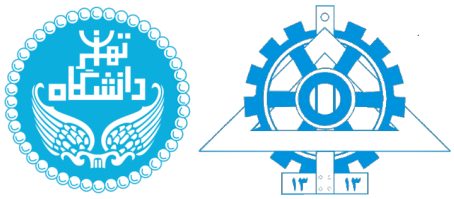
به نام خدا



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

پاسخ تمرین کتبی سوم (الگوریتم‌های حریصانه)
طراح: علیرضا توکلی، AlirezaTa3akoli@gmail.com

---

۱. (آ)

i. در هر مرحله با ارزش‌ترین شئ برداشته شود. مثال نقض به این صورت خواهد بود که اگر حجم کوله‌پشتی $V = 10$ باشد و شئ اول به صورت $c_1 = 7, v_1 = 10$ باشد و ۱۰ شئ دیگر به صورت $c = 1, v = 1$ داشته باشیم، بهتر است که ۱۰ شئ برداشته شود؛ ولی الگوریتم شئ اول را انتخاب می‌کند.

ii. در هر مرحله کم حجم‌ترین شئ برداشته شود. مثال نقض به این صورت خواهد بود که دوباره اگر حجم کوله‌پشتی $V = 10$ باشد و شئ اول به صورت $v_1 = 10, c_1 = 100$ باشد و ۱۰ شئ دیگر به صورت $v = 1, c = 1$ داشته باشیم، بهتر است که شئ اول برداشته شود؛ ولی الگوریتم اشیاء دیگر را انتخاب می‌کند.

(ب) ابتدا تمامی اشیائی که حجم‌شان از حجم کوله‌پشتی بیش‌تر است را کنار می‌گذاریم. از بین اشیاء باقی‌مانده، شئی را با بیش‌ترین چگالی (نسبت ارزش به حجم) در نظر بگیرید. در صورت وجود چندین شئ، شئ با حجم کم‌تر انتخاب شود. از این شئ آن‌قدر در کوله‌پشتی می‌گذاریم تا دیگر نتوان از این شئ در آن جا داد. حال ادعا می‌کنیم اگر جواب بهینه M باشد، کوله‌پشتی ما حداقل ارزش $\frac{M}{2}$ دارد.

می‌دانیم که ما پس از این کار، حداقل نصف کوله‌پشتی را پر کرده‌ایم. زیرا اگر نصف کوله‌پشتی پر نشده بود، می‌توانیم باز هم از این شئ به کوله‌پشتی اضافه کنیم. پس بیش‌تر از نصف کوله‌پشتی از جسمی با بیش‌ترین چگالی پر شده است. حال اگر جواب بهینه را در نظر بگیریم، حداکثر همین مقدار از حجم کوله‌پشتی توسط همین چگالی و باقی‌مانده‌ی آن توسط اشیائی با چگالی کم‌تر پر شده است. پس جواب ما حداقل نصف جواب واقعی است.

توجه شود که الگوریتم استفاده شده تنها طول آرایه را طی می‌کند. پس جواب از مرتبه $O(n)$ خواهد شد.

۲. گربه‌ها را بر اساس وزن‌شان مرتب می‌کنیم. متغیری ($last$) خواهیم داشت که نشان می‌دهد آخرین گربه‌ای که وزن نهایی‌اش را مشخص کرده‌ایم، چه وزنی داشته است. در ابتدا برای این که متغیر تأثیری نگذارد، $last = 0$ قرار می‌دهیم. حال از کم وزن‌ترین گربه شروع می‌کنیم. هر دفعه اگر وزن گربه‌ی مورد بررسی w باشد و $last < w - 1$ باشد، گربه را رژیم می‌دهیم و وزن آن $w - 1$ می‌شود. در صورتی که $last = w - 1$ باشد، گربه در همین وزن مانده و در صورتی که $last = w$ باشد، وزن گربه را یک واحد زیاد می‌کنیم. توجه کنید که در انتهای هر تصمیم‌گیری باید متغیر $last$ را به وزن نهایی به روز رسانی کنیم. تعداد گربه‌های متفاوت نیز برابر با تعداد گربه‌هایی خواهد بود که $last \neq w$ است.

توجه شود که در این الگوریتم تنها یک مرتب کردن و یک حرکت روی اعضای آرایه را داشتیم. پس مرتبه‌ی زمانی $O(n \log n)$ خواهد شد.

۳. کلیت الگوریتم به این صورت است که از ابتدا تا جایی که ماشین تکراری ببینیم را یک بازه می‌کنیم. پس از آن دوباره همین فرآیند را تکرار می‌کنیم تا ماشین‌ها تمام شوند. اما در ادامه جزئیات این الگوریتم گفته می‌شود. فرض کنید آرایه‌ی کمکی B را داریم که طول آن به اندازه‌ای است که $B[a_i]$ معتبر باشد. از اولین ماشین شروع کرده و اگر در حال بررسی ماشین iام باشیم و $B[a_i] = 1$ باشد، بازه‌ی فعلی را به عنوان یک بازه اعلام کرده و به ازای تمامی a_i های این بازه، $B[a_i] = 0$ را انجام می‌دهیم که حواسمان باشد از این نوع ماشین استفاده نکردیم. الگوریتم تنها یک بار تمامی ماشین‌ها را می‌بیند، پس مرتبه زمانی آن $O(n)$ خواهد بود.

۴. برای این که جایگاه هر عدد در آرایه‌های A و B یادمان بماند، هر عضو را به همراه جایگاه‌ش به صورت زوج مرتب می‌نویسیم. سپس هر دوی A و B را به صورت نزولی مرتب کرده و اشاره‌گرهایی به هر کدام از آن‌ها به نام‌های pA و pB داریم که در ابتدا روی اولین عناصر هر کدام از آن‌ها هستند. حال تا جایی که $A[pA] \leq B[pB]$ است، pB را زیاد می‌کنیم. اگر pB به انتها رسید که کار تمام است. در

غیر این صورت، به حالتی می‌رسیم که $A[pA] > B[pB]$ است. تعداد سکه‌ها را یکی زیاد کرده و از آن‌جایی که اندیس اولیه‌ی این دو عنصر را داریم، می‌توانیم کاری کنیم که در جایگشت اندیس‌هایشان یکی شود. پس از این کار pA را یکی زیاد کرده و همین فرآیند را ادامه می‌دهیم تا زمانی که یا حالت گفته شده پیش بیاید و یا pA به انتهای A رسیده باشد.

در این الگوریتم از مرتب کردن استفاده شد پس مرتبه زمانی الگوریتم $O(n \log n)$ خواهد شد.

۵. لم ۱: هر فرد حداکثر دو گزینه برای رفتن به طباخی دارد. زیرا فرض کنید حداقل سه گزینه داشته باشد. از آن جایی که تمامی طباخی‌ها روی یک خط قرار دارند، دومین طباخی از یکی از دو طباخی دیگر نزدیک‌تر است و به تناقض می‌رسیم.

لم ۲: در صورتی که y فرد B بین A و طباخی C قرار گرفته باشد. ($y_A < y_B < y_C$) هم‌چنین طباخی C نزدیک‌ترین طباخی به A باشد، A قادر نیست مجانی کله‌پاچه بگیرد. زیرا فرد B زودتر به کله‌پاچه می‌رسد و نزدیک‌تر است. هم‌چنین نزدیک‌ترین طباخی به فرد B همان طباخی C است.

طباخی‌ها که در آرایه‌ی A هستند و افراد که در آرایه‌ی B هستند را بر حسب y مرتب می‌کنیم. هم‌چنین به ازای هر طباخی وکتوری داریم که در آن افرادی که به سوی این طباخی می‌آیند را می‌ریزیم. اسم این وکتور V است. برای هر فرد توسط باینری سرچ روی اعضای A می‌توانیم طباخی‌های نزدیک به آن فرد را پیدا کنیم. اگر تنها یک طباخی به این فرد نزدیک بود، به V طباخی مورد نظر این فرد را اضافه می‌کنیم. زیرا جای دیگری وجود ندارد که برود. حال برای افرادی که دو گزینه دارند، گزینه‌ای که طبق لم ۲ منقضی می‌شود را حذف می‌کنیم. یک بخشی از این افراد یک گزینه‌ای می‌شوند و مانند یک گزینه‌ای‌ها با آن‌ها برخورد می‌کنیم. اما دو گزینه‌های باقی‌مانده را چه کنیم؟ آن‌ها را بر حسب y مرتب کنید. از کوچک‌ترین y شروع کرده و بین دو طباخی مورد نظر، اگر طباخی با y کم‌تر، فردی نزدیک‌تری نداشت که به آن می‌خواست برود، به سمت آن طباخی بفرستیم. در غیر این صورت، به طباخی بعدی می‌فرستیم. این کار را تا انتها انجام می‌دهیم و به ازای هر طباخی مشخص می‌کنیم از بین گزینه‌های موجود، کدام یک زودتر به مقصد می‌رسند. مرتبه‌ی زمانی الگوریتم به دلیل مرتب کردن آرایه‌ها $O(n \log n + m \log m)$ خواهد بود.

اثبات الگوریتم: فرض کنید جواب بهینه با الگوریتم ما متفاوت باشد. می‌دانیم که جواب بهینه برای افرادی که یک گزینه دارند و کسانی که طبق لم ۲ یک گزینه دارند، باید حتما مشابه ما باشد. (زیرا تنها یک حالت برای انجام دادن این کار وجود دارد.) پس تفاوت در دو گزینه‌ای‌ها خواهد بود. فرض کنید به ترتیب الگوریتم ما چک می‌کنیم که آیا انتخاب ما با انتخاب بهینه تفاوت دارد یا خیر. اولین جایی را در نظر بگیرید که تفاوت پیدا می‌کند. این تفاوت دو نوع ممکن است باشد:

- ما این فرد را به طباخی با y کم‌تر فرستاده‌ایم اما الگوریتم بهینه او را به طباخی با y بیش‌تر فرستاده است. در این صورت، طبق الگوریتم گفته شده، ما مطمئن خواهیم بود که فرد با رفتن به طباخی با y کم‌تر حتما مجانی کله‌پاچی می‌خورد. پس اگر در جواب بهینه نیز این فرد را به این طباخی بفرستیم، مشکلی پیش نخواهد آمد.
- ما این فرد را به طباخی با y بیش‌تر فرستاده‌ایم اما الگوریتم طباخی با y کم‌تر را انتخاب کرده است. از آن‌جایی که در الگوریتم ما تنها در صورتی فرد به y بیش‌تر می‌رفت که اگر به طباخی با y کم‌تر می‌رفت، کله‌پاچه‌ی مجانی نمی‌توانست بگیرد. پس اگر در جواب بهینه این فرد را به طباخی با y بیش‌تر بفرستیم، جواب بیش‌تر مساوی خواهد شد.

پس الگوریتم ما جواب درست را در خروجی می‌دهد.

۶. الگوریتم به این صورت خواهد بود که لیستی از زیردنباله‌های منتهی به ۰ و لیستی از زیردنباله‌های منتهی به ۱ داریم. از ابتدا شروع کرده و هرگاه ۱ دیدیم یکی از عناصر لیست ۰ ها را یک به انتهایش اضافه کرده و در لیست ۱ ها می‌گذاریم. هر گاه ۰ دیدیم، اگر لیست ۱ ها خالی نبود، یکی از آن‌ها را گرفته و ۰ به انتهایش اضافه کرده و در لیست صفرها می‌گذاریم و در غیر این صورت یک ۰ به لیست صفرها اضافه می‌کنیم. واضح است که مرتبه‌ی الگوریتم از مرتبه‌ی $O(n)$ است. الگوریتم اگر در زمانی نتواند کار گفته شده را انجام دهد، یا در انتها لیست ۱ هایش خالی نباشد، جواب خیر خواهد بود و در غیر این صورت جواب بله است و لیست زیردنباله‌ها مشخص خواهد بود.

حال الگوریتم را اثبات می‌کنیم. در صورتی که جواب الگوریتم بله باشد که مشکلی نیست و زیردنباله‌ها را مشخص کرده‌ایم. حال فرض کنید الگوریتم خیر می‌گوید اما افرازی وجود دارد. مرحله به مرحله جلو می‌رویم تا به جایی برسیم که الگوریتم ما با جواب تفاوت داشته باشد. دو حالت داریم:

- عدد فعلی یک است. دو الگوریتم این عدد را به انتهای یک زیردنباله باید اضافه کنند. پس تفاوت الگوریتم ما با جواب این است که ما یک را به انتهای یک زیردنباله و جواب آن را به انتهای یک زیردنباله‌ی دیگر اضافه کرده است. اگر در ادامه‌ی جواب اصلی ما بیایم و جای این دو زیردنباله را با هم عوض کنیم، تغییری در جواب اصلی رخ نمی‌دهد. پس فرقی نمی‌کند که ۱ به کدام زیردنباله اضافه شده باشد.

- عدد فعلی صفر است. اگر الگوریتم ما و جواب اصلی هر دو این ۰ را به انتهای یک دنباله‌ی ۱ اضافه کرده باشند که به دلیل گفته شده در مورد قبلی، مشکلی نخواهیم داشت. پس تنها حالتی که می‌ماند این است که الگوریتم ما این صفر را به انتهای یکی از اعضای لیست ۱ ها اضافه کرده باشد و جواب اصلی آن را یک زیردنباله‌ی جدید در نظر گرفته است. در این صورت از آن جایی که در جواب اصلی یک زمانی یک صفری به انتهای ۱ اضافه خواهد شد، پس می‌توانیم جای این دو صفر را با هم عوض کنیم و مشکلی پیش نیاید.

پس الگوریتم ما جواب درست را خروجی می‌دهد.