به نام خدا

تمرین چهارم درس ساختمان داده‌ها و الگوریتم‌ها

چمران معینی

۹۹۳۱۰۵۳

۳- **تیم هماهنگ**: نرگس به تازگی به عنوان مدیر پروژه برای پیاده سازی نرم افزار تلفن همراه کاربران شرکت انتخاب شده است و او باید اعضای تیم خود برای این کار را انتخاب کند. نرگس این اعتقاد را دارد که تیمی هماهنگ است که هر فرد آن، حداقل با k فرد دیگر تیم (به غیر از نرگس) قبلا در یک تیم کار کرده باشد. نرگس باید تیم خود را از بین n کارمند شرکت (به غیر از خودش) انتخاب کند. شرکت یک پایگاه داده دارد که اطلاعات همکاری افراد با یکدیگر در پروژه های مختلف را در خود دارد. برای استفاده از این سامانه، می‌توان از ۴ دستور زیر استفاده کرد:

الف) *مقدار دهی اولیه*: تمامی افراد قابل انتخاب در شرکت را به عنوان اعضای تیم اضافه می کند. پس از آن، می توان سه تابع دیگر را صدا زد. زمان اجرای این تابع $O(n^2)$ است.

ب) *تعداد افراد همکاری کرده با فرد x*: برای فرد x داده شده که هنوز در تیم است، تعداد افراد دیگر موجود در تیم را که x قبلا با آنها همکاری کرده است را باز می‌گرداند. زمان اجرای این تابع $O(1)$ است.

ج) *حذف فرد x از گروه*: فرد x را از تیم حذف می کند. پس از این کار، در محاسبه همکاری افراد (تابع ب)، این فرد دیگر در نظر گرفته نمی‌شود. اگر فرد x در هنگام حذف، با m نفر دیگر از افراد تیم موجود همکاری داشته است، زمان اجرای این تابع $O(m)$ خواهد بود.

د) *گرفتن لیست افراد موجود در تیم*: لیستی از افرادی که در حال حاضر در تیم هستند را بر می گرداند. اگر در حال حاضر p فرد در تیم هستند، زمان اجرای این تابع $O(p)$ خواهد بود.

به نرگس کمک کنید تا یک الگوریتم با زمان کلی $O(n^2)$ طراحی کند که تشخیص بدهد آیا نرگس می‌تواند تیمی با مشخصات مورد نظرش تشکیل بدهد یا نه. و اگر بله، بزرگترین تیم ممکن (از نظر تعداد افراد) را تشکیل دهد. دقت کنید که ممکن است بیش از یک تیم قابل تشکیل باشند و تشکیل یکی از آنها کافی است.

ابتدا تابع الف صدا می‌شود.

پس از آن، تابع ب را برای تمام اعضای دیگر صدا می‌کنیم و اگر مقداری که برای هر عضو برمی‌گرداند کمتر از k بود، تابع ج را برای آن فرد اجرا می‌کنیم.
سپس تابع د را صدا می‌زنیم و روی لیستی که باز گردانده است، همین فرآیند را دوباره تکرار می‌کنیم.

یا این فرآیند آن‌قدر تکرار می‌شود که دیگر هیچ عضوی در تیم باقی نماند که در این صورت یعنی چنین چیزی ممکن نیست، یا به جایی می‌رسم که هیچ عضوری در طول فرآیند حذف نمی‌شود، که یعنی اعضای باقیمانده، بزرگ‌ترین گروه ممکن هستند.

۴- **گزارش عملکرد**: کاظم چند سالی است که مدیر عامل شرکت است. در طول این مدت، آنها n قرارداد با مشتری ها امضا کرده‌اند و ارزش قرارداد iام برابر با v_i است. فرض کنید که قراردادها به ترتیب زمان امضا هستند، یعنی اگر $i<j$، آنگاه قرارداد iام قبل از قرارداد jام امضا شده است (هرچند ارزش قرارداد iام می‌تواند کمتر، مساوی، و یا بیشتر از قرارداد jام باشد.) قرار است هفته آینده او ارائه‌ای برای گروهی از سرمایه گذاران داشته باشد. کاظم می‌خواهد که به جای نمایش و ذکر تمامی قراردادها، زیر مجموعه‌ای از آنها را بیان کند که ارزش آنها اکیدا صعودی باشد. به عبارت دیگر، او میخواهد مجموعه ای از قراردادها مانند $i_1, i_2, ..., i_k$ را انتخاب کند که $i_1<i_2<...<i_k$ و همچنین $v_{i1}<v_{i2}<..._v_{ik}$. هدف کاظم از این کار ارائه یک نمای رو به رشد از شرکت (با توجه به افزایش ارزش قراردادها در طول زمان) است. یک الگوریتم برای کمک به کاظم طراحی کنید که بزرگ‌ترین زیرمجموعه قراردادها را که شرط مورد نظر را دارند، پیدا کند. زمان الگوریتم شما باید $O(n^2)$ باشد .

نکته: دقت کنید که این مسئله را می‌توان در زمان $O(nlgn)$ حل کرد. برای این تمرین لازم نیست این کار را بکنید، اما می‌توانید روی آن هم فکر کنید!

اولین راه حلی که برای حل این سوال به نظر می‌رسد، این است که تمام زیرمجموعه‌های ممکن را بررسی کنیم و اگر صعودی بودند، طول آن‌ها را محاسبه کنیم و سپس از بین نتایج، ماکسیمم را به عنوان نتیجه اعلام کنیم، اما این روش را نمی‌توان در O(n2) انجام داد. با کمی دقت، متوجه می‌شویم که در این روش، بسیاری از مواقع برخی از محاسبات چند بار انجام می‌شوند. برای مثال دنباله‌ی زیر را در نظر بگیرید:

1, 2, 4, 3, 6, 7

در این دنباله، می‌دانیم که طولانی‌ترین دنباله‌ی اکیدا صعودی که از ۱ تا ۴ وجود دارد، به طول ۳ است، اما در محاسبه‌ی طول 1, 2, 4, 6, 7 و 1, 2, 3, 6, 7 ، دوبار این شمارش تکرار شده است. پس از برنامه نویسی پویا کمک می‌گیریم تا برای مثال هرگاه که خواستیم اعداد چهار و چهار به بعد را بررسی کنیم، بدانیم که بلندترین دنباله‌ی اکیدا صعودیِ قبل از ۴ ، به طولِ ۳ است.

می‌دانیم که هر عضو، حداقل زیرمجموعه‌ای اکیدا صعودی به طول یک دارد (مجموعه‌ای که تنها شامل خود عدد باشد) ، پس ابتدا آرایه‌ی dp را به طول n تعریف می‌کنیم و تمام اعضای آن را ۱ می‌گذاریم. قرار است که عضو i ام از این دنباله، نشان‌دهنده‌ی این باشد که طولانی‌ترین دنباله‌ی اکیدا صعودی که به این عضو ختم می‌شود، چه طولی دارد.

این مقدار برای اولین عضو همواره ۱ است، یعنی همیشه dp[0]=1

برای عناصر بعدی، یک واحد به بزرگ‌ترین dp ی قبلی‌شان که مقدار v اش از v عنصر مورد نظر کوچک‌تر باشد، اضافه می‌کنیم. برای ۲ ، این مقدار یکی بیشتر از dp ی ۱ خواهد بود، یعنی همان ۲ . برای ۴ این مقدار یکی بیشتر از دی‌پیِ ۲ خواهد بود، یعنی ۳ ، برای ۳ هم این مقدار یکی بیشتر ای دی‌پیِ یک خواهد بود، یعنی ۳

برای ۶ ، این مقدار برابر دی‌پیِ ۴ به علاوه‌ی یک خواهد بود، یعنی همان ۵ و برای هفت هم دی‌پی۶ به علاوه‌ی یک را خواهیم داشت که برابر ۵ می‌شود.