پیشنهادهایی برای پروژه‌های کارشناسی

در این مستند برخی از موضوعات پیشنهادی برای پروژه‌ی کارشناسی فهرست شده‌اند. بیشتر این موضوعات در دو دسته‌ی کلی الگوریتم‌ها و ساختمان‌های داده و برنامه‌های سیستمی قرار می‌گیرند. دانشجویانی که علاقمند به انجام یکی از این موضوعات در پروژه‌ی کارشناسی خود هستند، برای توضیحات بیشتر با gholamirudi@nit.ac.ir تماس بگیرند. لازم به اشاره است که این فهرست به روز می‌شود و با گذشت زمان پیشنهادهای بیشتری به آن اضافه می‌شوند.

نوشتن یک بسته‌ی نیتراف برای پایان‌نامه‌های فارسی

تیراف (Troff) یکی از قدیمی‌ترین و همین‌طور قدرتمندترین ابزارهای حروفچینی (Typesetting) است که در کنار سیستم عامل یونیکس طراحی و نوشته شده است. مشابه یونیکس که تأثیر چشم‌گیری بر سیستم عامل‌های بعد از خود گذاشته است، ایده‌های بسیار جالبی در تیراف و معماری آن معرفی شده‌اند که با وجود گذشت بیشتر از چهار دهه از تولد آن، ساختار، انعطاف و خروجی این ابزار در میان ابزارهای حرفچینی مشابه بسیار شاخص است. نیتراف (Neatroff) یک پیاده‌سازی جدید از تیراف است که امکانات لازم برای حروفچینی متن فارسی را پشتیبانی می‌کند.

هدف اصلی این پروژه، نوشتن یک بسته‌ی نیتراف برای حروفچینی پایان‌نامه‌های کارشناسی به زبان فارسی است. در بسته‌های حروفچینی، ساختار و شکل مستندها با تعریف ماکروهایی مشخص می‌شوند؛ نوشتن این بسته‌ها نیاز به مهارت در استفاده از تیراف و دستورات آن دارد. از این رو در قسمت اول این پروژه، دستورات و شیوه‌ی استفاده از تیراف مستند می‌گردند و مهارت‌های لازم برای استفاده از تیراف کسب می‌شوند.

گام‌های اصلی پروژه:

۱ مستندسازی دستورات و شیوه‌ی استفاده از تیراف

۲ طراحی بسته‌ای برای پایان‌نامه‌های کارشناسی

۳ مستندسازی بسته‌ی معرفی شده

اطلاعات بیشتر:

معرفی تیراف	http://www.troff.org/54.pdf
تفاوت‌های نیتراف نسبت به تیراف	http://litcave.rudi.ir/neatroff.pdf
معرفی نیتراف به زبان فارسی	http://litcave.rudi.ir/neatfarsi.pdf

برچسب‌گذاری فاصله در گراف‌ها

در بسیاری از کاربردهای مبتنی بر گراف لازم است فاصله‌ی هر دو رأس از گراف محاسبه شود. در صورتی که وزن هر یال حداکثر w و n تعداد رأس‌های گراف باشد، فاصله‌ی دو رأس حداکثر $w(n-1)$ خواهد بود و نگهداری آن به $O(\log(wn))$ بیت احتیاج خواهد داشت. بنابراین برای نگهداری فاصله‌ی هر رأس از هر رأس دیگر $O(n^2 \log(wn))$ بیت لازم است (برای گراف‌های غیر وزن‌دار w برابر یک است و نگهداری همه‌ی فاصله‌ها به $O(n^2 \log n)$ بیت احتیاج دارد). در صورتی تعداد رأس‌های گراف بسیار زیاد باشد، گاهی اختصاص این مقدار حافظه برای نگهداری فاصله‌ی هر دو رأس امکان ندارد.

یک راه برای کاهش این مقدار حافظه، اختصاص برچسب‌هایی به رأس‌ها است (Graph distance labeling) که با داشتن فقط برچسب هر دو رأس بتوان فاصله‌ی آن دو رأس را محاسبه کرد (برچسب‌گذاری فاصله مزیت‌های دیگری نیز، مخصوصا هنگامی که گراف توزیع شده باشد، دارد). اگر برچسب اختصاص داده شده به رأس u با $l(u)$ نمایش داده شود، الگوریتم محاسبه‌ی فاصله با گرفتن برچسب‌های $l(u)$ و $l(v)$ می‌تواند فاصله‌ی دو رأس u و v را محاسبه کند. برای ارزیابی روش‌های مختلف برچسب‌گذاری فاصله گراف، دو مسئله اهمیت زیادی دارند: طول برچسب و پیچیدگی زمانی الگوریتمی که با گرفتن برچسب دو رأس، فاصله‌ی آنها را محاسبه می‌کند. در ساده‌ترین حالت، برچسب یک رأس می‌تواند فاصله‌ی آن رأس تا هر رأس دیگر باشد که در آن صورت طول هر برچسب $O(n \log(nw))$ خواهد بود و فاصله‌ی دو رأس با توجه به برچسب آنها با پیچیدگی زمانی $O(1)$ قابل محاسبه خواهد بود. اما این برچسب‌ها را می‌توان با الگوریتم‌هایی بهبود داد. در این پروژه برخی از این الگوریتم‌ها مطالعه، پیاده‌سازی و عملکرد آنها روی گراف‌های بزرگ ارزیابی می‌شوند.

گام‌های اصلی پروژه:

۱ مطالعه‌ی چند روش برچسب‌گذاری فاصله در گراف‌ها

۲ پیاده‌سازی برخی از روش‌های مطالعه شده

۳ ارزیابی روش‌های پیاده‌سازی شده

اطلاعات بیشتر:

یکی از روش‌های برچسب‌گذاری فاصله http://arxiv.org/pdf/1504.04498v1

پیش-پردازشگرهای نیتراف

معماری تیراف انعطاف زیادی برای گسترش این ابزار حروفچینی ارائه می‌دهد. اضافه کردن یک پیش-پردازشگر (Preprocessor) جدید یا نوشتن تعدادی ماکرو برای تیراف یا پیش‌پردازشگرهای آن، دو راه برای انطباق تیراف برای کاربردهای جدید می‌باشد. برای مثال، پیش-پردازشگر pic، با استفاده از دستورات سطح پایین تیراف، امکان کشیدن شکل را در تیراف فراهم می‌سازد.

یکی از کاربردهای ممکن برای ابزارهای تولید مستند، کشیدن مدارهای الکترونیکی می‌باشد. در این پروژه، یک پیش-پردازشگر برای کشیدن مدارهای الکترونیکی پیاده‌سازی می‌شود. در پروژه‌ی مشابهی می‌توان پیش-پردازشگری نوشت که انواع مختلف نمودارها را بکشد.

گام‌های اصلی پروژه:

۱ معرفی پیش-پردازشگرهای مرتبط

۲ پیاده‌سازی پیش‌پردازشگر برای کشیدن مدار

۳ مستندسازی پیش-پردازشگر

اطلاعات بیشتر:

معرفی پیش-پردازشگر pic http://plan9.bell-labs.com/10thEdMan/pic.pdf

برخی از پیش‌پردازشگرهای تیراف http://troff.org/macros.html

محاسبه‌ی فراگیرنده

پردازش گراف‌های بسیار بزرگ به منابع زیاد و گاهی غیر قابل دسترس احتیاج دارد. برای مثال اگر گرافی با n رأس و m یال چند میلیون رأس داشته باشد، تخصیص $O(n^2)$ کلمه‌ی حافظه یا اجرای یک الگوریتم با پیچیدگی زمانی $O(n^2)$ با کامپیوترهای رایج غیر ممکن یا بسیار کند است. اما در صورتی که تعداد یال‌های گراف ورودی کم باشد، الگوریتم‌هایی که پیچیدگی زمانی یا حافظه‌ی آنها $O(m)$ باشد، به راحتی قابل اجرای خواهند بود. از این رو، یکی از راه‌هایی که برای پردازش گراف‌های بسیار بزرگ به کار گرفته می‌شود، حذف تعدادی از یال‌های این گراف‌ها است تا پردازش آن سریع‌تر گردد و از سوی دیگر ویژگی‌های مورد نظر در گراف چندان تغییر نکنند. در صورتی که ویژگی مورد نظر فاصله‌ی رأس‌ها از یکدیگر باشد، گراف حاصل یک فراگیرنده (Spanner) نامیده می‌شود.

یک فراگیرنده H از گراف G دارای کشش (α, β) است اگر به ازای هر دو رأس مثل u و v شرط $d_G(u,v) \leq d_H(u,v) \leq d_G(u,v) \times \alpha + \beta$ برقرار باشد ($d_G(u,v)$ فاصله‌ی رأس‌های u و v در گراف G است). در این پروژه برخی الگوریتم‌های انتخاب فرگیرنده از یک گراف مطالعه، پیاده‌سازی و ارزیابی می‌شوند.

گام‌های اصلی پروژه:

۱ مطالعه‌ی چند الگوریتم انتخاب فراگیرنده

۲ پیاده‌سازی برخی از الگوریتم‌های مطالعه شده

۳ ارزیابی الگوریتم‌های پیاده‌سازی شده

اطلاعات بیشتر:

یکی از الگوریتم‌های انتخاب فراگیرنده http://arxiv.org/pdf/1403.0178

طراحی و پیاده‌سازی رابطی مبتنی بر فایل

یکی از ایده‌های بسیار موفق یونیکس، معرفی رابطی (Interface) مبتنی بر فایل برای بسیاری از منابع موجود در سیستم عامل بوده است. استفاده از چنین رابطی سبب سادگی بسیاری از جنبه‌های یونیکس، از جمله برنامه‌های سیستمی و رابط‌های سیستم عامل شده است. استفاده از فایل به عنوان رابط، مزیت‌های دیگری نیز دارد، از جلمه: عدم وابستگی به یک زبان برنامه‌نویسی، استفاده از مکانیزم‌های کنترل دسترسی به فایل‌ها برای کنترل دسترسی به منابع، و استفاده از برنامه‌هایی که برای کار با فایل نوشته شده‌اند بدون تغییر. در سیستم‌های عامل جدیدتر نیز برای بسیاری از منابع سیستم عامل که در زمان سیستم عامل یونیکس مرسوم نبوده‌اند رابطی مبتنی بر فایل در نظر گرفته شده است. در سیستم عامل Plan 9 حتی برای منابعی مثل اتصالات شبکه نیز رابط مبتنی بر فایل طراحی شده است.

در این پروژه، رابطی مبتنی بر فایل برای برخی از منابع گوشی‌های همراه، مشابه خدماتی که سیستم عامل اندروید (Android) به پردازه‌ها ارائه می‌دهد، طراحی و پیاده‌سازی می‌شود. ابتدا خدماتی که سیستم عامل به پردازه‌های کاربری ارائه می‌دهد دسته‌بندی می‌گردند و سپس برای برخی از این منابعی رابط جدیدی مبتنی بر فایل ارائه داده می‌شود و مستند می‌گردد. سپس برای ارزیابی این رابط، با استفاده از فایل سیستم‌های محیط کاربری (Userspace) آنها پیاده‌سازی می‌گردند.

گام‌های اصلی پروژه:

۱ دسته‌بندی خدمات ارائه شده به برنامه‌ها در اندروید

۲ طراحی رابط برای برخی از خدمات دسته‌بندی شده

۳ پیاده‌سازی خدمات طراحی شده با FUSE

اطلاعات بیشتر:

فایل سیستم‌های محیط کاربری در لینوکس https://github.com/libfuse/libfuse

رابط گرافیکی برای نیتوی

یکی از قدیمی‌ترین و قدرتمندترین ویرایشگرهای یونیکس، وی (VI) می‌باشد که کاربران بسیار زیادی دارد. این ویرایشگر، امکانات بسیار زیادی را برای ویرایش سریع فایل‌ها در اختیار کاربر قرار می‌دهد که در ویرایشگرهای گرافیکی جدید یافت نمی‌شود. این ویرایش‌گر در دو محیط اصلی دستورات را اجرا می‌کند: در محیط EX دستورات خط به خط خوانده می‌شوند و اجرا می‌گردند و در محیط VI دستورات ویرایشی به صورتی تعاملی (Interactive) وارد و اجرا می‌گردند. نیتوی (Neatvi) یک پیاده‌سازی جدید از وی می‌باشد که امکان ویرایش خط‌های راست-به-چپ و فارسی را پشتیبانی می‌کند.

در این پروژه، یک رابط گرافیکی با استفاده از کتابخانه‌ی GTK یا QT برای نیتوی طراحی می‌شود که با آن بتوان به صورت گرافیکی متن فارسی را ویرایش کرد.

گام‌های اصلی پروژه:

۱ مستندسازی VI و شیوه‌ی کار با آن

۲ تغییر نیتوی برای افزودن رابط گرافیکی

اطلاعات بیشتر:

کد نیتوی http://repo.or.cz/neatvi.git

انتقال یک مترجم به معماری‌های جدید

بسیاری از مترجم‌ها (Compilers) می‌توانند کد نهایی را برای محیط‌ها یا معماری‌های گوناگونی تولید کنند. از این رو در مترجم-ها معمولا قسمت‌های مربوط به تولید کد نهایی به شکلی پیاده‌سازی می‌شود که افزودن پشتیبانی یک معماری جدید به راحتی قابل انجام باشد. یکی از کامپایلرهایی که با این دید نوشته شده است نیتسیسی (Neatcc) می‌باشد.

در این پروژه، قابلیت تولید کد نهایی برای یکی از معماری‌های رایج مثل MIPS یا ARM64 به کامپایلر نیتسیسی اضافه می‌شود. چون تولید کد نهایی، احتیاج به اطلاع از دستورات و جزئیات این معماری‌ها دارد، لازم است در گام اول این پروژه مهارت استفاده از دستورات این معماری‌ها ایجاد گردد.

گام‌های اصلی پروژه:

۱ مستندسازی ویژگی‌های اصلی و دستورات معماری

۲ انتقال مترجم به معماری جدید

۳ انجام آزمون‌های درستی کد نهایی

اطلاعات بیشتر:

معماری MIPS	https://en.wikipedia.org/wiki/MIPS_instruction_set
معماری ARM64	https://en.wikipedia.org/wiki/ARM64
کد نیتسیسی	http://repo.or.cz/neatcc.git

پس-پردازشگرهای نیتراف

مشابه کد میانی در کامپایلرها، هسته‌ی تیراف کدی تولید می‌کند که توسط پس-پردازشگرهای (Post-processor) آن به فرمت-های نمایش خروجی مثل PostScript تبدیل می‌گردد. وجود کد میانی تیراف سبب می‌شود که بدون تغییر برنامه‌ی اصلی تیراف و پیش-پردازشگرهای آن، امکان تولید مستند به یک فرمت خروجی جدید فراهم شود. یکی از فرمت‌های نمایش جدید OpenXPS (Open XML Paper Specification) می‌باشد.

در این پروژه یک پس-پردازشگر نیتراف برای تولید خروجی به فرمت OpenXPS پیاده‌سازی می‌گردد. این برنامه با خواندن کد میانی تیراف، آن را به فرمت OpenXPS تبدیل می‌کند. نوشتن چنین پس‌پردازشگری نیاز به آشنایی با معماری نیتراف و کد میانی آن و همین‌طور فرمت OpenXPS دارد.

گام‌های اصلی پروژه:

۱ معرفی فایل‌های توصیف قلم و کد میانی تیراف

۲ معرفی فرمت خروجی OpenXPS

۳ پیاده‌سازی پس-پردازشگر نیتراف

اطلاعات بیشتر:

فرمت OpenXPS http://www.ecma-international.org/publications/standards/Ecma-388.htm