



ساختمان های داده

دکتر سید ابوالقاسم میرروشندل

طراح: امیرحسین مهدی نژاد

پروژه ی نهایی

هدف پروژه:

هدف از این پروژه، تشخیص ساختارهای اجتماعی در گراف و پیاده سازی الگوریتم های کاربردی مرتبط با گراف در شبکه های اجتماعی است. محاسبه ی تأثیر الگوریتم های متفاوت در زمان و حافظه ی مصرفی و مقایسه ی نتایج آزمایش ها، دستاورد نهایی این پروژه خواهد بود.

مهم ترین بخش این پروژه، پیاده سازی ساختمان داده ها و همچنین الگوریتم های مربوط به مسئله است که در فاز دوم پروژه، دانشجو باید در قالب پیاده سازی یک اپلیکیشن مارکت مانند Café Bazaar یا تعریف سناریوی جدید خود، کاربرد آن الگوریتم ها را به تصویر بکشد.

تعاریف:

- گراف همبند: گرافی است که هر دو رأس دلخواه از آن به یکدیگر مسیر داشته باشند.
- رأس برشی: رأسی از گراف است که حذف آن باعث افزایش تعداد مؤلفه های همبندی گراف می شود. رأس برشی در شبکه های کامپیوتری اهمیت ویژه ای دارد.
- یال برشی: مشابه رأس برشی است، به طوری که حذف آن باعث افزایش تعداد مؤلفه های همبندی گراف می شود.
- اجتماع: زیرگرافی که تعداد یال های بین رؤوس داخلی آن، نسبت به یال های خارج شده از زیرگراف (دیگر رؤوس موجود در گراف) بیشتر باشد.

شرح الگوریتم اصلی:

در این الگوریتم، با حذف یال های بین اجتماع ها، گراف به زیرگراف های متراکم تر تقسیم می شود. معیاری که این الگوریتم برای امتیازدهی به یال ها استفاده می کند، عبارت است از:

$$c_{ij} = \frac{z_{ij} + 1}{\min[k_i - 1, k_j - 1]}$$

که در آن z_{ij} برابر با **تعداد دور های ساده به طول سه** است که یال بین رأس i و j در آن دور دیده می شود. بدیهی است که اگر یالی بین دو رأس i و j وجود نداشته باشد، مقدار c_{ij} برای آن ها تعریف نمی شود. همچنین k_i ، درجه رأس i و k_j ، درجه رأس j است. بنابراین، اگر درجه ی یکی از رأس ها عدد یک باشد، آنگاه مخرج کسر برابر با صفر می شود که تعریف نشده است؛ یعنی در این حالت، مقدار c_{ij} را باید برابر با عددی بسیار بزرگ در نظر بگیرید.

قدم های الگوریتم به این صورت است:

۱. محاسبه ی امتیاز c_{ij} برای تمام یال های گراف
۲. مرتب سازی صعودی یال ها بر اساس امتیاز محاسبه شده
۳. حذف یال با کوچکترین مقدار c_{ij} از گراف
۴. اگر گراف به **دو بخش** تقسیم شده است، آنگاه پایان الگوریتم
۵. برو به قدم اول

در ورودی برنامه، هر رأس یا یک عدد طبیعی نشان داده می شود و بیانگر گراف ورودی است. هر سطر از ورودی، شامل یک جفت رأس (بیانگر یک یال بدون جهت) است. پس از تعیین فایل ورودی، کاربر باید قادر به دیدن نتایج، شامل **زمان اجرای الگوریتم (پنج قدم ذکر شده)** و همچنین **حافظه ی مصرفی توسط الگوریتم (با توجه به ساختمان داده ی مورد استفاده)** باشد و نهایتاً فایلی حاوی چنین اطلاعاتی ساخته شود:

#VertexNumber : A or B

یعنی در هر سطر از فایل نتیجه، شماره اندیس رأس و اجتماعی که در آن قرار دارد مشخص شود. پیشفرض این است که رأس شماره ۱ همواره در اجتماع A باشد. یعنی سطر اول فایل خروجی به این شکل است:

#1 : A

توجه کنید تنها پیاده سازی دو اجتماع A و B کافیهست، اما پیاده سازی اجتماع های بیشتر **نمره ی مثبت** دارد. (بدیهتاً برای این کار، باید قدم چهارم الگوریتم را نیز تغییر دهید)

برای ذخیره سازی گراف در حافظه، باید از هر دو ساختمان داده ی **لیست مجاورت** و **ماتریس مجاورت** استفاده کنید. جهت مرتب سازی نیز باید از الگوریتم های **Quick Sort** و **Insertion sort** و **Merge Sort** استفاده شده و پس از پیاده سازی هر بخش، نتایج زمانی بدست آمده در نمودارهای ترسیمی با یکدیگر مقایسه شود. (هم برای حافظه و هم برای پیچیدگی زمانی)

جزئیات پیاده سازی:


توجه کنید اگر هر سناریوی دیگری به جز اپلیکیشن مارکت انتخاب می کنید (مثلاً شبکه ی اجتماعی)، باید با دوستان حل تمرین مشورت کنید. به فرض اینکه تمام اپلیکیشن های ما "بازی" باشند، ویژگی های مورد نظر به این صورت است:

۱. کلاس اپلیکیشن:

شامل یک لیست static از تمامی بازی های موجود در مارکت
شامل نام بازی، نام برنامه نویس یا کمپانی، تعداد دانلودها و از همه مهم تر امتیاز بازی در هریک از بخش های "Strategy", "Sports", "Simulation", "Arcade" و... که البته پیاده سازی ۲ دسته بندی کافیهست. رتبه ی بازی در هر دسته، عددی بین ۰ تا ۵ خواهد بود (میانگین امتیازاتی که کاربران می دهند) و تمامی بازی ها باید در تمامی دسته ها امتیاز داشته باشند.
توجه کنید اضافه کردن ویژگی های بیشتر به این کلاس (در جهت بهتر شدن ساختار مارکت) ضمن پیچیده تر کردن نمودارها و الگوریتم ها و ساختمان داده های مورد استفاده، **نمره ی مثبت** نیز خواهد داشت.

۲. کلاس کاربر:

شامل یک لیست static از تمامی کاربرهای مارکت
شامل نام کاربر و اپلیکیشن های دانلود شده توسط کاربر
توجه کنید پیاده سازی آفلاین تمامی اجزای گفته شده کافیهست، اما نظر به اینکه کاربرد گراف در شبکه های اجتماعی بیشتر قابل درک باشد، پیاده سازی قابلیت های گفته شده تحت شبکه ی داخلی، **نمره ی مثبت** خواهد داشت.
صفحه ی گرافیکی هر اپلیکیشن باید شامل موارد زیر باشد:

نام اپلیکیشن: بازی آنلاین یوزپلنگ ایرانی نام برنامه نویس: امیرحسین مهدی نژاد	
امتیازات کسب شده (از نظرات ۱ کاربر): Strategy: 2 🌟🌟 Sports: 1 🌟 Simulation: 5 🌟🌟🌟🌟🌟	
تعداد دانلودها: ۱	دانلود کنید
اپلیکیشن های مرتبط: بازی آنلاین پلنگ مازندران	

سیستم Recommender یا پیشنهاددهنده ی اپلیکیشن های مرتبط باید مبتنی بر یکی از دو ساختار زیر باشد:

۱. استفاده از LCS رشته ی اسامی اپلیکیشن ها

۲. استفاده از Euclidean distance score بین امتیازات اپلیکیشن ها:

$$d(app_i, app_j) = \sqrt{\Delta x^2 + \Delta y^2}$$

که درجه ی شباهت از رابطه ی $\frac{1}{1+d(app_i, app_j)}$ بدست می آید. توجه کنید که این رابطه تنها برای ۲ دسته بندی x و y درست است و اگر تعداد دسته بندی های امتیازدهی اپلیکیشن ها بیشتر شود، باید تغییرات لازم را در فرمول لحاظ کنید.

نهایتاً ساختار های اجتماعی گراف بدست آمده (از اتصال اپلیکیشن های مرتبط به یکدیگر)، باید توسط الگوریتم شرح داده شده در بخش اول پروژه، تحلیل شود.

نکات تکمیلی:

- در الگوریتم اصلی، هرگاه به دو یال با شرایط یکسان رسیدید، اولویت با یال برشی است و پس از آن با یالی است که یک سر آن رأسی برشی وجود داشته باشد.
- پیاده سازی بخش اول پروژه (الگوریتم اصلی برای تشخیص اجتماع ها) به صورت جداگانه الزامی است و هیچ توجیهی در غیر قابل تست بودن آن وجود ندارد. (بدین معنا که اگر برنامه ی شما فقط فاز اول پروژه را شامل شود، این بخش باید به تنهایی قابل تست باشد و خروجی مناسب را تولید کند حتی اگر مارکت را پیاده سازی نکرده باشید)
- یکی از سناریوهای مناسب دیگری که قابلیت پیاده سازی به جای مارکت را دارد، شبکه ای مانند linkedin است. (در اینجا به جای امتیازدهی به دسته بندی های اپلیکیشن، به توانایی های یک فرد در زمینه های مختلف امتیاز می دهیم)
- قبل از انتخاب هر سناریوی متفاوت با دو سناریوی پیشنهاد داده شده (linkedin, café bazaar)، با دوستان حل تمرین مشورت کنید.
- قابلیت مشاهده ی سطح ارتباط بین دو رأس (بدست آمده از جستجوی اول سطح در گراف مربوطه)، **نمره ی مثبت** خواهد داشت.
- پیاده سازی به صورت تک نفره است و هیچ محدودیتی برای زبان برنامه نویسی وجود ندارد.
- همه ی ساختمان داده های مورد نیاز، باید با توجه به تعریف پروژه پیاده سازی شوند. استفاده از ساختمان داده های آماده مجاز نیست.
- بحث و بررسی میان دانشجویان آزاد است اما هر دانشجو موظف است پروژه را به تنهایی انجام دهد و در هنگام تحویل حضوری، به تمام جزئیات کد کاملاً مسلط باشد. با موارد تقلب و کپی کردن، طبق تشخیص دوستان حل تمرین، برخورد جدی خواهد شد.
- توجه کنید که کدهای شما باید خوانا و دارای کامنت گذاری مناسب باشد.
- در کلاس کوئرای **Data Structures 971** با رمز **guilan96** ثبت نام کرده، نام و شماره ی دانشجویی خود را به درستی وارد کنید. عواقب بی دقتی در این مورد، به عهده ی دانشجو است.

- برای پرسش و پاسخ درباره ی پروژه، فقط از سامانه ی کوئرای درس مربوطه، اقدام کنید.
- پوشه ی مربوط به کد پروژه را (در صورت نیاز همراه با فایل pdf شرح انجام پروژه، نحوه ی اجرای برنامه، گزارش مربوط به تحلیل ساختمان داده ها و محاسبات انجام شده) در قالب یک فایل zip و در بخش مربوطه بارگذاری کنید. عواقب بی دقتی در این مورد، به عهده ی دانشجو است.
- زمان بندی و چگونگی تحویل حضوری پروژه، متعاقباً اعلام خواهد شد.