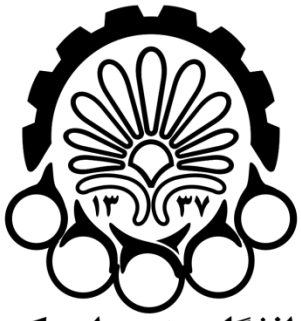


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

طراحی الگوریتم

تمرین اول

بهمن‌ماه ۱۳۹۴

- **پیچیدگی زمانی**

در این بخش قصد داریم پیچیدگی زمانی چهار مورد از الگوریتم‌های مطرح برای مرتب‌سازی لیستی از اعداد را که در درس ساختمان داده با آن‌ها آشنا شدید، مورد بررسی و مقایسه قرار دهیم. برای این منظور، کافیست الگوریتم‌های زیر را بر روی مجموعه داده‌های فراهم شده، که در صفحه اینترنتی درس در دسترس است، اجرا نموده و زمان‌های اجرا را در یک نمودار به ازای تعداد ورودی مختلف بیاوریم.
نمودار فوق، تابع زمان اجرای الگوریتم بر حسب تعداد ورودی خواهد بود که رفتار مجانبی آن، مشخص-کننده پیچیدگی زمانی الگوریتم می‌باشد. گزارشی تهیه نمایید از نحوه رفتار زمانی الگوریتم‌های زیر:

- Insertion sort
- Quick sort
- Heap sort
- Merge sort

در این گزارش، نمودار زمانی الگوریتم‌های فوق را در یک دیاگرام فراهم نموده و رفتار زمانی الگوریتم‌ها را مورد بررسی و مقایسه قرار دهید.

توجه نمایید که نمودار فوق را می‌توانید با استفاده از نرم‌افزارهای مختلف بکشید و نیازی به پیاده‌سازی نرم‌افزاری نمی‌باشد. به علاوه، اجرای برنامه بر روی تمام الگوریتم‌ها الزامی نیست؛ اما استفاده از مجموعه‌های داده‌ای باید در حدی انجام گیرد که به شکل نمودارهای زمانی خللی وارد نشود.

- **پیمایش گراف**

۱. ارسال درخواست در شبکه های نظیر به نظیر[۱]

امروزه در بسترهای اینترنتی، اشتراک گذاری فایل‌ها و اطلاعات مختلف بسیار حائز اهمیت است. نحوه اتصال سیستم‌های کامپیوتری به یکدیگر نقش بسزایی در بالابردن سرعت انتقال اطلاعات بین کاربران ایفا می‌کند. دو معماری مختلف مطرح در این زمینه به نام‌های client-server و نظیر به نظیر به طور گسترده‌ای مورد استفاده قرار می‌گیرند و نرم‌افزارهای معروفی مثل Bit Torrent برای بالابردن سرعت ارسال اطلاعات و کاهش حجم درخواست‌ها روی سرور از معماری نظیر به نظیر استفاده می‌کنند.

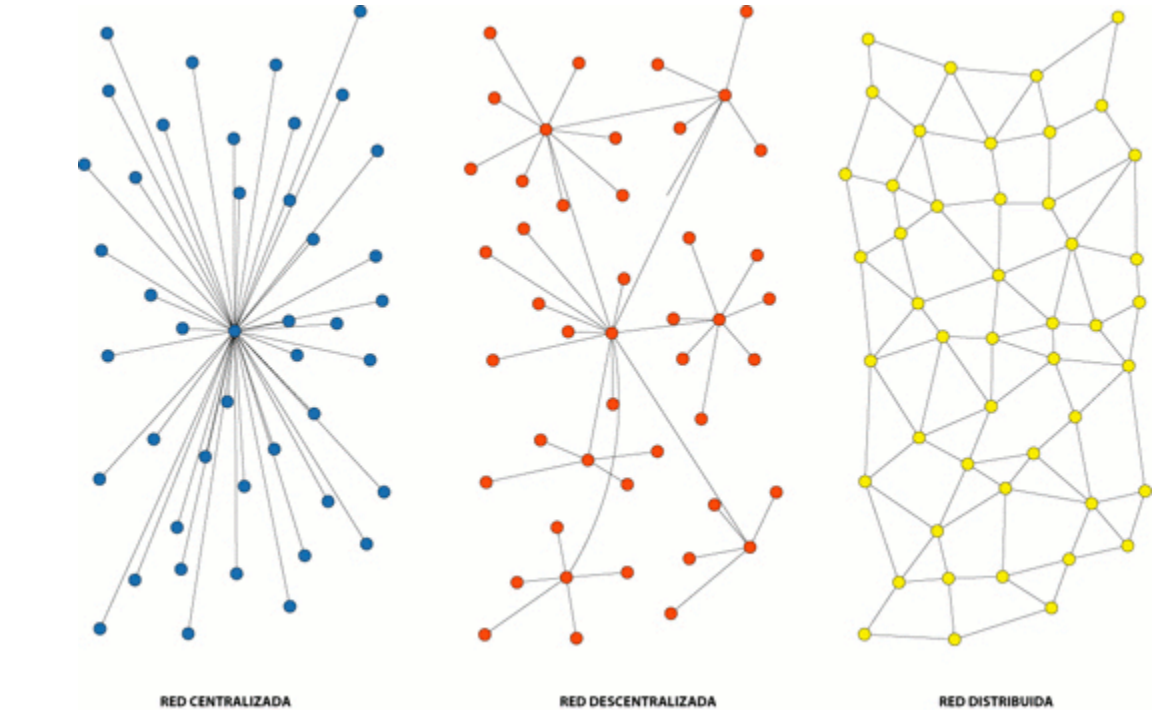
در این معماری تمام کاربران می‌توانند به هر شکل دلخواه به هر کاربر دیگر متصل شده و بخش‌های مختلفی از فایل درخواستی خود را از کاربران مختلف دریافت نمایند. شکل۱ نمایی از نحوه اتصال کاربران در یک شبکه نظیر به نظیر را به همراه معماری کلاینت-سرور مشخص می‌نماید. شکل سمت

---

[۱] Peer to Peer (P۲P)

چپ یک معماری کلاینت-سرور و شکل‌های دیگر معماری‌های مختلفی از شبکه‌های نظیر به نظیر را مشخص می‌نمایند.

همان‌طور که در شکل مشخص است، در شبکه‌های نظیر به نظیر هر نوع اتصالی بین هر کاربر دلخواه می‌تواند وجود داشته باشد. یکی از مسائل مهم و پرکاربرد در این شبکه‌ها پیدا کردن گروه‌های متصل کاربران به یکدیگر می‌باشد.



شکل ۱ توپولوژی‌های نظیر به نظیر و کلاینت-سرور. شکل سمت چپ نمایش‌دهنده نحوه اتصالات در روش کلاینت-سرور است. یک گره مرکزی در این روش به عنوان سرور انتخاب شده و تمام گره‌های دیگر که همان کاربران هستند، فایل و اطلاعات مورد نیاز خود را مستقیما از سرور دریافت می‌کنند. در شکل وسط و سمت راست، نحوه اتصالات به شکل نظیر به نظیر مشخص شده است. در این شکل کاربران می‌توانند به طور دلخواهی به یکدیگر متصل شده و بخش‌های مختلف فایل و اطلاعات خود را از کاربران مختلف دریافت نمایند.

فرض کنید شما در یک شبکه نظیر به نظیر هستید و می‌خواهید تبلیغات محصولات خود را به کاربران دیگر ارسال نمایید. در چنین مواردی لازم است گروه کاربرانی را که به شما متصل هستند بیابید تا بتوانید میزان تاثیر تبلیغات خود را بررسی نمایید.

برای چنین منظوری نیاز به داشتن برنامه ای داریم که گراف اتصالات کاربران در یک شبکه را، که به شکل ماتریس مجاورت داده شده است، دریافت نموده و مولفه‌های هم‌بند موجود در گراف را مشخص نماید. برای این کار می‌توان از الگوریتم جستجوی اول سطح[۲] روی گراف استفاده نمود.

---

۲ BFS

توجه داشته باشید خروجی برنامه شما باید شامل تعداد مولفه‌های هم‌بند موجود در گراف، لیست کاربران موجود در هر مولفه و شماره بهترین گروه کاربران برای ارسال تبلیغات باشد. بهترین گروه برای تبلیغات بزرگترین مولفه هم‌بند گراف می‌باشد.

۲. تشخیص دور منطقی

همه ما با استدلال های علت و معلولی آشنا هستیم. این استدلالات معمولا بر پایه روابط بین علت و معلول شکل می‌گیرند. هر ارتباط منطقی به این شکل را می‌توان با قراردادن یک فلش از علت به سمت معلول نمایش داد؛ برای مثال برای بیان رابطه بین گرم‌شدن اتاق و روشن بودن آتش از آنجاکه علت گرم شدن اتاق، روشن بودن آتش است، می‌توان به از نماد زیر استفاده کرد:

$$a \rightarrow b$$

که در آن a، گزاره روشن بودن آتش و b، گزاره گرم شدن اتاق می‌باشد.

یکی از چالش‌های بزرگی که همواره در مسیر استدلالات فلسفی گریبان‌گیر فیلسوفان بوده و هست پیداکردن دورهای منطقی در استدلالات خود است. این مساله به خصوص وقتی تعداد ارتباطات علّی مورد استفاده در استدلال بسیار زیاد باشد، بسیار دشوارتر خواهد بود.

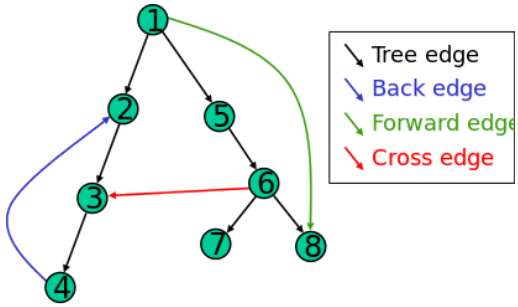
در این بخش قصد داریم برنامه‌ای بنویسیم که به فیلسوفان در اعتبارسنجی یک استدلال به لحاظ عدم وجود دور منطقی در روابط مورد استفاده کمک کند. برنامه‌ها و ابزارهای از این دست، یکی از ابزارهای مهم در اعتبارسنجی کدهای نوشته شده توسط برنامه‌نویسان در زبان‌های مبتنی بر منطق مثل prolog می‌باشند که می‌توانید درباره آن‌ها تحقیق و مطالعه نمایید.

در این بخش قصد داریم با استفاده از الگوریتم‌های مختلف مطرح شده در درس ساختمان داده برای تشخیص وجود دور در یک گراف استفاده نماییم. همان‌طور که می‌دانیم می‌توان یال‌های یک گراف را بسته به مسیر حرکتمان از یک گره به همسایه‌ها به شکل زیر نام‌گذاری نمود.

همان‌طور که در شکل۲ آمده است، یال‌ها را می‌توان به ۴ دسته مختلف تقسیم نمود که توضیحات آن-ها را می‌توانید در منابع درسی درس ساختمان داده بیابید.

برای نوشتن برنامه مذکور، کافیست برنامه‌ای بنویسیم که با گرفتن یک گراف به شکل ماتریس مجاورت و اجرای الگوریتم DFS روی آن، با تشخیص اولین Back Edge، کار پردازش را تمام کرده و استدلال

ارائه شده را دارای دور بداند. در صورتی که تمام گره‌های گراف مشاهده شدند و هیچ یالی از نوع مذکور به چشم نخورد، استدلال ارائه شده بدون دور و معتبر خواهد بود.



شکل ۲ نمایش دسته‌های مختلف یال در یک گراف

توجه داشته باشید، خروجی برنامه شما باید شامل معتبر بودن یا نبودن الگوریتم بوده و همین‌طور استدلالی را که موجب ایجاد دور شده است نمایش دهد.

موفق و پیروز باشید.