برحسب اندازه‌هایشان از بزرگ به کوچک مرتب کن.

۲. قرار بده: $I = 1, J = m$.

۳. برای $K = 1$ تا N انجام بده:

۴. اگر $s_I \geq g_k$، در آن صورت قرار بده $Assign[K] = I, I++$.

۵. در غیر این صورت قرار بده $Assign[K] = J, J++$.

۶. برگردان $Assign$.

هزینه کل الگوریتم طبق این پیاده برابر با $O(mlogm)$ خواهد بود.

۳. (آ) به صورت حریصانه هنگامی که به پمپ بنزین i رسیدیم، اگر به اندازه رسیدن به پمپ بنزین i+۱ بنزین داشته باشیم، در این پمپ بنزین نمی‌ایستیم و به حرکت ادامه می‌دهیم. در غیر این صورت، به ناچار در این پمپ بنزین می‌ایستیم و باک را پر می‌کنیم و به

مسیر ادامه می‌دهیم. برای اثبات بهینه بودن، فرض کنید پاسخ الگوریتم S باشد و این پاسخ بهینه باشد. نزدیک‌ترین پاسخ بهینه به S را در نظر می‌گیریم و آن را S' در نظر می‌گیریم(نزدیکی دو پاسخ برابر است با بزرگترین ای‌i که هردو پاسخ تا پمپ بنزین امi در ایستادن یا حرکت کردن مثل هم عمل کرده‌اند). فرض کنید S و S' تا پمپ بنزین k-۱ مثل هم عمل کرده باشند و در پمپ بنزین امk متفاوت عمل کرده باشند. در این صورت طبق تعریف در S حرکت کرده‌ایم و در S' ایستاده‌ایم. در این صورت پاسخی مثل S'' را در نظر بگیرید که در پمپ بنزین k نمی‌ایستاد و به جای آن در پمپ بنزین k+۱ می‌ایستاد. شمار توقف‌های این دو پاسخ مثل هم است ولی S'' به S نزدیک‌تر است که با فرض ما در تناقض است. از این تناقض بهینه بودن الگوریتم حاصل می‌شود.

(ب) اینجا نیز از ایده مشابهی استفاده می‌کنیم: فرض کنید در پمپ بنزین i ایستاده‌ایم. در این پمپ بنزین به آن اندازه بنزین می‌زنیم که بتوانیم به نخستین پمپ بنزینی برسیم که قیمت بنزین در آن ارزان‌تر است و اگر نتوانیم این کار را انجام دهیم، باک را پر می‌کنیم و به پمپ بنزین بعدی می‌رویم. برای اثبات بهینه بودن، مشابه قسمت قبل S و S' را تعریف میکنیم. نخستین جایی را در نظر بگیرید که S و S' مثل هم عمل نمی‌کنند. دو حالت ممکن است رخ دهد:

i. در صورتی که باک را کامل پر کرده چون نمی‌توانسته با یک باک پر به پمپ بنزینی با هزینه کمتر برود: در این صورت S' حتما بنزین کمتری زده که باعث می‌شود که در پمپ بنزین گرانتری بنزین بزند. در اینجا می‌توان S' را با کامل بنزین زدن در این پمپ بنزین بهتر کرد که با فرض در تناقض است.

ii. در صورتی که به میزانی بنزین زده که به نخستین پمپ بنزین ارزان‌تر برسد: اگر S' کمتر بنزین زده باشد که مجبور است در پمپ بنزین گرانتری بایستد و بنزین بزند که می‌توان آن را بهتر کرد. در صورتی که بنزین بیشتری زده باشد هم می‌توان بنزین اضافه را در پمپ بنزین ارزانتری زد که باز هم در تناقض است.

از تناقض در هر دو حالت، بهینه بودن الگوریتم نتیجه می‌شود.

۴. ابتدا همه ي بازه ها را بر اساس زمان شروع مرتب می کنیم و در یک آرایه می ریزیم. براي انتخاب اولین محافظ (بازه) باید بازه اي را انتخاب کنیم که شروع آن a باشد. پس از اول آرایه حرکت می کنیم و از بین تمام بازه هایی که ابتداي آن ها a است را انتخاب می‌کنیم و باقی بازه‌ها را حذف می‌کنیم. براي بازه ي دوم دوباره به حرکت ادامه می دهیم و بزرگترین بازه اي که شروع آن ها از اتمام بازه قبلی(F_0) کوچکتر مساوی و پایان آنها بزرگتر مساوی F_0 است را در نظر می‌گیریم. این روند را تا جایی ادامه می‌دهیم تا جایی ادامه می‌دهیم که بازه‌ای با پایان بزرگتر مساوی b انتخاب شود.

براي اثبات درستی الگوریتم از برهان خلف استفاده میکنیم: فرض میکنیم که پاسخ تولید شده توسط الگوریتم بهینه نیست. هر پاسخ به مسئله را توسط یک لیست مرتب صعودي طبق زمان شروع از بازه هاي موجود در آن پاسخ نشان میدهیم، به عنوان مثال پاسخ تولید شده توسط الگوریتم را توسط $S =< I_1 I_2 \cdots, I_k >$, نشان می‌دهیم. دقت کنید که با توجه به نحوه ساخته شدن S لیست بازه ها مرتب صعودي طبق زمان پایان هم هست. حال شبیه ترین پاسخ به S را $S' =< I'_1, \cdots, I'_m >$ تعریف می‌کنیم به صورتی که کوچکترین i که $I_i \neq I'_i$ است، بیشینه باشد. با توجه به اینکه S' یک پاسخ بهینه است میدانیم که $< I'_1, \cdots, I'_m >$ مرتب صعودی طبق زمان پایان هم هست چرا که اگر برای یک j داشته باشیم $F_{I'_{j-1}} \geq F_{I'_j}$ آنگاه با حذف بازه I'_j به یک پاسخ معتبر با تعداد کمتري بازه میرسیم. حال پاسخ S'' را با جایگزین کردن I'_i توسط I_i در S' می‌سازیم. میدانیم که S'' یک پاسخ معتبر است زیرا $S_{I_i} \leq S_{I'_{i-1}}$ و همچنین با توجه به نحوه انتخاب I_i حتما داریم $F_{I_i} \geq F_{I'_i} \geq S_{I'_{i+1}}$. از طرف دیگر S'' یک پاسخ بهینه است چرا که تعداد بازه هایش با S' مساویست اما این با این فرض که S' شبیه ترین پاسخ بهینه به S است در تناقض است. در این الگوریتم ابتدا $O(nlogn)$ هزینه براي مرتب سازي و یکبار پیمایش آرایه (n) نیاز است. پس هزینه ي کل برابر است با $O(nlogn)$.

۵. (آ) کاری که کمترین تعداد روز را نیاز دارد را انتخاب می‌کنیم و آن را انجام می‌دهیم. در روزی که کار تمام می‌شود، تعدادی کار باقی می‌ماند که به صورت بازگشتی این الگوریتم را روی آن اجرا می‌کنیم. برای اثبات بهینه بودن الگوریتم، فرض کنید راه‌حل بهینه S وجود دارد که کار i را اول انجام داده و از روز ۱ تا d_i مشغول کار i بوده و کار با کمترین تعداد روز لازم(کار ۱) از روز d تا $d + d_1$ انجام شده است. به سادگی قابل مشاهده است که اگر جای کار i و ۱ را با یکدیگر عوض کنیم، پروژه‌های بین این دو، زودتر تحویل داده می‌شوند و کارهای بعد از کار ۱ نیز تغییری نمی‌کنند و لذا راه‌حل بهتر می‌شود که با فرض در تناقض است. از این تناقض بهینه بودن الگوریتم نتیجه می‌شود.

(ب) برای این حالت نیز کافی است کار با کمترین روز موردنیاز انتخاب شود. هر زمان که کار در حال انجام تمام شد یا کار جدیدی به ما داده شد، این انتخاب را دوباره انجام می‌دهیم(کاری که d روز از آن مانده است، مشابه کاری است که d روز وقت میگیرد). اثبات درستی این الگوریتم مشابه حالت قبل است.