برنامه سازی پیشرفته — تمرین شماره‌ی ۷ - فاز ۲

مدرس: **رامتین خسروی**

طراحان: سید احمد عبداله پوری حسینی

مهلت تحویل:پنج شنبه ۲۲ خرداد، ساعت ۲۳:۵۵

UTRIP, but a little smarter!

مقدمه

هدف از این فاز پروژه این است که در راستای باهوش تر کردن برنامه‌ای که در فاز قبل نوشتید یک سری امکانات به آن اضافه بکنیم. برای پیاده‌سازی این امکانات از داده‌هایی که خود برنامه تولید می‌کند استفاده خواهیم کرد.

این فاز نیز مشابه فاز قبل تمرکز ویژه‌ای بر روی بخش خاصی از درس ندارد و در نتیجه انتظار می‌رود که در طراحی و کدزدن آن از آموخته‌های خود در تمام طول درس استفاده بکنید.

توجه داشته باشید که تمامی تعاملات با کاربر همچنان از طریق خط فرمان[1] انجام خواهد شد و رابط کابری گرافیکی[2] مختص فاز آخر خواهد بود.

همچنین در این فاز شما باید اطلاعات هتل‌ها را از فایل جدیدی که به همراه صورت پروژه در اختیارتان قرار گرفته است، به اسم hotels.csv بخوانید. روش خواندن اطلاعات از این فایل کاملا مشابه فاز قبل است، به جز اینکه اسم چند تا از ستون‌ها نسب به فاز قبل تغییر کرده است. که البته اگر طراحیتان خوب بوده باشد، اعمال این تغییرها در کدتان باید کار بسیار ساده‌ای باشد.

به عنوان آخرین نکته توجه کنید که در پیاده‌سازی این فاز کتابخانه‌ی <algorithm> در جاهای مختلفی می‌تواند به ساده‌تر، خواناتر و احتمالا سریع‌تر شدن کدتان کمک بکند. توصیه می‌شود قبل از شروع به پیاده‌سازی یک نگاه کلی به توابع این کتابخانه، و به طور خاص توابع زیر بیندازید:

- transform
- copy / copy_if
- sort / stable_sort
- accumulate
- all_of / any_of
- find / find_if

از شما انتظار می‌رود که در جاهایی از کد که می‌توان به وضوح از یکی از این توابع استفاده کرد تا کد کم حجم تر و خوانا تر بشود، از این توابع استفاده کنید و نسخه‌ی مشابه آن‌ها را خودتان مجددا پیاده‌سازی نکنید.

در ادامه به سراغ بخش‌های اصلی که باید برای فاز ۲ پیاده‌سازی کنید می‌رویم.

---

[1] command line

[2] graphical user interface (GUI)

بودجه‌ی پیش‌فرض

در فاز قبل، برای برنامه‌ی utrip فیلترها را پیاده‌سازی کردید، از جمله فیلتر میانگین هزینه‌ی اتاق‌ها.اما نکته‌ای در رفتار کاربران وجود دارد که به ما اجازه می‌دهد با استفاده از این فیلتر لیست هتل‌هایی که به کاربران نشان می‌دهیم را مفید تر و خلاصه تر بکنیم. آن نکته این است که درآمد یک فرد و در نتیجه قیمت اتاق‌هایی که رزرو می‌کند در زمان کوتاه تغییر زیادی نمی‌کنند. در نتیجه می‌توانیم با اضافه کردن یک فیلتر پیش‌فرض روی قیمت اتاق‌ها، هتل‌هایی که قیمتشان با تاریخچه‌ی رزروهای کاربر تفاوت زیادی دارد را حذف بکنیم.

اما چطور تصمیم بگیریم کدام هتل ها قیمتشان به قدری متفاوت هست که باید حذف شوند ؟ در اینجا می‌توانیم از علم آمار کمک بگیریم. به طور خاص، در اینجا از شاخصه‌های میانگین و انحراف معیار[3] استفاده خواهیم کرد. به این صورت که قیمت تک تک اتاق‌هایی که فرد تا به حال رزرو کرده را به عنوان نمونه[4] در نظر گرفته، و میانگین نمونه (میانگین قیمت اتاق‌ها) و انحراف معیار نمونه (انحراف معیار قیمت اتاق‌ها) را برای آن محاسبه می‌کنیم. در نهایت فقط هتل‌هایی را به کاربر نمایش خواهیم داد که فاصله‌ی میانگین قیمت اتاق‌هایشان از میانگین نمونه،کمتر یا مساوی ۲ برابر انحراف معیار باشد، یعنی:

$$|p - m_s| \leq 2 \times s_s$$

که در آن p میانگین قیمت اتاق‌های هتل، m_s میانگین نمونه، و s_s انحراف معیار نمونه است. توجه کنید که انحراف معیار نمونه با انحراف معیار جمعیت[5] فرق دارد و از فرمول زیر محاسبه می‌شود:

$$s_s = \sqrt{\frac{(\sum x_i - m_s)^2}{N-1}}$$

که در آن، x_i ها قیمت اتاق‌ها، و N اندازه‌ی نمونه (تعداد اتاق ها) است.

اما این فیلتر چه زمانی فعال خواهد بود ؟ وقتی که ۳ شرط پایین برقرار باشند:

۱- کاربر خود هیچ فیلتری روی قیمت اتاق ها نذاشته باشد.

۲- کاربر حداقل ۱۰ reservation داشته باشد (چرا که این فیلتر مبتنی بر داشتن تعداد معقولی reservation برای تخمین مقادیر میانگین و انحراف معیار است.)

۳- این فیلتر توسط کاربر خاموش نشده باشد.

در توضیح مورد ۳ باید گفت که این فیلتر، توسط کاربر به شکل زیر قابل روشن و خاموش کردن است:

ورودی	خروجی
`POST default_price_filter ? active true|false`	`OK` | Bad Request | Permission Denied

[3] standard deviation
[4] sample
[5] population

بعد از عبارت active باید کلمه‌ی true یا false بیاید. اگر کاربر هیچ وقت از این دستور استفاده نکرده باشد، به صورت پیش‌فرض، فیلتر روشن خواهد بود. طبیعتا اگر کاربر لاگین نکرده باشد، یا فرمت دستور با فرمت مشخص شده هماهنگ نباشد، باید پیغام خطای نظیر چاپ شود.

همچنین اگر این فیلتر فعال باشد (هر ۳ شرط برقرار باشند)، هر زمانی که کاربر از دستور GET hotels استفاده می‌کند، باید در یک خط بالای نتایج عبارت زیر نوشته شود:

* **Results have been filtered by the default price filter.**

علت این کار این است که به عنوان یک قاعده‌ی کلی، خوب نیست که یک سیستم به صورت پیش فرض بخشی از نتایج را به کاربر نشان نداده و هیچ هشداری هم به کاربر در این باره ندهد. چرا که در این صورت، ممکن است کاربر فکر کند که آن نتایج محدود شده، کل نتایج موجود در سیستم هستند.

برای روشن شدن بیش‌تر توضیحات بالا، چند تست کیس نمونه در پوشه‌ی testcases/default_price_filter در اختیار شما قرار داده شده است، که به دلیل حجم بالا در داخل صورت پروژه آورده نشده است.

در مورد تست کیس اول، توجه بکنید که قیمت اتاق های رزرو شده به شکل زیر هستند:

{200 x 1, 210 x 3, 350 x 1, 90 x 2, 66 x 1, 70 x 1, 85 x 1, 90 x 3, 200 x 1, 70 x 1}

عدد سمت راست x تعداد اتاق، و سمت چپ قیمت اتاق است، در نتیجه میانگین و انحراف معیار باید به ترتیب مقادیر 134.73 و 67.81 باشد.

مرتب سازی

در این بخش باید یک دستور برای مرتب سازی لیست هتل‌های نمایش داده شده، بر اساس مقدار یک خصیصه‌ی مشخص اضافه کنید. مشابه بخش فیلترها، در این بخش نیز ابتدا کاربر اطلاعات مربوط به مرتب سازی را توسط دستور وارد می‌کند، و نتیجه را پس از استفاده از دستور GET hotels می‌بیند. این اطلاعات شامل خصیصه‌ای که مرتب سازی بر اساس آن صورت می‌گیرد، و ترتیب مرتب سازی است:

ورودی	خروجی
`POST sort ? property <property> order ascending|descending`	`OK` | Bad Request | Permission Denied

مقدار <property> می‌تواند یکی از مقادیر زیر باشد:

- id
- name
- star_rating
- city: مرتب سازی در این مورد باید مشابه id ها بر اساس ترتیب lexicographic باشد.
- standard_room_price
- deluxe_room_price
- luxury_room_price

- premium_room_price
- average_room_price: میانگین قیمت اتاق‌های یک هتل،مطابق توضیحات فاز ۱.

اگر مقدار داده شده در لیست بالا نبود، باید خطای Bad Request چاپ شود. توجه کنید که هتل‌هایی که مقدار خصیصه‌ی مشخص شده برای هر دو برابر است، باید بر اساس id و به صورت صعودی مرتب شوند.

اگر هیچ دستور sort ای قبل از استفاده از GET hotels داده نشده بود، هتل ها باید مثل فاز ۱ مرتب سازی شوند (بر اساس id ، صعودی).

همچنین توجه کنید که نیازی نیست الگوریتم sort را خودتان پیاده‌سازی کنید، و می‌توانید از کتابخانه‌های استاندارد c++ برای اینکار استفاده کنید. در واقع چیزی که مهم است، طراحی خوب، و کد خوانا است، و می‌توانید از هر روشی که بهتر این دو هدف را محقق می‌کند استفاده کنید.

تست کیس‌های نمونه‌ی این بخش در پوشه‌ی testcases/sort قرار دارند که باز هم به دلیل حجم بالا، در صورت پروژه آورده نشده اند.

این بخش (مرتب سازی) برنامه را برای کاربر usable تر می‌کند، و تقریبا می‌توان گفت مرتب سازی برای هر برنامه‌ای که لیستی از نتایج را به کاربر نشان می‌دهد ضروری است، ولی استفاده‌ی اصلی این بخش در فراهم کردن مقدمه برای بخش آخر است.

تغییر روش محاسبه‌ی امتیاز هتل ها

با توجه به رشد کاربران برنامه‌ی Utrip، تیم فنی تصمیم گرفته که برای بالاتر بردن سرعت محاسبه‌ی میانگین امتیازات هتل ها، و با توجه به اینکه up-to-date بودن این امتیازات اهمیت زیادی ندارد، بهتر است که به جای محاسبه‌ی مستقیم میانگین‌ها، آن‌ها صرفا از یک فایل خوانده شوند که در زمان‌های مشخصی به روزرسانی می‌شود. البته شما نیاز نیست نگران به‌روزرسانی فایل باشید، شما صرفا از فایل استفاده می‌کنید.

دومین آرگومان خط فرمان[6] که به برنامه‌ی شما داده می‌شود آدرس فایلی خواهد بود که حاوی این امتیازات است. کاری که شما باید بکنید این است که کد مربوط به محاسبه‌ی میانگین امتیازات هتل را حذف کرده، و به جای آن میانگین امتیازات هتل ها را از فایلی که آدرس آن را دریافت کرده اید بخوانید. این فایل همیشه یک فایل csv خواهد بود، و توضیح ستون های آن در ادامه آمده است:

۱- unique_id: شناسه‌ی یکتای هتل.

۲- rating_location: میانگین امتیازات کاربران در مورد location این هتل.

موارد ۳، ۴، ۵ و ۶ همگی مشابه مورد ۳ هستند و میانگین امتیازات کاربران را در مورد این هتل در ضمینه‌ی مشخص شده را نشان می‌دهند.

۷- rating_overall: میانگین سطح رضایت کلی کاربران از هتل.

در ادامه یک تست کیس ساده قرار داده شده است،که رفتار جدید برنامه را نشان می‌دهد. این تست کیس با این فرض است که کاربر login کرده، و برنامه در حال استفاده از فایل ratings.csv که در اختیارتان قرار داده شده است.

[6] command-line argument

ورودی	خروجی
GET ratings ? hotel 9f44a0a7acfd5ce8d7abe69978b2a9e0	location: 5.00 cleanliness: 3.50 staff: 2.90 facilities: 3.40 value_for_money: 4.30 overal_rating: 3.50

مرتب سازی بر اساس امتیاز و امتیازهای شخصی

با توجه به اینکه میانگین سطح رضایت کلی کاربران یک معیار خوب برای سنجش کیفیت هتل‌ها است، طبیعی است که بخواهیم قابلیت مرتب سازی بر اساس آن را داشته باشیم. در این بخش باید خصیصه‌ی rating_overall را به سایر خصیصه‌هایی که در دستور POST sort می‌توانستیم به برنامه بدهیم اضافه بکنید. با انتخاب این خصیصه، هتل ها باید بر حسب ستون rating_overall که در فایل حاوی امتیازها قرار دارد مرتب بشوند.

در ادامه می‌خواهیم کمی در مفهوم سطح رضایت کلی عمیق تر وارد شویم. این درست است که میانگین سطح رضایت‌کاربران می‌تواند معیار خوبی برای سنجش کیفیت یک هتل باشد، ولی رضایت افراد از یک هتل تا حد خوبی به سلیقه‌ی افراد نیز وابسته است. به طور مثال ممکن است برای یک فرد تمیزی هتل خیلی زیاد مهم باشد و در نتیجه اگر ایرادی در تمیزی هتل باشد، هر چقدر هم که امکانات دیگر هتل خوب باشد، مثل موقعیت، برخورد کارکنان، قیمت و ... باز هم سطح رضایت بسیار پایینی از هتل داشته باشد. یا ممکن است مورد مشابه برای فردی پیش بیاید که برخی امکانات جانبی هتل مثل اینترنت با کیفیت و یا داشتن سالن مطالعه برایش بسیار مهم است.

از طرفی دیگر، می‌دانیم نظرات افراد در مورد اینکه یک هتل «چقدر تمیز است»، یا اینکه «برخورد کارکنان چقدر خوب است» (امتیازات مربوط به هر دسته‌بندی)، کمتر از سلیقه تاثیر می‌پذیرند و کمابیش به هم نزدیک اند.

با توجه به این مقدمات، می‌خواهیم روشی به کار بگیریم که سطح رضایت یک فرد را از هتل‌هایی که هنوز به آن‌ها نرفته است تخمین بزند، و بر این اساس هتل‌ها را مرتب‌سازی بکند. روش ما به این صورت است که اولا سطح رضایت کلی را به صورت میانگینی وزن‌دار از امتیازهای هتل در دسته‌بندی‌های مختلف (موقعیت، تمیزی، برخورد کارکنان، ...) مدل‌سازی می‌کنیم که وزن هر دسته‌بندی نشان می‌دهد که آن دسته بندی چقدر برای فرد مهم است. در قدم بعدی سعی می‌کنیم با توجه به امتیازهایی که فرد تا به حال به هتل های مختلف داده است، و رابطه‌ی امتیازها در دسته‌بندی‌های مختلف با سطح رضایت کلی، این وزن‌ها را تخمین بزنیم تا بتوانیم برای سایر هتل‌ها استفاده کنیم. در نهایت برای یک هتل جدید که کاربر به آن امتیاز نداده است، می‌توانیم با میانگین وزن دار گرفتن از امتیازهای هتل در دسته‌بندی های مختلف ، و وزن‌هایی که تخمین زده‌ایم، سطح رضایت کاربر از آن هتل جدید را تخمین بزنیم.

بخش اول: میانگین وزن‌دار

در این بخش، باید سیستم «میانگین وزن دار» را پیاده‌سازی بکنید و در بخش بعد به سراغ تخمین وزن ها می‌رویم. در واقع فعلا فرض می‌کنیم که کاربر خودش وزن‌ها را به ما بدهد. برای این‌کار، شما ابتدا باید دستور جدیدی را پیاده‌سازی کنید که وزن‌های نظیر به هر دسته بندی را از کاربر دریافت کرده و ذخیره کند. این دستور دو حالت دارد که در زیر آمده است:

ورودی	خروجی
POST manual_weights ? active true location <location> cleanliness <cleanliness> staff <staff> facilities <facilities> value_for_money <value_for_money> POST manual_weights ? active false	OK | Bad Request | Permission Denied

در مورد حالت غیر فعال در بخش بعدی صحبت می‌کنیم و فعلا فرض می‌کنیم که کاربر قبل از اینکه لیست هتل‌ها را بخواهد حتما یک‌بار از دستور در حالت اول خود استفاده کرده‌است.

توجه کنید که به صورت پیش‌فرض (اگر کاربر از دستور بالا استفاده نکند)، وزن‌های manual غیر فعال هستند، و همچنین تمامی وزن‌های داده شده، باید اعداد حقیقی بوده و در بازه‌ی [1, 5] باشند. (در غیر این صورت خطای Bad Request چاپ شود.)

همچنین شما باید دستوری برای نمایش این وزن‌ها داشته باشید:

ورودی	خروجی
GET manual_weights	active true location <location> cleanliness <cleanliness> staff <staff> facilities <facilities> value_for_money <value_for_money> active false | Permission Denied

وزن‌های چاپ شده باید دقیقا ۲ رقم اعشار داشته باشند، حتی اگر ارقام بعد از اعشار صفر باشند. اگر عددی بیش از ۲ رقم اعشار داشت، باید ارقام اضافه دور ریخته شوند (truncate). (برای مثال عدد 2.1 باید 2.10 چاپ شود و عدد 2.146 باید 2.14 چاپ شود).

کار بعدی که باید انجام بدهید این است که خصیصه‌ی rating_personal را به لیست خصیصه‌هایی که دستور POST sort می‌تواند بگیرد اضافه کنید. وقتی این خصیصه برای sort استفاده شود، هتل ها باید برحسب امتیاز شخصی فرد به آن هتل مرتب شوند، که به شکل زیر محاسبه می‌شود:

۱- اگر فرد به آن هتل قبلا امتیاز داده است، امتیاز شخصی، همان rating_overall ای است که شخص به هتل داده.

۲- در غیر این صورت اگر وزن‌های manual فعال هستند، امتیاز شخصی، میانگین وزن‌دار امتیازات هتل در طبقه‌بندی های مختلف (که از فایل ratings.csv خوانده‌اید) با استفاده از وزن‌های manual است.به عنوان یادآوری، میانگین وزن‌دار داده‌های x_i با استفاده از وزن‌های w_i به شکل زیر حساب می‌شود:

$$m = \frac{\sum_i w_i \times x_i}{\sum_i w_i}$$

۳- در غیر این‌صورت (وزن‌های manual غیر فعال هستند)، اگر کاربر حداقل به ۵ هتل امتیاز داده باشد،شما مشابه بخش ۲ عمل می‌کنید، با این تفاوت که باید ابتدا وزن‌ها را تخمین بزنید و بعد از آن‌ها برای محاسبه‌ی میانگین وزن‌دار استفاده کنید. روش تخمین در بخش بعدی آمده است. اگر کاربر به کمتر از ۵ هتل امتیاز داده باشد، باید عبارت Insufficient Ratings را چاپ کنید.

تست کیس‌های نمونه‌ی این بخش در پوشه‌ی testcases/personal_rating هستند.

بخش دوم: تخمین وزن‌های شخصی

برای تخمین زدن نیاز است که کاربر به حداقل ۵ هتل امتیاز داده باشد. امتیازاتی که کاربر به هتل‌ها داده است را در نظر بگیرید. امتیازاتی که مربوط به دسته‌بندی ها هستند را x، و امتیازاتی که مربوط به سطح رضایت کلی (overall_rating) هستند را y می‌نامیم. اگر فرض کنیم فرد به n هتل امتیاز داده باشد، x^j امتیاز فرد به هتل j ام، و x_i^j امتیاز فرد به دسته‌بندی i ام هتل j ام است (ترتیب دسته‌بندی ها را مشابه آنچه در دستور GET manual_weights آمده است فرض کنید). y^j نیز سطح رضایت کلی فرد از هتل j ام است. برای مثال در تست کیس testcases/weight_estimation/1.in داریم:

$x^1 = [2.1, 1.6, 4.6, 2.3, 1.5]$, $y^1 = 2.58$

$x^2 = [4.2, 1.8, 2.9, 2.7, 4.9]$, $y^2 = 3.44$

$x_1^1 = 2.1,\ x_3^2 = 2.9$

حال سطح رضایت کلی را به شکل تابعی از امتیازهای مربوط به دسته‌بندی ها (x^j) تخمین می‌زنیم:

$$\widehat{y^j} = f(x^j,\ w) = \frac{\sum_{i=1}^{5} w_i \times x_i^j}{\sum_{i=1}^{5} w_i}$$

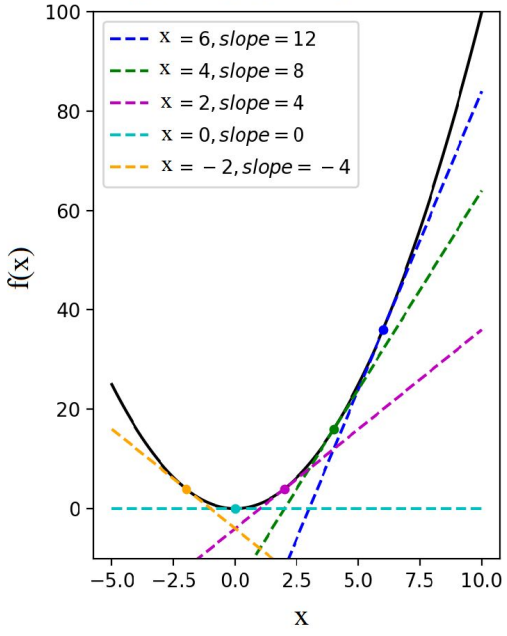
با در نظر گیری این مقدمات می‌توان گفت که هدف ما این است که w_i ها را طوری بیابیم که $\widehat{y^j}$ ها به y^j ها تا جای ممکن نزدیک شوند. یا به عبات دیگر، اگر تابع خطا[7] را به شکل زیر تعریف کنیم:

$$E\,(x^j,\ w,\ y^j)\ =\ (f(x^j,\ w)\ -\ y^j)^2$$

حال هدف ما یافتن w_i ها است به شکلی که تابع E به ازای تمام مقادیر j کمینه (صفر) شود.

اما چطور می‌توان این کار را انجام داد ؟ شما باید از روش بیش‌ترین شیب[8] برای حل این مسئله استفاده خواهیم کرد. این روش از یک نقطه‌ی تصادفی کار خود را شروع می‌کند، یعنی تمامی w_i ها یک مقدار تصادفی (در بازه‌ی [1, 5]) خواهند داشت. سپس، در هر قدم، از نقطه‌ی فعلی در جهتی که بیش‌ترین کاهش در E را ایجاد می‌کند یک گام برمیدارد. برای این کار کافیست که مشتق جزئی E را نسبت به هر یک از وزن‌ها حساب کنیم و هر یک از وزن‌ها را در جهت خلاف این مشتق حرکت بدهیم. برای درک چرائی این موضوع می‌توانید شکل ۱ را ببینید. در این شکل دیده می‌شود که وقتی در سمت راست نقطه‌ی کمینه (صفر) هستیم، علامت مشتق مثبت بوده، و وقتی در سمت چپ آن هستیم، علامت مشتق منفی است. پس برای رسیدن به کمینه کافی است در عکس جهت مشتق حرکت کنیم.

شکل ۱. خطوط مماس بر $f(x) = x^2$



برای محاسبه‌ی مشتق، نیازی نیست از فرمول بسته‌ی آن برای تابع E استفاده کنید، بلکه کافیست آن را با استفاده از فرمول زیر تقریب بزنید:

$$\frac{\partial E}{\partial w}(x,\ w_0,\ y)\ \simeq\ \frac{E(x,\ w_0+\varepsilon,\ y)-E(x,\ w_0,\ y)}{\varepsilon}$$

[7] error function

[8] steepest descent - این روش را به نام gradient descent نیز می‌شناسند.

که در آن g یک تابع دلخواه بوده و ε مقداری بسیار کوچک است . توصیه می‌شود مقدار ε را حدود 10^{-4} یا کمتر بگذارید. پس از تقریب مشتق، به این شکل وزن‌ها را به روز رسانی می‌کنیم:

$$w_i = w_i - \alpha \times d_i \qquad for\ i = 1, ..., 5$$

که تعریف d_i به شکل زیر است:

$$d_i = \sum_{j=1}^{n} \frac{\partial E(x^j, w, y^j)}{\partial w_i}$$

و α در واقع همان step size یا چیزی است که اندازه‌ی هر قدم را مشخص می‌کند، و در این مساله می‌توانید آن را ۱ فرض کنید. دقت کنید که هر پنج w_i موجود باید همزمان به روز رسانی شوند. در واقع باید ابتدا هر پنج d_i را حساب کرده، و بعد w_i ها به روزرسانی کنید. در نهایت نیز باید مطمئن شوید که w_i های حاصل در بازه‌ی [1., 5.] باشند، و اگر این شرط برقرار نبود، آن را با 5 یا 1 جایگزین کنید (هرکدام که نزدیک تر بود).
این حلقه‌ی محاسبه‌ی مشتق و به روز رسانی وزن‌ها را باید تعداد دفعات زیادی تکرار کنید تا به نتیجه‌ی مطلوب برسید.

توضیحات این بخش را می‌توان در قالب شبه کد زیر خلاصه کرد:

```
for i from 1 to 5:
    w[i] = random uniform number between 1.0 and 5.0

for k from 1 to 1000:
     for i from 1 to 5:
          d[i] = 0

    for j from 1 to n_ratings:
      for i from 1 to 5:
           d[i] += ∂E(x^j, w, y^j)/∂w_i

    for i from 0 to 4:
      w[i] = w[i] - alpha * d[i]
      w[i] = clamp(w[i], 1, 5)
```

توجه کنید که تنها تست اتوماتیک از این بخش، ترتیب هتل‌های نمایش داده شده پس از یادگیری وزن‌ها است، و امتیاز این هتل‌ها به قدری متفاوت خواهد بود که تفاوت‌های جزئی در وزن‌های نهایی مشکل ایجاد نکند. اما با این حال، مقادیر درست وزن‌ها برای تست کیس اول این بخش (testcases/weight_estimation/1.in) برای کمک به debug برنامه در زیر آمده است:

[location: 4.9, cleanliness: 1.2, staff: 3.4, facilities: 4.1, value_for_money: 2.1]

اگر وزن‌هایی که به دست می‌آورید تفاوتشان با این وزن‌ها از 0.05 کمتر باشد، به احتمال بسیار زیادی پیاده‌سازی تان درست بوده است. البته توجه کنید که وزن‌های یادگرفته شده نسبت به scaling بی تفاوت هستند، یعنی اگر همه‌ی وزن‌های صحیح را در یک عدد ثابت ضرب بکنیم، مجموعه‌ی حاصل همچنان جواب صحیحی برای مسئله خواهد بود، و قبل از مقایسه‌ی وزن‌ها باید این مساله را در نظر بگیرید.

توجه داشته باشید که تخمین وزن‌ها نسبتا کار وقت گیری است، و در نتیجه بهتر است مطمئن باشید که این کار را به کمترین تعداد دفعات ممکن انجام بدهید، یعنی اگر وزن‌هایی که قبلا برای یک کاربر یادگرفته اید معتبر هستند، مجدد آن ها را حساب نکنید.

همچنین طبیعی است که بخواهیم وزن‌های یادگیری شده را از سیستم بخوانیم. در نتیجه باید دستور زیر را نیز پیاده سازی کنید:

ورودی	خروجی
`location <location> cleanliness <cleanliness> staff <staff> facilities <facilities> value_for_money <value_for_money>` | Permission Denied | Insufficient Ratings	`GET estimated_weights`

این دستور وزن‌های هر دسته‌بندی را برای کاربر فعلی محاسبه کرده و بعد چاپ می‌کند. در چاپ اعداد اعشاری این بخش نیز باید به نکاتی که در بخش GET manual_weights در مورد چاپ اعداد اعشاری گفته شد توجه کنید.

تست کیس‌های این بخش در پوشه‌ی testcases/weight_estimation قرار دارند.

نکات پایانی

- توجه کنید که بخش‌های این فاز، از ساده شروع شده و به مرور سخت تر می‌شوند. به طور خاص، بخش آخر (تخمین وزن‌های شخصی) احتمالا به مقدار قابل توجهی از بخش‌های قبلی وقت گیر تر خواهد بود. در هنگام برنامه‌ریزی، به این نکته توجه داشته باشید!
- در مورد بخش آخر (تخمین وزن‌های شخصی)، دقت کنید که رفع خطا از برنامه‌هایی که محاسبات عددی زیادی انجام می‌دهند، دشوار است، به خصوص اگر خطای آن از جنسی باشد که باعث شود بخشی از برنامه خروجی «نادقیق» تولید کند. چرا که برای فهمیدن اینکه خروجی نادقیق است باید محاسبات عددی مربوطه را مجدد انجام داد و با یک نگاه کردن نمی‌توان خطا را در آن فهمید. در نتیجه، برای این بخش هر component ای که می‌نویسید را به خوبی و با ورودی های متفاوت تست کنید تا از درستی کارکردشان مطمئن شوید. پیدا کردن component های خراب بعد از اینکه همه در کنار هم قرار گرفتند بسیار سخت تر خواهد بود.

- از آن جایی که در فاز بعدی رابط کاربری شما دیگر فقط خط فرمان نخواهد بود، بهتر است طراحی برنامه‌ی شما طوری باشد که اولا مرز بین منطق برنامه و رابط کاربری کاملا مشخص باشد، و ثانیا منطق برنامه هیچ وابستگی به رابط کاربری نداشته باشد تا بتوان به راحتی آن را عوض کرد.
- می‌توانید از کتابخانه‌ی <random> برای تولید اعداد تصادفی استفاده کنید.
- طراحی خوب، رعایت سبک برنامه نویسی درست و تمیز بودن کد برنامه‌ی شما در نمره‌ی تمرین تأثیر زیادی دارد.
- تمام فایل‌های خود را در قالب یک **پرونده‌ی زیپ** با نام A7-<SID>.zip در صفحهٔ CECM درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۸۹۹۹ است، نام پروندهٔ شما باید A7-810198999.zip باشد.
- **دقت کنید** که پرونده زیپ آپلودی شما باید پس از Unzip شدن شامل پرونده‌های پروژه شما (از جمله Makefile) باشد و از زیپ کردن پوشه‌ای که داخل آن فایل‌های پروژه‌تان قرار دارد خودداری فرمایید.
- برنامهٔ شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++11 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- **دقت کنید** که پروژه شما باید Multi-file باشد و Makefile داشته باشد. همین‌طور در Makefile خود مشخص کنید که از استاندارد c++11 استفاده می‌کنید.
- درستی برنامه‌ی شما از طریق **آزمون‌های خودکار** سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- **دقت کنید** که نام پرونده‌ی اجرایی شما باید **utrip.out** باشد.
- سوالات خود را تا حد ممکن در فروم درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن‌ها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این فاز پروژه ارتباط برقرار کنید. توجه داشته باشید که دیگر شبکه‌های اجتماعی مانند تلگرام راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی موظف به پاسخگویی در محیط‌های غیررسمی نیستند.