بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

دانشگاه صنعتی نوشیروانی بابل

برق و کامپیوتر

پروژه کارشناسی رشته مهندسی کامپیوتر گرایش نرم‌افزار

اتوماسیون فرآیند انتخاب واحد بوسیله ارائه پیشنهاد برنامه های ترمی خودکار

نگارش

طه رستمی

استاد راهنما

دکتر سید محمود سخائی

آبان ۱۳۹۷

فهرست نوشتار

عنوان صفحه

فصل چهارم:نسخه آزمایشی و نتایج آزمایشی

فصل پنجم: نتیجه‌گیری و پیشنهادات

فهرست تصویر

عنوان صفحه

فصل اول:

مقدمه

۱-۱ موضوع پروژه

در این پروژه سعی شده است تا با تحلیل و بررسی و سپس توسعه برنامه ای کاربردی در قالب نمونه آزمایشی، راهکاری عملی برای اتوماسیون فرآیند انتخاب واحد که در ابتدای هر ترم توسط دانشجویان انجام می‌شود ارائه گردد.
لازم به ذکر است که نسخه آزمایشی برای مهندسی کامپیوتر و مطابق برنامه درسی مصوب سال ۹۲ وزارت علوم برای این رشته پیاده‌سازی شده است. همچنین برای انجام این پروژه علاوه بر طراحی و پیاده‌سازی الگوریتم‌های اساسی مورد نیاز، لازم بود تا جزییات مهمی مانند محدودیت‌ها و قوانین آموزشی، اولویت‌های دانشجویان و ... در نظر گرفته شود تا پروژه نه فقط یک برنامه کاربردی بلکه یک برنامه کاربردی کاربر پسند باشد. غیر از اینها این دید را هم داشتم تا پروژه طوری پایه گذاری شود که برای رشته‌های دیگر و سایر برنامه‌های درسی مصوب وزارت علوم به راحتی قابل توسعه باشد.

۱-۲ اهمیت و کاربردها

موضوع این پروژه از آن جهت دارای اهمیت است که می‌تواند فرآیند انتخاب واحد را برای دانشجویان تسریع و تسهیل نماید چرا که با پیشنهاد دادن تعدادی برنامه ترمی، مطابق با نیاز ها و اولویت های تعیین شونده توسط دانشجو، وی را قادر می‌سازد تا با توجه به سلیقه خویش برنامه‌ای را جهت انتخاب واحد برگزیند و حال آنکه پیدا کردن چنین برنامه-ای به صورت دستی در عمل، معمولا کاری دشوارتر و اغلب زمانبرتر است.

علاوه بر این دانشگاه ها نیز می‌توانند مشتریان این پروژه باشند و از آن برای اعمال تغییراتی در فرآیند انتخاب واحد ترمی دانشجویان استفاده کنند. برای مثال آنها می‌توانند دانشجویان را قادر سازند تا پس از انتخاب برنامه، علاوه بر آنکه دانشجو می‌تواند کُد درس‌ها را به صورت دستی در سامانه ثبت کند، امکان ثبت اتوماتیک درس‌ها با توجه به برنامه انتخابی را نیز اضافه کنند.

۱-۳تعریف صورت مسئله به صورت دقیق‌تر

در ابتدای هر ترم، دانشگاه‌ها برنامه‌ای تحت عنوان **دروس ارائه شده در ترم** ارائه می‌کنند که در واقع جدولی است که سطرهای آن شامل مواردی از جمله درس ، دانشکده ارائه دهنده درس ، استاد ارائه دهنده درس ، زمان و مکان برگزاری کلاس‌ها و تاریخ امتحان ، ظرفیت و ... می‌باشد و دانشجو در ابتدای هر ترم باید با توجه به این جدول برنامه‌ای را برای ترم خود برگزیند.

هر دانشجو علاوه بر اینکه خود را ملزم می‌داند تا برنامه‌ای که انتخاب می‌کند مطابق با قوانین، از جمله قوانین مصّوب وزارت علوم و آئین نامه‌های داخلی دانشگاه باشد، به دنبال برنامه‌ای است که با اولویت‌ها و سلیقه وی مطابق باشد.

بنابراین توضیح صورت مسئله به صورت دقیق‌تر این است که ما می‌خواهیم دانشجو را در یافتن برنامه ترمی مورد نظرش یاری کنیم و به جای آنکه دانشجو به طور دستی بخواهد برنامه ای را برای خود برگزیند ما بر اساس اولویت های وی

برخی از برنامه‌های ترمی مجازی که محدودیت‌های قانونی را ارضا می‌کنند پیدا کنیم و تعدادی از بهترین برنامه‌هایی را که در بازه زمانی تعیین شده پیدا کرده‌ایم به او ارائه دهیم.

تا به اینجا ما **صورت مسئله** را به صورت دقیق‌تر بیان کردیم اما این صورت مسئله همچنان ابهام دارد چرا که ما هنوز محدوده پروژه را مشخص نکرده‌ایم و به همین دلیل از محدودیت‌هایی که با توجه به حدود پروژه مشخص می‌شوند غافل مانده‌ایم.

ما محدوده پروژه را بدین صورت تعریف می‌کنیم که، پروژه ما با توجه به سیستم **گلستان** که در حال حاضر یکی از سامانه های مورد اقبال در زمینه سیستم های جامع آموزشی در میان دانشگاه های کشور محسوب می‌شود و برای رشته-های فنی و مهندسی و براساس قوانین دانشگاه صنعتی نوشیروانی بابل انجام شود.به علاوه مسئله فقط لازم است برای دانشجویانی که تنها در یک رشته تحصیلی مشخص تحصیل می‌کنند و برای دوره کارشناسی حل شود.

با مشخص شدن حدود پروژه محدودیت های جدیدی به پروژه اعمال می‌شوند برای مثال گفته شد پروژه قرار است برای سیستم گلستان نوشته شود و در حال حاضر در هنگام انتخاب واحد در این سیستم شما باید واحدها را به ترتیبی ثبت کنید که همواره برنامه ترمی شما در وضعیت معتبری قرار داشته باشد یعنی برای مثال جهت اخذ درسی که همنیازی دارد ابتدا باید همنیاز آن درس را وارد کرده سپس خود درس را وارد کنید بنابراین به صورت مسئله این خواسته به صورت ضمنی اضافه می‌شود که علاوه بر پیشنهاد برنامه ترمی، درس‌های آن برنامه پیشنهادی را با ترتیبی قابل اخذ برای سیستم گلستان به نمایش درآورد.

بنابراین آنچه پیش‌تر بیان شد به علاوه محدودیت‌هایی که حدود پروژه را تعیین می‌کند صورت مسئله را به صورت دقیق تعریف می‌کنند.

٤-١ اهداف پروژه

ما می‌خواهیم مسئله‌ای که پیش‌تر در آخر ۳-۱ به صورت دقیق تعریف کردیم را تحلیل کرده و یک برنامه کاربردی در قالب نسخه‌ای آزمایشی که بتواند مسئله را به صورت کارا حل کند پیاده‌سازی کنیم. همچنین ما می‌خواهیم در تحلیل و حتی‌الامکان در پیاده‌سازی خود به گونه‌ای عمل کنیم که این پروژه به جای آنکه به محدودیت های تعیین شده توسط حدود پروژه وابسته باشد، این محدودیت‌ها را ارضا کند اما تحلیل و پیاده‌سازی به گونه‌ای باشد که بتوان پروژه را برای محدوده‌های دیگر به راحتی و به سرعت انجام داد برای مثال با صرف زمانی اندک یا حتی بدون صرف هیچ زمانی بتوان آن را برای دانشگاه دیگری پیاده‌سازی کرد یا بتوان آن را برای برنامه‌های درسی وزارت علوم که در آینده می‌آیند، به سرعت پیاده‌سازی کرد.

۵-۱ گام‌های انجام پروژه

پروژه‌ای که با آن سر و کار داریم یک پروژه نرم‌افزاری است و برای انجام یک پروژه نرم‌افزاری به صورت کلی می‌توانیم گام های زیر را در نظر بگیریم:

- نیاز های پروژه را تشخیص داده و پروژه را تحلیل کنیم.
- به طراحی و پیاده‌سازی نرم‌افزار بپردازیم.
- به بازبینی و تست نرم‌افزار و ارزیابی تست بپردازیم.

مواردی که در بالا به آن اشاره کردیم بسیار کلی هستند و علیرغم آنکه تقریبا برای هر پروژه نرم‌افزاری لازم و ضروری می‌باشند اطلاعات ملموسی در اختیار ما نمی‌گذارند همچنین از ترتیب انجام کارها نیز اطلاعاتی در اختیار نمی‌گذارند بنابراین فکر کردم بهتر است به صورت تیتر وار مراحلی که واقعا برای انجام این پروژه طی شده است را ذکر کنم. این مراحل به ترتیب شامل موارد زیر می‌باشند:

- مطالعه و بررسی جدیدترین برنامه‌های درسی فنی مهندسی مصوب وزارت علوم و تحلیل کلی پروژه که طی آن داده‌های ورودی مسئله و همچنین مراحل لازم برای انجام مسئله از قبیل گام کاستن۱، گام کاستن۲، الگوریتم پیشنهاد برنامه ترمی و امکانات بیشتر بدست آمدند به علاوه مشخص کردن تکنولوژی‌های مورد استفاده برای پیاده‌سازی و بررسی معماری های مختلف نیز در این بخش انجام شد.
- تهیه داده های اولیه - پیاده‌سازی بخش فراهم کننده خدمات فایل‌ها و مدلسازی دروس ارائه شده در ترم مبتنی بر سیستم گلستان – به علاوه تست بخش پیاده‌سازی شده.
- مطالعه مجدد برنامه های درسی وزارت علوم به علاوه برنامه های تازه تصویت شده در فاصله این بخش و بخش دوره قبل با عمق بیشتر – تحلیل و بررسی این برنامه ها و تلاش برای ارائه ساختاری عمومی که بتواند برنامه‌های درسی وزارت علوم را مدل کند – پیاده‌سازی ساختار ایجاد شده و تست آن.
- پیاده‌سازی و تست بخش مقدماتی نرم‌افزار – پیاده‌سازی و تست گام‌های کاستن.
- تحلیل و پیاده‌سازی و تست امکانات بیشتر شامل نمایش جدولی برنامه‌های پیشنهاد شده توسط نرم‌افزار و نمایش آنها با ترتیبی قابل اخذ برای سیستم گلستان به علاوه امکان افزودن اطلاعات جانبی برای هر برنامه ترمی پیشنهادی و ذخیره‌سازی آن.
- تهیه داده های جامع برای تست و ارزیابی نرم‌افزار.
- طراحی الگوریتم برای جستجو و یافتن و سپس پیشنهاد دادن برنامه های ترمی مناسب – پیاده‌سازی و تست الگوریتم.
- پیاده‌سازی بخش های **خانه** و **تنظیمات** و تست آنها.

- ارزیابی نرم‌افزار، تشخیص عیوب و سپس بازنگری و اصلاح کد به علاوه تغییر بعضی از نام‌گذاری‌ها.

ارزیابی مجدد نرم‌افزار.

ما فعلا از توضیحات فنی که دقیقا هر کدام از این موارد ذکر شده چه هستند صرف نظر می‌کنیم و در بخش‌های بعدی با این که هر کدام از این موارد دقیقا چه هستند بیشتر آشنا می‌شویم.

همانطور که در بالا ذکر شده است گام‌های انجام پروژه مانند آنچه در ابتدا به صورت کلی و در سه بخش ارائه شده بود قابل تفکیک و جدا از هم نیست بلکه بنا به فراخور هر مرحله برای انجام آن مرحله اقدام شده‌است با این وجود از آنجا که نوشتن گزارشی بر مبنای ده موردی که به آنها اشاره شده بود کاری دشوار است ما فصل‌بندی ها را به گونه‌ای قرار می‌دهیم که فهم پروژه را تسهیل کند و توضیح و تشریح آن آسان تر گردد.

۶-۱ خلاصه

در این فصل ابتدا پروژه را به صورت کلی معرفی کردیم و بعد از آن گفتیم که چرا این پروژه می‌تواند حائز اهمیت باشد سپس به تعریف دقیق مسئله‌ای که این پروژه قصد حل کردن آن را دارد پرداختیم و پس از برشمردن اهداف پروژه، گام‌های انجام آن را ابتدا در حالت کلی و سپس در حالت واقعی بیان کردیم و پس از مقایسه این دو حالت، دلایلی آوردیم که شیوه تنظیم فصل‌های ما را روشن می‌کند.

فصل دوم:
طراحی اولیه و آشنایی با مفاهیم پایه

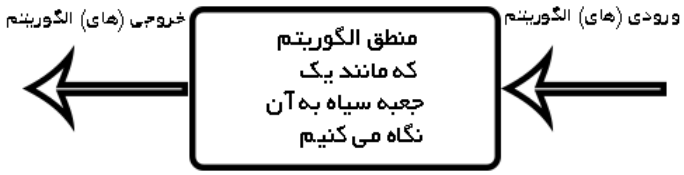
۲-۱ مقدمه

شرح کامل و جز به جز طراحی پروژه می‌تواند کاری خسته کننده باشد چرا که در فرآیند طراحی اغلب به جزئیاتی برمی‌خوریم که برای دقیق‌تر بودن پروژه نیاز است آنها را بررسی کنیم اما در عمل و پس از اتمام این فرآیند متوجه می‌شویم به بسیاری از این جزئیات نیازی نداشته‌ایم. از این جهت ما در اینجا به جای آنکه بخواهیم جز به جز آنچه بررسی کرده‌ایم را بیان کنیم، به تشریح نتایج حاصل از طراحی می‌پردازیم چرا که آنچه که عملا با آن سر و کار خواهیم داشت آنها هستند.

۲-۲ فشرده اول از طراحی

پس از تفکر پیرامون مسئله‌ای که در این پروژه قصد حل کردن آن را داشتیم متوجه شدیم که قلب این پروژه طراحی الگوریتمی جهت پیشنهاد برنامه ترمی می‌باشد.

از آنجا که هر الگوریتم در حالت کلی به ترتیب شامل سه قسمت: ورودی، منطق الگوریتم و خروجی آن می‌باشد، سعی کردیم صرف نظر از جزئیات تصور کنیم که ورودی و خروجی الگوریتم پیشنهاد برنامه ترمی باید چگونه باشد.



شکل ۲-۱ دید ما از یک الگوریتم در حالت کلی

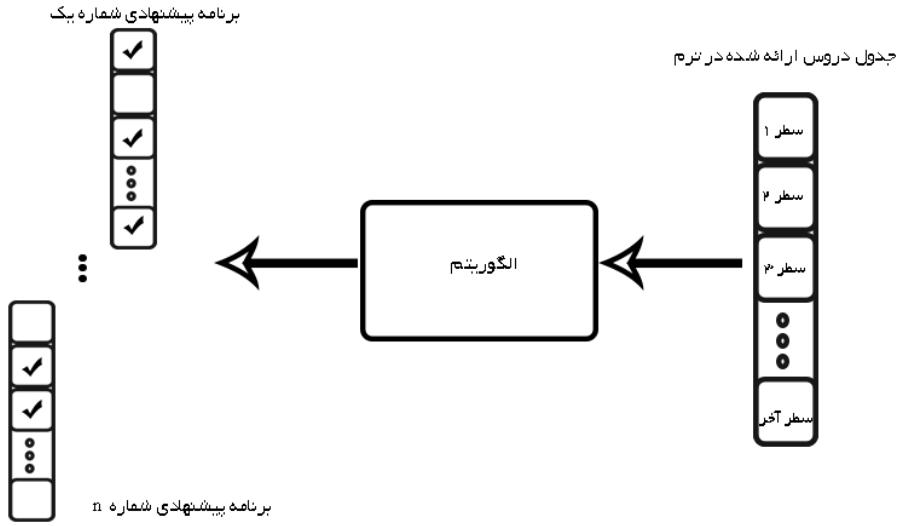
در واقع منطق الگوریتم را مانند جعبه سیاهی در نظر گرفتیم که فعلا نمی‌دانیم که قرار است چگونه باشد اما با مشخص کردن ورودی ها و خروجی های حاصل از آن می‌خواهیم عملکرد و تاثیری که این جعبه سیاه در خروجی هایش می‌گذارد را شناسایی کنیم.

در فصل قبل گفتیم **دروس ارائه شده در ترم** به صورت یک جدول، هر ترم توسط دانشگاه ارائه می‌گردد و این جدول شامل سطر هایی شامل مواردی از جمله درس ، دانشکده ارائه دهنده درس ، استاد ارائه دهنده درس ، زمان و مکان برگزاری کلاس‌ها و تاریخ امتحان ، ظرفیت و ... می‌باشد. در شکل زیر می‌توانید نمونه‌ای واقعی از سطرهایی از این جدول را که در سیستم گلستان ارائه شده است مشاهده کنید.

شماره و گروه درس	نام درس	واحد		ظرفیت	ثبت نام شده	تعداد لیست انتظار	جنسیت	نام استاد	زمان و مکان ارائه/ امتحان	توضیحات
		کل	ع							
۱۲۲۱۰۵۴_۰۱	هوش مصنوعي و سيستم هاي خبره	۳	۰	۵۰	۴۴	۰	مختلط	عمرانپور بندپي حسام	درس(ن): يك شنبه ۱۷:۳۰-۱۶:۰۰ مكان: ۲۰۸ دانشكده برق درس(ن): سه شنبه ۱۷:۳۰-۱۶:۰۰ مكان: ۲۰۸ دانشكده برق امتحان(۱۳۹۷/۰۳/۲۸) ساعت : ۱۶:۰۰-۱۴:۰۰	
۱۲۲۱۰۵۵_۰۱	اصول طراحي كامپايلر	۳	۰	۴۰	۲۷	۰	مختلط	غلامي رودي علي	درس(ن): يك شنبه ۱۵:۳۰-۱۴:۰۰ مكان: ۱۰۹ دانشكده برق درس(ن): سه شنبه ۱۵:۳۰-۱۴:۰۰ مكان: ۱۰۹ دانشكده برق امتحان(۱۳۹۷/۰۳/۲۰) ساعت : ۱۰:۰۰-۸:۰۰	
۱۲۲۱۰۵۶_۰۲	آزمايشگاه سيستم هاي عامل	۱	۱	۱۲	۱۲	۰	مختلط	جواهري عطيه	درس(ع): سه شنبه ۱۲:۰۰-۱۰:۰۰	
۱۲۲۱۰۵۶_۰۳	آزمايشگاه سيستم هاي عامل	۱	۱	۱۲	۱۱	۰	مختلط	غلامي رودي علي	درس(ع): سه شنبه ۱۲:۰۰-۱۰:۰۰	
۱۲۲۱۰۵۷_۰۲	آزمايشگاه مدارهاي منطقي و معماري كامپيوتر	۱	۱	۱۲	۱۲	۰	مختلط	كاكوئي محمد	درس(ع): سه شنبه ۱۰:۰۰-۸:۰۰	
۱۲۲۱۰۵۷_۰۳	آزمايشگاه مدارهاي منطقي و معماري كامپيوتر	۱	۱	۱۲	۱۲	۰	مختلط	كاكوئي محمد	درس(ع): سه شنبه ۱۸:۰۰-۱۶:۰۰	
۱۲۲۱۰۵۷_۰۴	آزمايشگاه مدارهاي منطقي و معماري كامپيوتر	۱	۱	۱۲	۱۲	۰	مختلط	كاكوئي محمد	درس(ع): شنبه ۱۶:۰۰-۱۴:۰۰	
۱۲۲۱۰۵۸_۰۱	روش پژوهش و ارايه	۲	۰	۴۵	۴۰	۰	مختلط	رماني فاطمه	درس(ن): يك شنبه ۱۶:۰۰-۱۴:۰۰ مكان: ۱۰۵ دانشكده برق امتحان(۱۳۹۷/۰۴/۰۹) ساعت : ۱۶:۰۰-۱۴:۰۰	
۱۲۲۱۰۵۹_۰۱	كارآموزي (مهندسي كامپيوتر)	۱	۰	۱۰	۱	۰	مختلط	اساتيد گروه آموزشي		
۱۲۲۱۰۶۶_۰۱	پروژه نرم افزار	۳	۳	۳۰	۳۰	۰	مختلط	اساتيد گروه آموزشي		دانشجويان جهت ثبت استاد وعنوان پروژه ازطريق پيشخوان خدمت اقدام نمايند.

شکل ۲-۲. سطرهایی از جدول دروس ارائه شده در یکی از ترم ها

ما همه‌ی ورودی‌ها و خروجی‌های الگوریتم را در این مرحله تعیین نمی‌کنیم اما بدیهی است که یکی از ورودی‌های اصلی برای مسئله ما جدول **دروس ارائه شده در ترم** می‌باشد و الگوریتم باید در خروجی تعدادی از سطرهای این جدول را که می‌خواهد به عنوان برنامه پیشنهاد دهد برگرداند. با این اوصاف می‌توانیم هر سطر از این جدول را مانند جعبه سیاهی در نظر بگیریم که الگوریتم در مورد انتخاب شدن یا انتخاب نشدن آن سطر در هر برنامه پیشنهادی باید تصمیم گیری کند.



شکل ۲-۳ تشخیص نحوه برخورد الگوریتم با ورودی ها از روی خروجی ها

اما نکته مهم این است که تعداد سطرهایی که در جدول دروس ارائه شده در ترم وجود دارند نسبت به آنچه دانشجو مجاز به اخذ آنها در آن ترم است بسیار بیشتر است. این نکته برای ما الهام بخش ایده‌ای بود که آن را **کاستن** نام نهادیم.

ایده کاستن به این معنی است که ما سعی کنیم از فضای بزرگ مسئله بکاهیم و آن را کوچک‌تر کنیم.در واقع این ایده سعی می‌کند برخی از سطرهای جدول دروس ارائه شده در ترم را، که با توجه به شرایط دانشجو غیر قابل اخذ می‌باشند قبل از ارسال برای الگوریتم از ورودی حذف کند.

۳-۲ فشرده دوم از طراحی

مشاهده کردیم که ایده کاستن چگونه می‌تواند با تلاش برای کوچک کردن فضای مسئله، فضای جستجو برای الگوریتم را کوچک‌تر کند. اما تا به اینجا ایده کاستن را توصیف کردیم و هنوز نگفتیم که این ایده قرار است چگونه در عمل به‌کار گرفته شود. در ادامه به تشریح این مقوله می‌پردازیم.

۱-۳-۲ گام کاستن ۱

هر دانشجو شرایطی دارد که با توجه به آن ممکن است دامنه سطرهای قابل اخذ جدول برنامه دروس ارائه شده در ترم برایش محدودتر شود.برای مثال باید سطرهایی که برای جنسیت زن به‌کار برده شده است را از دامنه سطرهای قابل اخذ برای دانشجو مرد حذف کرد.

دیدیم که یکی از راه‌های کاستن استفاده از شرایط دانشجو است. ما پس از تحلیل و بررسی بیشتر، شرایطی از دانشجو را که می‌توان از آن برای کاستن استفاده کرد مشخص کردیم و از میان آنها آن شرایطی را که می‌خواستیم در نرم‌افزار پیاده‌سازی کنیم مشخص کردیم.

در نهایت شرایطی را که برای پیاده‌سازی انتخاب کردیم شامل جنسیت، شماره ترمی که دانشجو در آن می‌خواهد انتخاب واحد کند(یعنی دانشجو ترم چند است)،تاریخچه دانشجو و تعداد واحد گذرانده شده توسط دانشجو (که این یک خصوصیت محاسباتی است و می‌توان آن را با توجه به تاریخچه دانشجو محاسبه کرد) می‌باشد.

ما تمام شرایط موجود دانشجو را انتخاب نکردیم برای مثال در سیستم گلستان سطرهایی وجود دارند که مشخص می‌کنند آن سطر خاص فقط برای ورودی های خاصی مثلا ورودی های ۹۱ و قبل از آن قابل اخذ می‌باشد و ما چنین شرطی را در لیستی که پیشتر گفتیم انتخاب نکردیم در این رابطه باید به چند نکته مهم توجه کرد. نکته اول آنکه برای این مثال خاص باید بدانید که سیستم گلستان برای هر سطر یک قسمت مربوط به توضیحات دارد و از آنجا که توضیحات احتمالا می‌توانند هر چیزی باشند ما در حالت کلی نمی‌توانیم توضیحات را توسط نرم‌افزار پردازش کنیم مگر آنکه به برنامه قابلیت **پردازش زبان طبیعی** بدهیم و از آنجا که چنین موردی در دامنه فعلی پروژه ما نمی‌گنجید آن را انتخاب نکردیم. نکته دوم اینکه با این حال موردی که ذکر شد جز مواردی است که باید حتما حذف سطر و کاستن برایش اتفاق بیفتد چرا که اگر در برنامه پیشنهادی الگوریتم چنین سطری موجود باشد، آن برنامه پیشنهادی نامعتبر خواهد بود ما برای مواجه با این مسئله دو راه حل داشتیم اولین راه‌حل آنکه هر چند قصد نداریم زبان طبیعی را پردازش کنیم اما این نوع توضیحات در سیستم گلستان و در دانشگاه صنعتی نوشیروانی بابل تا به اینجا اغلب و یا شاید همیشه با الگوهای خاص قابل تشخیصی نوشته شده‌اند بنابراین راه اول ما می‌تواند این باشد که این الگوها را شناسایی کنیم و پردازش را به همین الگوهای شناسایی شده محدود کنیم اما همانطور که در فصل اول قسمت اهداف پروژه ذکر شده بود ما نمی‌خواهیم پروژه خود را وابسته به محدوده مسئله کنیم و حال آنکه چنین راهکاری دقیقا یکی از مواردی است که می‌تواند قسمتی از پروژه را وابسته به محدوده مسئله کند. در عوض ما راهکار دیگری را برگزیدیم که در آن این نوع تشخیص بر عهده کاربر قرار می‌گیرد که در ادامه با آن آشنا خواهید شد.اما نکته سوم یک تاکید بیشتر است بر این موضوع که توجه

داشته باشید که ما گفتیم می‌خواهیم **سعی**(سعی با انجام دادن حتمی کار بصورت کاملا موفق متفاوت است) کنیم از فضای حالات کلی بکاهیم و کاستن در اینجا به معنی کوچک‌تر کردن فضای مسئله با حذف **برخی** از سطرهای غیر مجاز است و نباید به اشتباه تصور کرد که الزاما آنچه بعنوان ورودی به الگوریتم داده می‌شود صرفا حالات مجاز است.

ما پیشتر از لفظ **تاریخچه دانشجو** استفاده کردیم مقصود ما از تاریخچه دانشجو اطلاعات مربوط به دروسی است که دانشجو آنها را با موفقیت گذرانده یا آنها را اخذ کرده اما نتوانسته با موفقیت بگذراند. ما با استفاده از اطلاعات حاصل از لیست انتخاب شده برای شرایط دانشجو به خصوص از تاریخچه دانشجو و با توجه به برنامه درسی وزارت علوم که دانشجو با آن در ارتباط است می‌توانیم متوجه شویم دانشجو در یک ترم خاص مجاز است چه درس‌هایی را در صورت ارائه شدن اخذ نماید.

از این پس ما به فرآیند حذف کردن سطرهایی از جدول برنامه دروس ارائه شده در ترم که با استفاده از شرایط انتخاب شده دانشجو شامل جنسیت، شماره ترمی که دانشجو در آن می‌خواهد انتخاب واحد کند، تاریخچه دانشجو و تعداد واحد گذرانده شده توسط دانشجو به علاوه برنامه و اطلاعات مخصوص مربوط به برنامه درسی مصوب وزارت علوم که مرتبط با دانشجو می‌باشد، **گام کاستن ۱** می‌گوئیم.

۲-۳-۲ گام کاستن ۲

گام کاستن ۱ تلاش لازمی بود که باید نهایتاً در قسمتی از پروژه اگر نه خود آن ولی حداقل عملکرد آن اعمال می‌شد با این حال این گام تنها گامی نیست که می‌توان در جهت کاهش فضای حالات برداشت. در این قسمت با گام کاستن ۲ آشنا می‌شوید که گامی است در حالت کلی، اختیاری ولیکن ممکن است در شرایطی به حالت اجبار نیز در آید.

شاید پیدا کردن یک برنامه ترمی مجاز با توجه به جدول برنامه دروس ارائه شده در ترم اغلب کار آسانی باشد اما کار زمانی سخت می‌شود که پای اولویت‌های محدود کننده دانشجو به میان می‌آید. بنابراین پیش از آنکه بگوئیم گام کاستن ۲ چیست می‌خواهیم بدانیم اولویت‌های دانشجویان برای انتخاب برنامه ترمی مورد نظرشان چگونه است.

دانشجویان برنامه ترمی خود را بر اساس اولویت‌هایی انتخاب می‌کنند که ما این اولویت‌ها را به دو دسته کلی **اولویت‌های باید** و **اولویت های شاید** دسته‌بندی کردیم. اولویت های **باید** اولویت‌هایی هستند که باید در برنامه ترمی دانشجو **الزاما** ارضا شوند در حالی که اولویت های **شاید** الزام آور نیستند و می‌توانند در حالت کلی با سه جمله **بهتر است این گونه باشد** ، **بهتر است اینگونه نباشد** و **فرقی ندارد اینگونه باشد یا نباشد** توصیف شوند در کنار دو نوع اولویت **باید** و **شاید**، اولویت دیگری به نام **نباید** وجود دارد اما از آنجا که می‌توان **نباید** را با نقیض **باید** توصیف کرد از قرار دادن آن به عنوان یک دسته جداگانه از اولویت ها خودداری کردیم.

رعایت قوانین از جمله قوانین وزارت علوم و آئین‌نامه‌های داخلی دانشگاه از اولویت های **باید** به حساب می‌آیند اما اولویت های **باید** الزاما محدود به رعایت قوانین نیستند بلکه اولویت های **باید** می‌توانند بر اساس آگاهی دانشجو از اموری **حتمی** ولی **غیر قابل مشاهده** برای ما(نرم‌افزار) الزام آور شوند.

می‌توان منشا اولویت‌های دانشجو را از **اهداف استراتژیک** وی دانست. اهداف استراتژیک اهدافی هستند که دانشجو با توجه به آنچه خود برنامه ریزی و هدف گذاری می‌کند مشخص می‌شوند. برای مثال ممکن است دانشجویی بخواهد همزمان با تحصیل کار نیمه وقتی داشته باشد و بنابراین زمان‌های خاصی را الزاما در برنامه ترمی خود بخواهد خالی نگه دارد یا مثلا دانشجویی با این هدف که هر چه زود تر فارغ‌التحصیل شود بخواهد هر ترم حداکثر تعداد مجاز واحد را اخذ کند یا بعنوان مثالی دیگر دانشجویی درس خاصی را بخواهد حتما با استاد خاصی بگیرد یا نگیرد و مثال های این چنینی دیگر که بسیارند. بنابراین برنامه ما باید این قابلیت را داشته باشد که دانشجو بتواند با استفاده از آن اولویت های مد نظرش را برای نرم‌افزار مشخص کند.

پس از بحث بالا در این نوبت باید مشخص کنیم که نرم‌افزار ما چه اولویت‌هایی را می‌خواهد پوشش دهد. برای رسیدن به این هدف ابتدا فهرستی از برخی نتایج معمول حاصل از اهداف استراتژیک دانشجویان به شرح زیر تهیه کردیم:

- مشخص کردن محدوده واحد (مثلا این که دانشجو بخواهد برنامه ترمی وی بین ۱۶ تا ۱۹ واحد باشد)
- اخذ کردن یک یا چند درس به صورت **باید**(دانشجو بخواهد درس یا درس‌هایی را در ترم جاری حتما اخذ کند)
- اخذ نکردن یک یا چند درس به صورت **نباید**(دانشجو نخواهد در هر حال یک یا چند درس خاص را در ترم جاری اخذ کند.)
- اخذ کردن یک گروه درسی خاص(معادل یک سطر از جدول برنامه دروس ارائه شده در ترم) از یک درس به صورت باید(مثلا از بین چهار گروه ارائه شده برای درس سیگنال و سیستم دانشجو بخواهد حتما یک سطر خاص از این درس اخذ شود)
- اخذ نکردن یک یا چند گروه خاص از یک درس به هیچ وجه.
- اخذ کردن یکی از چند درس مشخص بصورت **باید**(مثلا دانشجو بخواهد بین دو درس از گروه معارف حتما یک درس را از بین آنها در ترم جاری اخذ کند)

علاوه بر موارد بالا می‌توان برای بعضی از آنها حالت **شاید** نیز اضافه کرد (برای مثال اینکه بگوییم این درس بهتر است در این ترم اخذ شود یا نشود) همچنین احتمالا علاوه بر همه این‌ها بتوان موارد دیگری نیز به این فهرست افزود.

اگر بخواهیم از نگاه توانمندی کاربر به این فهرست نگاه کنیم هر چه این فهرست شامل موارد جزئی‌تر شود احتمالا کاربر در توصیف اولویت هایش توانا تر خواهد بود اما دو نکته از نظر تحلیل و طراحی وجود دارد اول آنکه زیاد شدن بیش از اندازه موارد می‌تواند به جای آنکه کار کاربر را آسان کند عملا باعث سختی بیشتر کار او شود چرا که

قبل از آنکه کاربر بتواند از امکانات نرم‌افزار استفاده کند باید بداند که آنها چیستند و چگونه کار می‌کنند. نکته دوم اینکه از دید تحلیلی فقط توصیف اولویت های باید می‌توانند به کاستن فضای حالات مسئله به صورت مستقیم کمک کنند چرا که الویت های **شاید** تنها درصدی از اقبال و علاقه کاربر نسبت به سطرهای جدول برنامه دروس ارائه شده در ترم را بیان می‌کنند و در نتیجه نمی‌توان آنها را از جدول حذف کرد. با این اوصاف ما باید امکاناتی را برای پیاده‌سازی انتخاب کنیم که هم بتواند کاربر را در توصیف اولویت هایش توانا کند و هم باید از زیاد شدن بیش از اندازه امکانات به گونه‌ای که عملا کار را برای کاربر دشوار کند پرهیز کنیم، غیر از آن نباید این امکانات به گونه‌ای کم یا غیر کاربردی باشند که کاربر تمایلی به استفاده از آنها نداشته باشد(توجه داریم که گام کاستن ۲ در حالت کلی اختیاری است) علاوه بر همه این ها تمایل داریم امکاناتی را انتخاب کنیم که از نظر پیاده‌سازی نیز ساده‌تر باشند. با بررسی کردن همه این موارد به این نتیجه رسیدیم که پنج مورد اول از فهرست پیش‌تر بیان شده را پیاده‌سازی کنیم. در واقع برای گزینه های شاید فقط حالتی که با **فرقی ندارد اینگونه باشد یا نباشد** بیان می‌شود را باقی گذاشتیم و مورد ششم از لیست ذکر شده را صرفا برای راحت تر شدن پیاده‌سازی و همچنین بخاطر ترس از پیچیده شدن بیش از اندازه الگوریتم از لیست انتخابی حذف کردیم.

اما برای اینکه بتوانیم با توجه به اولویت‌های کاربر در مورد مواردی از قبیل **ممکن بودن** اولویت کاربر، متناقض نبودن آن با سایر اولویت‌هایش آگاهی یابیم و همچنین برای تصمیم گیری در مورد حذف کردن برخی از سطرها جهت کاستن از فضای حالات، باید بدانیم که در بررسی‌هایمان کلاس‌هایی که ظرفیت آنها پُر شده است را فیلتر کنیم یا فیلتر نکنیم به علاوه باید بدانیم تداخل امتحانی نیز چک شود یا نشود. پس باید این دو مورد نیز از کاربر پرسیده شود.

از این پس به فرآیند دریافت اطلاعات و اولویت‌های کاربر و تلاش برای حذف سطرهایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت های مشخص شده، **گام کاستن۲** می‌گوئیم.

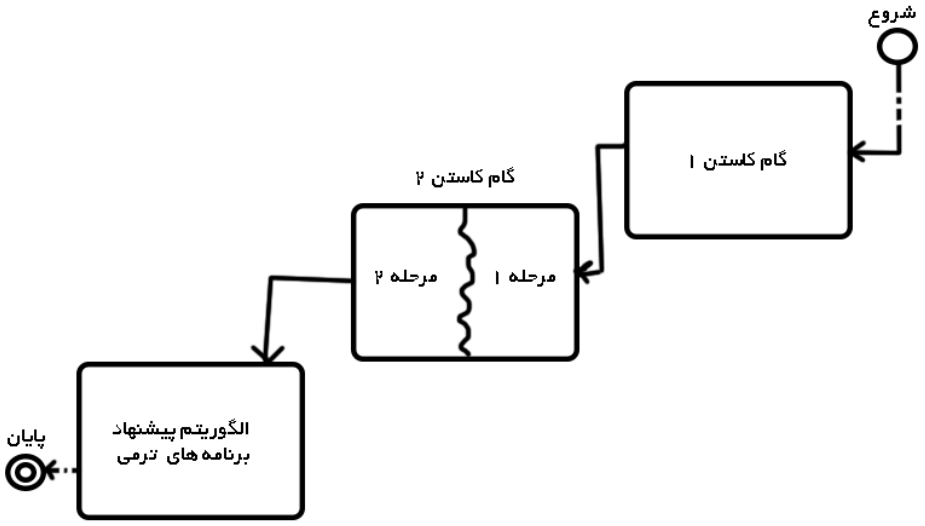
همچنین به فرآیند دریافت اولویت‌ها و اطلاعات کاربر شامل حداقل و حداکثر واحد برای اخذ در ترم جاری، بررسی یا عدم بررسی تداخل امتحانات و فیلتر کردن یا فیلتر نکردن سطرهای با ظرفیت پُر و تلاش برای حذف سطرهایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت های مشخص شده، **مرحله۱ از گام کاستن۲** می‌گوئیم.

همین طور به فرآیند دریافت اولویت‌های کاربر به جز آنچه در مرحله۱ از گام کاستن۲ مشخص شد و تلاش برای حذف سطرهایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت‌های مشخص شده، **مرحله۲ از گام کاستن۲** می‌گوئیم.

قبل از به پایان بردن این بخش می‌خواهیم به قسمتی که در آن این سوال مطرح شده بود که در قبال ستون توضیحات سیستم گلستان چگونه عمل کنیم بازگشته، پاسخ دهم. اکنون اگر به فهرست مواردی که برای گام کاستن۲ مشخص کرده‌ایم توجه کنید می‌بینید می‌توان به راحتی با استفاده از این گام مشکل مربوط به ستون توضیحات را به کمک کاربر حل کرد و این یکی از همان شرایطی است که طی آن گام کاستن۲ از حالت اختیاری خارج می‌شود.

٤-٢ سایر مولفه‌ها و سازماندهی بخش‌های مربوط

با توجه به آنچه تا به اینجا گفته شد می‌توان پروژه را پس از دریافت ورودی اولیه (یعنی ساختار توصیف کننده برنامه درسی مصوب وزارت علوم که با دانشجو در ارتباط است) به ترتیب به صورت بخش‌های گام کاستن۱، مرحله۱ از گام کاستن۲ ،مرحله۲ از گام کاستن۲ و سپس الگوریتم پیشنهاد برنامه ترمی در نظر گرفت.



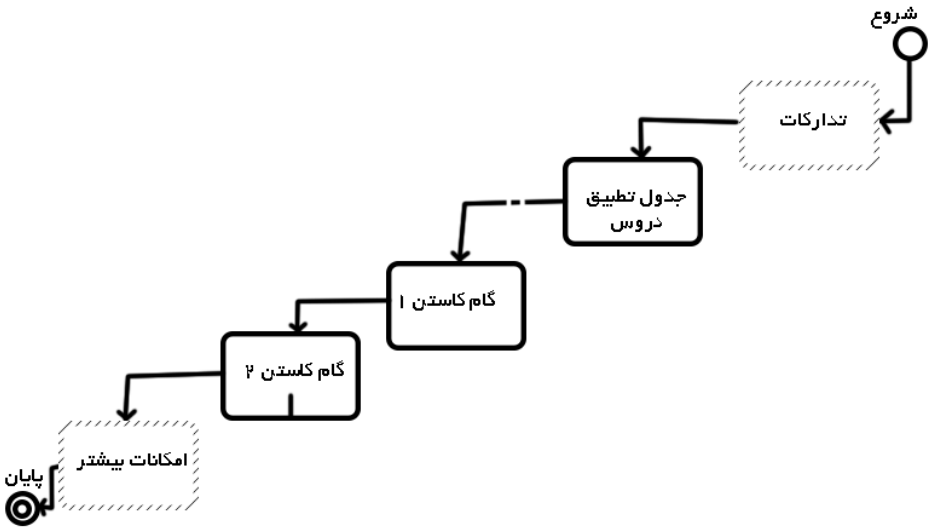
شکل ۲-٤ شمای کلی از مولفه های اصلی پروژه

حالا به طور فشرده به سایر مولفه‌ها می‌پردازیم.

دانشگاه‌های مختلف الزاما تمام دروس موجود در برنامه درسی مصوب وزارت علوم را ارائه نمی‌دهند به علاوه گاهی به جای یک درس موجود در برنامه مصوب وزارت علوم، درس معادلی را به جای آن ارائه می‌دهند. ما در مواجه با این موضوع مولفه‌ای به نام **جدول تطبیق دروس** را به مولفه های نرم‌افزار اضافه کردیم. این جدول شامل سطرهایی با چهار ستون می‌باشد که این ستون‌ها عبارتند از شناسه درس، عنوان درس در برنامه درسی مصوب وزارت علوم ، نام درس در دانشگاه مقصد و کُد درس در دانشگاه مقصد.

از این پس به درس‌هایی از برنامه درسی مصوب وزارت علوم مرتبط با دانشجو که در دانشگاه مقصد(دانشگاهی که دانشجو با توجه به برنامه ارائه شده توسط آن دانشگاه انتخاب واحد می‌کند) ارائه می‌شوند **درس‌های موجود** و به آنهایی که ارائه نمی‌شوند **درس‌های ناموجود** می‌گوییم.

علاوه بر مولفه جدول تطبیق دروس، دو مولفه به نام‌های مولفه **تدارکات** و مولفه **امکانات بیشتر** را نیز اضافه می‌کنیم. در واقع این دو مولفه را به عنوان **جانگهدار** برای قسمت‌هایی که ممکن است در تحلیل دیده نشده باشند به مولفه‌های پروژه افزودیم.



نمودار ۲-۵. تصویر کلی از مولفه های اصلی پروژه

پس از پیاده‌سازی، در نمونه آزمایشی پروژه، مولفه تدارکات شامل بخش‌های مربوط به **خانه** و **تعیین گرایش و تمرکز** شد و مولفه امکانات بیشتر نیز شامل بخش‌های **نمایش و افزودن اطلاعات بیشتر به برنامه های پیشنهادی و ذخیره سازی آنها** و **تنظیمات** شد.

برای قسمت‌هایی از پروژه مثلا برای تبدیل فایل‌های برنامه دروس ارائه شده در ترم به فایل قابل پردازش برای نرم‌افزار به مولفه‌ای مستقل از سایر بخش‌ها نیاز داریم که خدمات مربوط به فایل‌های مورد نیاز نرم‌افزار را فراهم کند. این مولفه را در عمل پیاده‌سازی می‌کنیم اما آنرا در شکل‌ها نشان نمی‌دهیم چرا که همان‌طور که گفتیم این بخش از سایر بخش‌ها مستقل پیاده‌سازی می‌شود. این مولفه را **فراهم کننده خدمات فایل‌ها** می‌نامیم.

۲-۵ خلاصه

در این فصل نتایج بدست آمده از طراحی پروژه را تشریح کردیم. در ابتدا سعی کردیم تاثیر و عملکرد الگوریتم پیشنهاد برنامه ترمی را شناسایی کنیم. در همین میان با ایده‌ای به نام کاستن آشنا شدیم و بعد از آن بخش قابل توجهی از این فصل را به تشریح این ایده پرداختیم و در پایان این فصل نیز مولفه‌های دیگری از پروژه را شناسایی کرده‌ایم و در نهایت مولفه‌های پروژه را سازماندهی کردیم.

فصل سوم:
ساختارها، الگوریتم‌ها و روش‌های اصلی حل مسئله

۳-۱ مقدمه

در فصل قبل نتایج حاصل از تحلیل و طراحی پروژه را تشریح کردیم. در این فصل به مدل‌سازی داده‌ها و ارائه الگوریتم‌ها و روش‌های به‌کار گرفته شده برای حل مسائل اصلی پیش رو، می‌پردازیم.

۳-۲ ساختار توصیف کننده برنامه درسی مصوب وزارت علوم

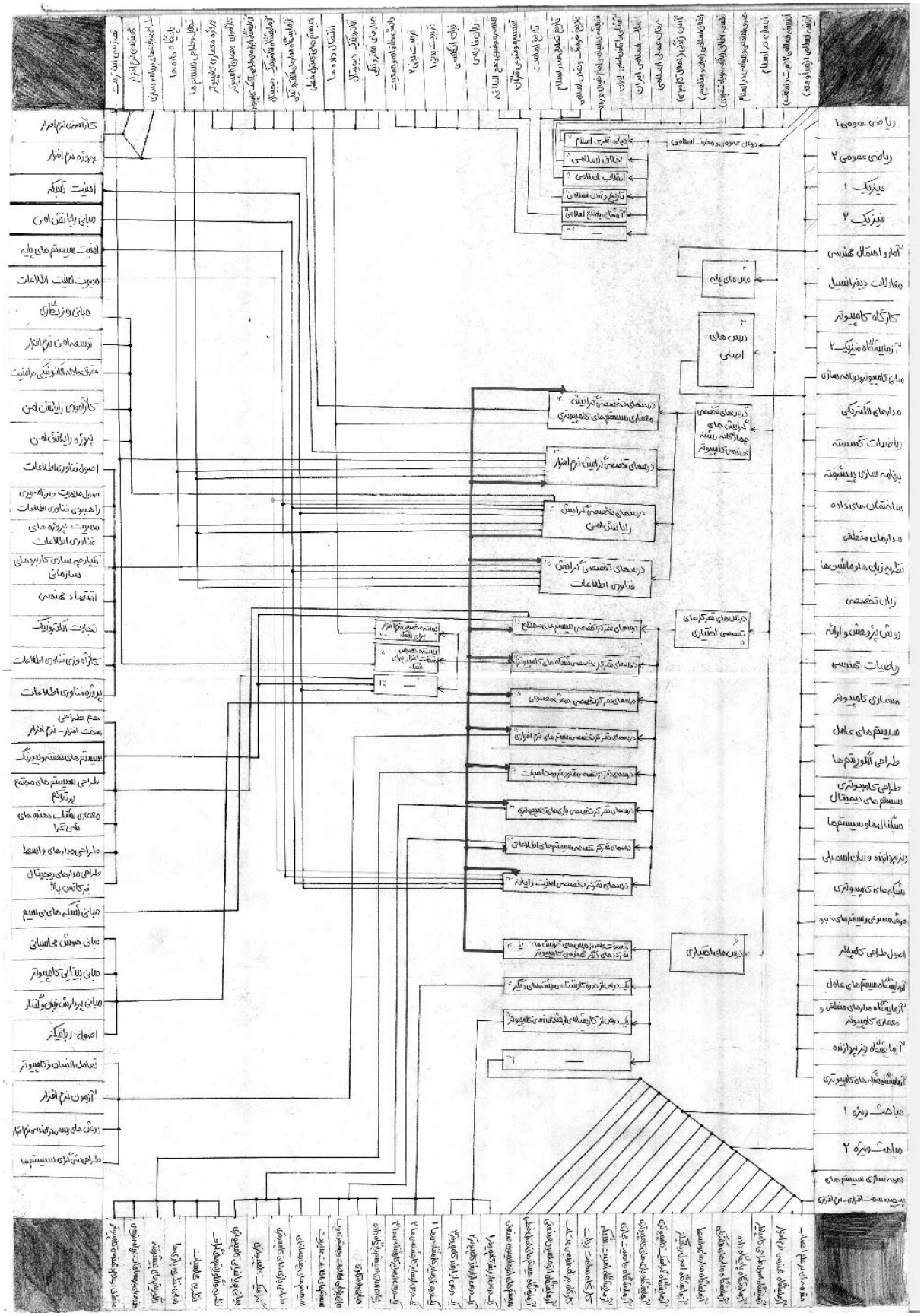
یکی از مهمترین ورودی‌های مسئله ما برنامه درسی مصوب وزارت علوم است که با دانشجو در ارتباط است. این ارتباط ممکن است براساس مواردی از قبیل رشته تحصیلی، گرایش، سال ورود به دانشگاه و ... تعیین شود برای مثال دانشجوی مهندسی کامپیوتر ورودی ۹۳ باید از برنامه درسی مصوب سال ۹۲ وزارت علوم پیروی کند در حالی که دانشجوی همین رشته اما ورودی ۹۷، با برنامه درسی مصوب سال ۹۶ آن، ارتباط دارد. تصمیم مهمی که باید می‌گرفتیم درباره چگونه مدل‌سازی کردن برنامه درسی مصوب وزارت علوم بود. در ادامه به تشریح مدل‌سازی خود می‌پردازیم.

برنامه‌های درسی مصوب وزارت علوم شامل دو مولفه **درس و محدودیت** می‌باشند. درواقع هر برنامه درسی مصوب وزارت علوم شامل تعدادی درس است که وزارت علوم با محدودیت‌هایی که در برنامه درسی مصوب خود اعمال می‌کند شرایط اخذ هر درس را مشخص می‌کند. نکته دیگر اینکه این محدودیت‌ها انواع شناخته شده و مشخصی دارند که عبارتند از: پیشنیازی و هم‌نیازی دروس، حداقل ترم برای اخذ درس، حداقل تعداد واحد برای اخذ درس، حداقل و حداکثر واحد مجاز برای اخذ درس‌های موجود در یک گروه از درس‌ها یا انواعی از درس‌های یک گروه(مثلا از درس‌های پایه در برنامه مصوب ۹۲ کامپیوتر باید ۲۰ واحد گذرانده شود یا به عبارت دیگر حداقل و حداکثر ۲۰ واحد)، به علاوه نوع دیگری از محدودیت‌ها که روی ارتباط دانشجو با برنامه درسی مصوب وزارت علوم اعمال می‌شود (مثلا برای دانشجو کامپیوتر ورودی ۹۳ این نوع از محدودیت را می‌توان، آن محدودیت‌هایی برای اخذ درس‌ها درنظر گرفت که به واسطه گرایش و تمرکز تخصصی دانشجو مشخص می‌شوند). می‌توان سه محدودیت اول نام‌برده شده را به عنوان خصیصه‌های یک درس در نظر گرفت. به این ترتیب مدل ما باید توانایی این را داشته باشد که سایر محدودیت‌های تعریف نشده را پشتیبانی کند و همچنین با محدودیت‌هایی که به عنوان خصیصه به درس‌ها انتساب داده شده‌اند سازگار باشد. با این اوصاف برای ساختار خود عناصری تعریف کردیم که با استفاده از آنها بتوان برنامه مصوب وزارت علوم را به شکل یا شکل‌هایی توصیف کرد.این عناصر عبارتند از:

- درس -> یک درس منبعی است که می‌تواند به یک یا چند گروه وصل باشد(ارتباط داشته باشد).
- گروه یا دسته درسی -> یک گروه یا دسته درسی، عنصری است که می‌تواند با دسته‌های درسی دیگر از طریق گیت‌های ورودی یا خروجی ارتباط داشته باشد یا با درس‌های دیگر نیز ارتباط داشته باشد.
- کارت اعتباری -> ساختاری است که از آن برای تعیین اعتبار و تشخیص سطح دسترسی استفاده می‌شود.

- محدودیت -> ساختاری است که با آن می‌توان محدودیت‌های حداقل و حداکثر واحد مجاز برای اخذ درس‌های موجود در یک گروه از درس‌ها یا انواعی از درس‌های یک گروه را توصیف کرد.
- گواهی‌نامه -> ساختاری است که می‌تواند شامل تعدادی محدودیت به علاوه سطوح دسترسی معین که به کمک کارت‌های اعتباری قابل بیان است، باشد.
- گیت -> ساختاری است که می‌تواند تعدادی گواهی‌نامه را بر روی دسته درسی مبداای به دسته درسی مقصدی مشخص کند.
- گیت ورودی و گیت خروجی -> گیت ورودی و گیت خروجی، مفاهیمی هستند که با توجه به جهت قراردادی، در حرکت از یک دسته درسی به دسته درسی دیگر شکل می‌گیرند. برای مثال فرض کنید بین دسته درسی الف و ب از طریق یک گیت، ارتباطی برقرار باشد و جهت قراردادی نیز، از الف به ب باشد در این صورت مرجع گیت در الف گیت خروجی و مرجع گیت در ب گیت ورودی است.
- گروه یا دسته درسی فقط خروجی -> دسته درسی‌ای که همه گیت‌های آن با توجه به جهت قراردادی، خروجی باشند را گروه یا دسته درسی فقط خروجی می‌نامیم.
- گروه یا دسته درسی فقط ورودی -> دسته درسی‌ای که همه گیت‌های آن با توجه به جهت قراردادی، ورودی باشند را گروه یا دسته درسی فقط ورودی می‌نامیم.
- گروه یا دسته درسی هم ورودی هم خروجی -> دسته درسی‌ای که با توجه به جهت قراردادی هم گیت‌های ورودی و هم گیت‌های خروجی داشته باشد را گروه یا دسته درسی هم ورودی هم خروجی می‌نامیم.(در ساختار ما این نوعِ گروه نباید با هیچ درسی ارتباط باشد)
- ریشه -> گروه یا دسته درسی فقط خروجی را که ارجاع به هیچ درسی نداشته باشد ریشه می‌نامیم.(هر ساختار توصیفی ما باید دقیقا یک ریشه داشته باشد.)
- گروه یا دسته درسی سطح اول -> گروه‌ها یا دسته‌های درسی‌ای که با ریشه ارتباط داشته باشند را گروه‌ها یا دسته‌های درسی سطح اول می‌نامیم.
- نود پایانی یا لبه -> گروه‌ها یا دسته‌های درسی فقط ورودی را که با حداقل یک درس در ارتباط باشند، نود پایانی، گره پایانی یا لبه می‌نامیم.

شاید درک این عناصر از روی توضیحات بالا کار دشواری باشد. در ادامه، کار خود را با یک مثال عملی ادامه می‌دهیم.



تصویر ۳-۱ساختار توصیف کننده برنامه درسی وزارت علوم مصوب سال ۹۲ برای رشته مهندسی کامپیوتر

تصویر پیشین، تصویری است از ساختار توصیف کننده برنامه درسی مصوب سال ۹۲ وزارت علوم برای رشته مهندسی کامپیوتر، البته در این شکل همه عناصر ساختاری که پیشتر معرفی کرده بودیم نمایش داده نشده‌اند برای مثال در این تصویر عناصر مربوط به گواهی‌نامه نمایش داده نشده است.

در چهارگوشه تصویر عناصر **درس** قرار گرفته‌اند که هر کدام از روی عنوان نوشته شده روی آنها قابل تشخیص می‌باشند.

جهت قراردادی ما از سمت **راست به چپ** می‌باشد و بنابراین جعبه سیاه رنگی که بالا و سمت راست تصویر قرار دارد و گروهی است فقط خروجی، نمایانگر **ریشه** می‌باشد.(سایر جعبه‌های سیاه صرفا جنبه نمایشی دارند)

جعبه‌هایی که در وسط صفحه قرار گرفته‌اند(یعنی همه جعبه‌ها به جز درس‌ها و سه جعبه تیره‌رنگ صرفا نمایشی)، **گروه**ها هستند.

گروه‌های دروس عمومی و معارف، درس‌های تخصصی گرایش‌های چهارگانه رشته مهندسی کامپیوتر، درس‌های تمرکز تخصصی اختیاری و دروس اختیاری، درس‌های تمرکز تخصصی شبکه‌های کامپیوتری و تا هشت واحد از درس‌های گرایش‌ها یا تمرکزهای دیگر مهندسی کامپیوتر **گروه‌های هم ورودی هم خروجی** هستند.

گروه‌های دروس عمومی و معارف، درس‌های پایه، درس‌های اصلی، درس‌های تخصصی گرایش‌های چهارگانه رشته مهندسی کامپیوتر، درس‌های تمرکز تخصصی اختیاری و دروس اختیاری گروه‌های **سطح اول** هستند.

گروه‌های درس‌های پایه، درس‌های اصلی، مبانی نظری اسلام، اخلاق اسلامی، انقلاب اسلامی، تاریخ و تمدن اسلامی، آشنایی با منابع اسلامی، خط تیره دروس عمومی و معارف اسلامی، درس‌های تخصصی گرایش معماری سیستم‌های کامپیوتری، درس‌های تخصصی گرایش نرم‌افزار، درس‌های تخصصی گرایش رایانش امن، درس‌های تخصصی گرایش فناوری اطلاعات، درس‌های تمرکز تخصصی سیستم‌های مجتمع، درس‌های تمرکز تخصصی هوش مصنوعی، درس‌های تمرکز تخصصی سیستم‌های نرم‌افزاری، درس‌های تمرکز تخصصی الگوریتم و محاسبات، درس‌های تمرکز تخصصی بازی‌های کامپیوتری، درس‌های تمرکز تخصصی سیستم‌های اطلاعاتی، درس‌های تمرکز تخصصی امنیت رایانه، یک درس از دوره کارشناسی رشته‌های دیگر، یک درس از کارشناسی ارشد مهندسی کامپیوتر، خط تیره دروس اختیاری، بسته مخصوص نرم‌افزار برای شبکه، بسته مخصوص سخت‌افزار برای شبکه و خط تیره درس‌های تمرکز تخصصی شبکه‌های کامپیوتری گروه‌های فقط ورودی هستند و **لبه**های این ساختار را تشکیل می‌دهند.

همانطور که قبل‌تر نیز اشاره کردیم در این تصویر همه‌عناصر نمایش داده‌نشده‌اند اما برای آنکه یک دید کلی از اینکه این عناصر چگونه و کجا هستند پیدا کنیم در این رابطه چند مثال ارائه می‌دهیم- یالی که با گیت ورودی ریشه به گیت خروجی دروس اصلی مشخص شده را در نظر بگیرید، بر روی این یال گواهی‌نامه‌ای قرار دارد که طبق آن حداقل و حداکثر ۵۹ واحد درسی می‌تواند از آن عبور کند(محدودیت) و اخذ درس‌های آن برای همه دانشجویان مجاز می‌باشد(کارت اعتباری) در واقع فرض کنید یک دانشجو کامپیوتر که با این برنامه درسی ارتباط دارد با گرایش و تمرکز

مخصوص به خود از ریشه شروع به حرکت به سمت لبه ها که منابع(درس های مختلف) آنجا قرار دارند می‌کند، در این مسیر دانشجو باید بتواند از یال‌ها بدون نقض کردن شرایط آنها عبور کند، مثلا اگر دانشجو تا به حال ۳واحد از درس‌های اصلی را گذرانده باشد می‌تواند از ریشه به لبه‌ی درس‌های اصلی برود و از آنجا برای خود یکی از دروسی که مجاز به اخذ آنها است را انتخاب کند- به عنوان مثالی دانشجویی دیگر با گرایش مهندسی نرم‌افزار که شش واحد از درس‌های تخصصی و دو واحد آزمایشگاه از دروس اختیاری خود را گذرانده است در نظر بگیرید. در این صورت این دانشجو اگر از مسیر ریشه به درس‌های تخصصی گرایش‌های چهارگانه مهندسی کامپیوتر حرکت کند، فقط مجاز به اخذ درس از درس‌های تخصصی گرایش نرم‌افزار می‌شود چرا که مجوز دسترسی به سه گروه تمرکز تخصصی دیگر را نداشته، تنها مجوز در این مسیر برای او مربوط به درس‌های تمرکز تخصصی گرایش نرم‌افزار می‌باشد و از آنجا که تا به حال شش واحد از دروس این گروه را گذرانده بدون نقض کردن محدودیت‌ها می‌تواند به این گروه رسیده از آن درس یا درس‌هایی مجاز برای خود اخذ کند. این در حالی است که همین دانشجو اگر بخواهد از ریشه به دروس اختیاری و سپس به دروس تخصصی گرایش نرم‌افزار برسد مجوز لازم برای اینکار را نداشته ولی مجوز دسترسی به دروس تخصصی سایر گرایش‌ها را دارد که اگر بتواند از میان درس‌های مجاز درسی را بدون نقض کردن محدودیت‌های موجود در طول مسیر بیابد می‌تواند آنرا اخذ کند. برای مثال اگر دانشجو درس مدارهای الکترونیکی را نگذرانده باشد و مجاز به اخذ این درس هم باشد(مثلا پیشنیازی همنیازی برقرار باشد)، می‌تواند این درس را اخذ کند چرا که هیچ محدودیتی را در طول مسیر نقض نمی‌کند؛ حالا فرض کنید این دانشجو درس آزمایشگاه الکترونیک دیجیتال را نگذرانده باشد و مجاز به اخذ آن نیز باشد، دراینصورت دانشجو نمی‌تواند این درس را اخذ کند چرا که پیشتر گفته بودیم دانشجو دو واحد آزمایشگاه از دروس اختیاری خود را گذرانده است و اگر بخواهد این درس را اخذ کند موجب نقض محدودیت در طول مسیر می‌شود(برای مثال روی یال ریشه به دروس اختیاری محدودیت حداقل و حداکثر دو واحد آزمایشگاه قرار داده شده است).

۱-۲-۳ نقص‌ها و چالش‌ها

ما در پیاده‌سازی خود از همین ساختار برای توصیف برنامه درسی مصوب وزارت علوم استفاده کرده‌ایم اما این ساختار در حالت کلی ضعف‌هایی دارد و همچنین با چالش‌هایی مواجه است.در ادامه به تشریح این موارد می‌پردازیم.

در برنامه درسی مصوب سال ۹۲ وزارت علوم برای رشته کامپیوتر، ایراداتی وجود داشت مثلا در جایی نام درس، گسسته و در جایی دیگر از نام ساختمان گسسته استفاده شده است در حالی که منظور از هر دو همان گسسته بود. هنگامی که من این برنامه را بررسی می‌کردم مشاهده کردم پیشنیازها یا همنیازهای یک درس در جاهای مختلف، متفاوت است برای مثال، در این برنامه درسی برای درس پایگاه داده‌ها در گروه تخصصی نرم‌افزار و همچنین رایانش امن، درس ساختمان‌های داده به عنوان پیشنیاز در نظر گرفته شده است در حالیکه برای همین درس در گروه تخصصی فناوری اطلاعات، درس تحلیل و طراحی سیستم‌ها به عنوان پیشنیاز ذکر شده است. همان‌طور که گفتیم این برنامه دارای ایراداتی است به همین خاطر، هنگام تحلیل این قسمت گمان کردم که این هم یک اشتباه از طرف وزارت علوم است و

مدل‌سازی خود را با همین فرض انجام دادم اما در حقیقت چنین موردی یک اشتباه از طرف وزارت علوم نبود؛ یعنی پیشنیازی و همنیازی یک درس از خصوصیات آن درس نمی‌باشد(که این همان کاری است که ما در تحلیل و پیاده‌سازی خود کردیم) بلکه پیشنیازی و همنیازی، در رابطه یک درس با یک گروه قابل تعریف است. بنابراین این یکی از ضعف‌های تحلیل و در پی آن پیاده‌سازی ما می‌باشد که برخی خصایص را به جای آنکه به رابطه بین درس و گروه اختصاص دهیم به اشتباه آنها را به عنوان خصایص خود درس در نظر گرفتیم. البته ما برای آنکه برنامه حتی در مواجه با چنین موردی به مشکل برنخورد راهکاری ارائه کردیم که در آینده درباره آن توضیح خواهیم داد اما به هر حال این یک ضعف مهم در تحلیل و سپس پیاده‌سازی ما به حساب می‌آید.

حال به یک چالش بسیار مهم می‌پردازیم، در ساختار فعلی ممکن است با یک کارت اعتباری مشخص بتوان به بعضی از درس‌ها از طریق چند گروه دسترسی پیدا کرد برای مثال یک دانشجوی کامپیوتر مرتبط با برنامه درسی مصوب سال ۹۲ وزارت علوم با گرایش مهندسی نرم‌افزار و تمرکز الگوریتم و محاسبات را در نظر بگیرید، این دانشجو از سه گروه درس‌های تخصصی نرم‌افزار، فناوری اطلاعات و رایانش امن به درس پایگاه داده‌ها دسترسی دارد، در اینگونه موارد باید مشکل را با **استنتاج منطقی** حل کرد. برای نمونه در مواجه با مثالی که بیان شد، می‌توانیم اینگونه استنتاج کنیم که دانشجو باید حداقل و حداکثر ۱۹ واحد از درس‌های تخصصی گرایش نرم‌افزار اخذ کند. اگر پایگاه داده را از این گروه اخذ نکند(یعنی اگر این درس را از یکی از دو گروه دیگر اخذ کند) محدودیت‌های این گروه دسترسی پذیر ارضا نخواهد شد، پس دانشجو باید این درس را از این گروه اخذ کند و نه گروه دیگری. اما مسئله همیشه به این راحتی نیست. در مثال قبل هم استنتاج ساده بود و هم نتیجه آن به یک امر اجباری منتهی شد یعنی دانشجو باید درس پایگاه داده را از گروه خاصی اخذ می‌کرد، اما می‌توان برنامه‌های درسی‌ای را متصور شد که استنتاج در آنها به نتیجه اختیاری (نه اجبار دانشجو) منتهی شود که در این صورت باید بدانیم یک درس را از چه مسیری اخذ می‌کنیم به علاوه اینکه استنتاج ممکن است بسیار پیچیده‌تر باشد مثلا با اخذ یک درس بدون اینکه هیچ محدودیتی در طول مسیر نقض شود دانشجو در شرایطی قرار بگیرد که امکان فارغ‌التحصیلی را از دست بدهد. با این اوصاف اگر بخواهیم مسئله را با همین مدل‌سازی برای حالت عمومی حل کنیم دست کم باید به پروژه خود یک **عامل استنتاج کننده** اضافه کنیم. اضافه کردن یک عامل استنتاج کننده برای من در این مرحله از نظر فنی کار آسانی نبود بعلاوه اینکه نمی‌دانستم، مسئله چقدر قرار است پیچیده‌تر شود. بنابراین راهکار دیگری برای مواجه با این چالش انتخاب کردم.

به جای افزودن عامل استنتاج کننده، به هنگام تعیین شدن کارت اعتباری دانشجو، فرآیندی به نام **اصلاح ساختار برنامه درسی** را معرفی کردم که در آن پس از تعیین شدن کارت اعتباری دانشجو، ساختار برنامه درسی با توجه به استنتاج‌های از قبل انجام شده توسط طراح، خود را به گونه‌ای تغییر می‌دهد که دانشجو به هر درس، حداکثر یک مسیر قابل دسترس داشته باشد. برای نمونه دانشجوی مثال قبل را با همان کارت اعتباری در نظر بگیرید، وقتی ساختار، کارت اعتباری دانشجو را دریافت کرد اتصال بین درس پایگاه داده‌ها و گروه‌های فناوری اطلاعات و رایانش امن را حذف می‌کند.

راهکاری که ارائه شد موجب می‌شود دامنه برنامه‌هایی که پروژه ما قادر به پشتیبانی از آنها است کاهش پیدا کند و در حالت کلی راهکار مطلوبی نیست با این حال باید در نظر داشت که برنامه‌های درسی فعلی وزارت علوم هم چندان

برنامه‌های پیچیده‌ای نیستند و همین راه‌کار می‌تواند در حال حاضر پاسخگوی بسیاری از نیازهای ما باشد.

۳-۳ الگوریتم گام کاستن ۱

در فصل قبل با گام کاستن ۱ آشنا شدیم. حالا بعد از آشنا شدن با ساختار توصیف کننده برنامه درسی مصوب وزارت علوم می‌خواهیم الگوریتم گام کاستن ۱ را تشریح کنیم.

در گام کاستن ۱ ما به عنوان ورودی با یک ساختار برنامه درسی مصوب وزارت علوم که با توجه به جدول تطبیق دروس و با استفاده از تاریخچه دانشجو پر شده است مواجهیم و همچنین اطلاعات کارت اعتباری دانشجو و یک سری اطلاعات جانبی نیز در اختیار داریم.در این الگوریتم باید به ازای هر درس موجود در ساختار، بررسی کنیم که آیا پس از اخذ شدن آن درس، آیا مسیری دسترس پذیر از درس در حال بررسی به ریشه به گونه‌ای که در طول مسیر محدودیتی نقض نشود وجود دارد یا خیر؟ فقط در صورتی که پاسخ مثبت باشد، درس مورد بررسی را در لیست نجات ذخیره می‌کنیم. پس از اتمام بررسی برای همه درس‌ها، درس‌های موجود در لیست نجات را به عنوان خروجی الگوریتم برمی‌گردانیم.

```
List<Course> Reduce1(Curriculum curriculum,CreditCard studCredit,
                     int cntPassedUnits, int currentTermNumber)
{
            bool[] visited = new bool[curriculum.Courses.Count];
            bool[] achivable = new bool[visited.Length];

            List<Course> lst = new List<Course>();
            curriculum.Courses.ForEach(course =>
            {
                if (Acheivable(visited, achivable, curriculum,studCredit,
                    course, cntPassedUnits, currentTermNumber))
                {
                    lst.Add(course);
                }
            });
            return lst;
}

bool Acheivable(bool[] visited, bool[] achivable,
                        Curriculum curriculum,CreditCard studCredit,
                        Course c, int cntPassedUnits,
                        int currentTermNumber){
            //memoization :=> dynamic programming
            if (visited[c.Id])
            {
                return achivable[c.Id];
            }


            if (!c.IsAvailable())
            {
                return false;
            }
            else if (c.IsPassed)
            {
                return false;
            }
            else if (c.MinRequireTerm > currentTermNumber)
            {
                return false;
            }
```

```
            else if (c.MinReuireUnits > cntPassedUnits)
            {
                return false;
            }

            for (int i = 0; i < c.PrerequisiteCourses.Count; i++)
            {
                var pr = c.PrerequisiteCourses[i];
                if (!pr.IsPassed)
                {
                    if (pr.NumberOfFailed > 1)
                    {
                        if (!Acheivable(visited, achivable, curriculum,
                            studCredit, pr, cntPassedUnits,
                            currentTermNumber))
                            return false;
                    }
                    else
                    {
                        return false;
                    }

                }

            }

            for (int i = 0; i < c.RequisiteCourses.Count; i++)
            {
                var r = c.RequisiteCourses[i];

                if (!r.IsAvailable())
                    return false;
                if (r.IsPassed == false && !Acheivable(visited, achivable,
                    curriculum, studCredit, r, cntPassedUnits,
                    currentTermNumber))
                    return false;

            }



            var res = if exists a path from c to root then return true
                      else return false:inputs(curriculum,studCredit,c);

            //memoization
            visited[c.Id] = true;
            achivable[c.Id] = res;

            return res;
        }
```

۴-۳ الگوریتم مرحله ۱ از گام کاستن ۲

در گام کاستن ۱ تعدادی از درس‌ها **نجات پیدا کردند** اما ممکن است همه‌ی این درس‌های نجات پیدا کرده در ترم جاری توسط دانشگاه ارائه نشده باشند. در این مرحله لیستی از دورس نجات پیدا کرده از مرحله قبل را به همراه برنامه دروس ارائه شده توسط دانشگاه در ترم جاری و به همراه اطلاعات جانبی به الگوریتم ارسال می‌کنیم. سپس الگوریتم سطرهایی از سطرهای ارائه شده از جدول برنامه دروس ارائه شده توسط دانشگاه را که درس متناظر آن در لیست دروس نجات یافته از مرحله قبل موجود باشد، برای این مرحله در لیست نجات مخصوص خود ذخیره می‌کند و در پایان، الگوریتم همه سطرهای نجات پیدا کرده را برمی‌گرداند.

توجه به این نکته نیز ممکن است خالی از لطف نباشد که ما در کدها یا شبه‌کدهایی که می‌آوریم هم برای مفهوم

لیست و هم برای مفهوم مجموعه از نمایش List<T> کمک می‌گیریم. در واقع مجموعه را لیستی با عناصر متمایز در نظر می‌گیریم که تعریف درستی نیست اما همین برای کار ما کفایت می‌کند.

```
List<OfferedCoursesRow> Reduce2(List<Course> courses, List<OfferedCoursesRow>
offeredCoursesRows, Gender studGender
,bool capacityFiltering=false)
{
   var groups = from r in offeredCoursesRows
                    where ((r.Gender == studGender
                            || r.Gender == Gender.COEDUCATIONAL) &&
                           (capacityFiltering?CalculateCapacity(r):true))
                          group r by r.CodeInDesUni;
   var newLst = new List<OfferedCoursesRow>();
   foreach (var group in groups)
   {
     var q = courses.FirstOrDefault(c => c.CodeInDesUni == group.Key);
     if (q != null)
     {
        foreach (var row in group)
        {
          newLst.Add(row);
        }
     }
   }

   return newLst;
}
```

٤-٥ ساختارها و الگوریتم‌های اصلی مرحله۲ از گام کاستن۲

بعد از مرحله۱ از گام کاستن۲ ما با تعدادی سطر از جدول دروس ارائه شده در ترم توسط دانشگاه مواجه‌ایم که هنوز احتمال انتخاب شدن برای آنها وجود دارد. در فصل قبل گفتیم در این مرحله می‌خواهیم با دریافت اولویت‌های کاربر سعی کنیم بازهم فضای مسئله را کاهش دهیم. حال در اینجا باید تصمیم بگیریم که چگونه می‌توانیم اولویت‌های فهرست شده را از کاربر دریافت کنیم و چگونه این کاهش را باید صورت داد؟! در ادامه به این موضوعات می‌پردازیم.

٤-٥-١ درس‌ها و سطرهای رنگی

برای آنکه کاربر را قادر سازیم تا بتواند اولویت‌های انتخاب شده در فصل قبل را بیان کند سه رنگ سبز، سفید و قرمز را در نظر می‌گیریم که هر درس و همچنین هر سطر می‌تواند یکی از این سه رنگ را داشته باشد. رنگ سبز برای یک سطر، به معنای این است که کاربر می‌خواهد آن را حتما اخذ نماید. رنگ سفید برای یک سطر به معنای این است که کاربر در رابطه با اخذ کردن یا اخذ نکردن آن سطر نظر قاطعی ندارد و رنگ قرمز به معنای این است که کاربر به هیچ وجه نمی‌خواهد سطر فعلی را اخذ نماید. به طور مشابه درس سبز به معنای این است که کاربر می‌خواهد درس فعلی حتما در برنامه ترم جاری‌اش وجود داشته باشد، درس سفید یعنی این که در این رابطه نظر قاطعی ندارد و درس قرمز هم یعنی این درس به هیچ عنوان در ترم جاری اخذ نشود. با استفاده از مفهوم درس‌ها و سطرهای رنگی می‌توان همه اولویت‌های انتخاب شده فصل قبل را توصیف کرد.

۲-۵-٤ الگوریتم‌های مرحله۲ از گام کاستن۲

در حین توصیف اولویت‌ها توسط کاربر، ما باید بررسی کنیم که اولویت‌های کاربر مجاز بوده و با هم در تناقص نباشند.در آخر پس از اتمام کار توصیف اولویت‌های کاربر نیز باید ورودی‌ها را برای الگوریتم مهیا کنیم. یکی از مهمترین بخش‌های این اعتبارسنجی این است که مجموعه درس‌های سبز انتخاب شده توسط کاربر باید ساختار را در وضعیت معتبر نگه دارد به علاوه اینکه باید حداقل یک برنامه ترمی مجاز و ممکن برای این مجموعه وجود داشته باشد. در این بخش به این موضوع که چطور می‌توانیم متوجه شویم پس از سبز کردن تعدادی درس آیا در وضعیت معتبر قرار داریم یا نه می‌پردازیم. توضیح اینکه چطور می‌توانیم متوجه شویم برای یک مجموعه درس، برنامه ترمی مجاز و ممکنی وجود دارد یا نه را تا رسیدن به بخش مربوط به الگوریتم پیشنهاد برنامه ترمی به تاخیر می‌اندازیم.

ما به عنوان ورودی، ساختار برنامه ترمی‌ای که با توجه به جدول تطبیق دروس و تاریخچه دانشجو پر شده است را به علاوه کارت اعتباری دانشجو و مجموعه دروس سبز رنگ درخواستی دانشجو دریافت می‌کنیم. برای آنکه بتوانیم تشخیص دهیم که آیا با اخذ کردن مجموعه دروس مشخص شده، ساختار در وضعیت معتبری است یا خیر کافی است بدانیم پس از اخذ همه این دروس، قانونی نقض شده است یا نه. اگر قانونی نقض نشده باشد یعنی در وضعیت معتبری قرار داریم و در واقع مجاز به اخذ آن مجموعه دروس می‌باشیم. اما در غیرآن صورت در وضعیت نامعتبر بوده و مجاز به اخذ آن مجموعه از دروس نمی‌باشیم.اما پیش از آنکه راهکاری برای این اعتبارسنجی پیشنهاد کنیم باید قوانین را بشناسیم و بدانیم چگونه باید آنها را بررسی کنیم.

قوانین را می‌توان از زوایای مختلف به شکل‌های گوناگون تقسیم بندی کرد ما در اینجا آنها را به دو نوع **قوانین پایه** و **قوانین ثانویه** تقسیم‌بندی کردیم. قوانین پایه، قوانینی هستند که نسبت به طول عمر برنامه درسی مصوب وزارت علوم ثابت‌اند مثلا چون در برنامه درسی مصوب سال ۹۲ وزارت علوم برای رشته مهندسی کامپیوتر، مفهومی مثل پیشنیازی و همینازی دروس وجود دارد و از آنجا که در این برنامه ذکر شده که: "این برنامه از تاریخ تصویب به مدت پنج سال قابل اجراء است و پس از آن نیازمند بازنگری است." ما انتظار داریم پیشنیازی و همنیازی و همچنین به صورت کلی قوانین مربوط به این برنامه در طول پنج سال عمر آن رعایت شده و ثابت بمانند. اما در کنار این قوانین، مواردی هستند که در طول عمر برنامه درسی مصوب وزارت علوم ممکن است تغییر کنند(برای مثالی از این قانون، مجاز بودن اخذ حداکثر یک درس معارف در یک ترم تحصیلی را در نظر بگیرید). پس ما نیاز است این دو نوع از قوانین را برای تشخیص معتبر بودن یا نبودن وضعیت پس از انتخاب درس‌های سبز بررسی کنیم. بررسی کردن قوانین ثانویه معمولا سریع‌تر انجام می‌شود از این جهت در بررسی‌های خود ترجیح دادیم ابتدا این قوانین را بررسی کنیم.

```
bool IsValidState(Curriculum curriculum,CreditCard studCredit, List<int> takenCoursesId)
{
  //secondary rules
  bool o = SecondaryRulesStateValidator.IsValidState(curriculum,
                                    studCredit, takenCoursesId);
  if (!o) return false;

  //basic rules
  return BasicRulesStateValidator.IsValidState(curriculum,studCredit,
                                                    takenCoursesId);
}
```

یکی از قوانینی که در قسمت اعتبارسنجی قوانین پایه باید صورت پذیرد، بررسی این است که آیا پس از اخذ مجموعه دروس سبز ورودی، محدودیت قابل دسترسی نقض می‌شود یا نه. برای بررسی این مورد شبه کدی ارائه نمی‌کنیم درعوض عملکرد کلی آن را شرح می‌دهیم. برای تشخیص این موضوع، ابتدا با استفاده از دروس سبز ورودی باید اطلاعات در ارتباط با ساختار توصیف کننده برنامه درسی مصوب وزارت علوم را بروزرسانی کرد سپس باید از ریشه شروع به جستجو کرده و از همه گروه‌های دسترسی پذیر عبور کرد، چنانچه در مسیر جستجو موردی مشاهده شد که دسترسی به آن مجاز بود اما محدودیتی نیز به واسطه آن دسترسی نقض می‌شود، یعنی وضعیت ساختار برنامه درسی نامعتبر است و اگر تا پایان جستجو چنین موردی یافت نشد یعنی در وضعیت معتبر قرار دارد.

٦-٤ الگوریتم پیشنهاد برنامه‌های ترمی

در این بخش می‌خواهیم شما را با الگوریتم اصلی برنامه که همان الگوریتم پیشنهاد برنامه(های) ترمی است آشنا کنیم. اما از آنجا که خود الگوریتم از زیرساخت‌ها و حل مسئله‌های دیگر بهره می‌برد ما ابتدا به این موضوعات پرداخته سپس به خود الگوریتم می‌پردازیم.

١-٦-٤ زیرساخت اول

ما پیشتر در این فصل هنگام مواجه شدن با مسئله‌ی اعتبارسنجی درس‌های سبز، گفتیم که درباره آن بعدا صحبت می‌کنیم؛ اکنون زمان آن رسیده است. جدول برنامه درسی ارائه شده در ترم را که توسط دانشگاه ارائه می‌شود به یاد بیاورید. گفته بودیم که این جدول شامل سطرهایی است که هر سطر آن نیز شامل ستون‌هایی است. به علاوه برخی از ستون‌ها را نام برده بودیم. یک ستون از ستون‌های هر سطر، مربوط به شماره و گروه درس(که از آن می‌توان کد درس را نیز بدست آورد) می‌باشد.

حال فرض کنید یک مجموعه از درس‌های سبز را بعنوان ورودی به شما می‌دهند و می‌پرسند با فرض این که اخذ این دروس، سایر قوانین را نقض نکرده باشد، همه برنامه‌های ترمی مجاز و ممکن را برای آن‌ها به عنوان خروجی برگردان.

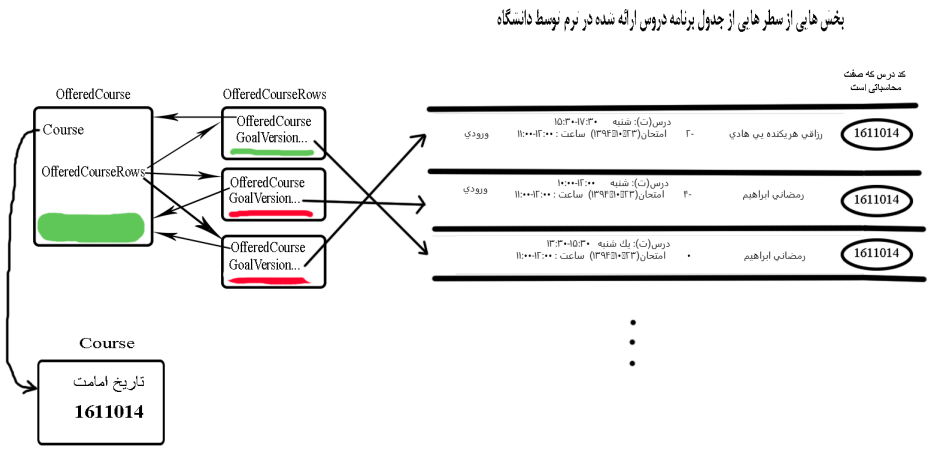
حال فرض کنید ما با پیش‌پردازش از یک نوع مدل‌سازی خاص استفاده کرده باشیم که در آن هر درس به همه سطرهای موجود در ترم آن دسترسی دارد(همه سطرهای جدول برنامه ترمی ارائه شده در ترم توسط دانشگاه که مربوط به آن درس است) و هر سطر هم به درسی که به آن ارجاع می‌کند، ارجاع دارد.

```
class OfferedCourse{
  Course Course;
  List<OfferedCourseRow> OfferedCourseRows;
  Color Color;
}
class OfferedCourseRow{
  OfferedCourse OfferedCourse;
  Main.OfferedCoursesRow GoalVersionOfferedCourseRow;
  Color Color;
}
```

در این مدل‌سازی OfferedCourse نماینده یک درس می‌باشد و دارای فیلدهای Course،OfferedCourseRows و Color است. Course نماینده یک درس می‌باشد(درسی مشابه آنچه از ساختار توصیف کننده برنامه درسی مصوب وزارت علوم می‌شناسیم)، OfferedCourseRows به سطرهایی ارجاع دارد که GoalVersionOfferedCourseRow آنها به سطری از جدول برنامه دروس ارائه شده در ترم توسط دانشگاه ارجاع داشته باشد که کد درس آن با کد درس Course در OfferedCourse یکسان باشد و Color همان سه رنگ سبز، سفید و قرمز می‌تواند باشد.

OfferedCourseRow نماینده یک سطر است که شامل OfferedCourse،GoalVersionOfferedCourseRow و Color می‌شود. خصوصیت OfferedCourse به نمونه‌ای از کلاس OfferedCourse که به این سطر ارجاع دارد اشاره می‌کند. GoalVersionOfferedCourseRow از نوع Main.OfferedCourseRow تعریف شده است. منظور ما از این نوع همان OfferedCourseRow ای است که در شبه‌کدهای بخش‌های قبلی استفاده کردیم که نماینده یک سطر از

جدول دروس ارائه شده در ترم توسط دانشگاه بود و بخاطر اینکه بتوانیم آنرا از نوع OfferedCourseRow ای که در اینجا تعریف کردیم تفکیک کنیم از این نوع نمایش استفاده شده است. Color نیز مانند قبل می‌تواند یکی از سه مقدار سبز، سفید و قرمز را داشته باشد.

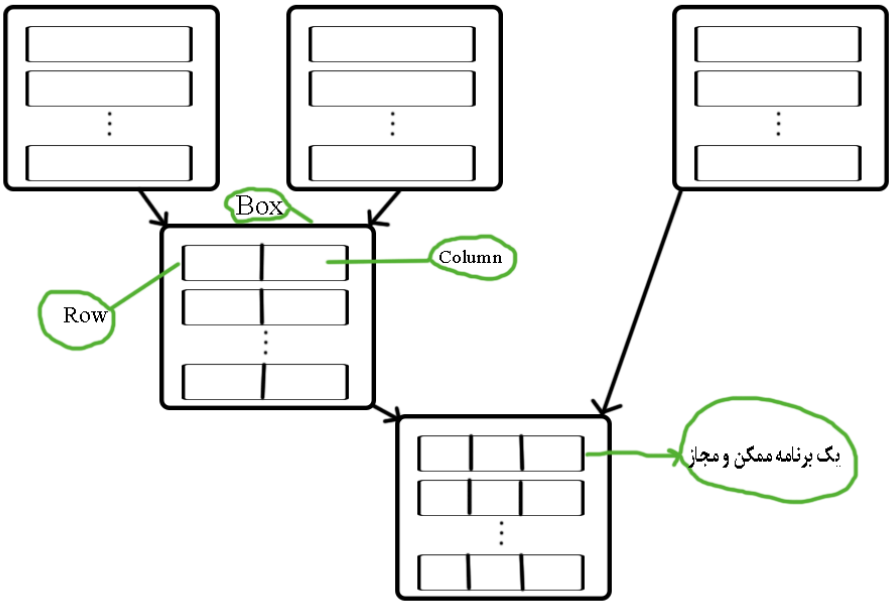


شکل ۲-۳ مدل گرافیکی نمونه‌ای از ارتباطات *Course*، *OfferedCourse*، *OfferedCourseRow* و *Main.OfferedCourseRow*

علاوه بر این ساختارها، دو ساختار دیگر نیز معرفی می‌کنیم که عبارتند از Box و Row و به صورت زیر قابل توصیف هستند:

```
class Row{
  List<OfferedCourseRow> Columns;
}
class Box{
  List<Row> Rows;
}
```

ما به ازای هر درس از مجموعه دروس ورودی یک Box می‌سازیم و سپس به صورت درخت مانند، هر مرحله، دو به دو همه برنامه‌های ممکن و مجاز؛ یعنی آن Boxهایی که Columnهای آنها با هم تداخل زمانی ندارند را محاسبه می‌کنیم. این کار را تا بررسی آخرین Box با تعداد Column برابر یک ادامه می‌دهیم. اگر در طول محاسبات مرحله‌ای، به Box ای برخوردیم که طول Rows آن پس از انجام محاسبه صفر بود یعنی برنامه ممکن و مجازی برای این مجموعه دروس وجود ندارد و اگر هرگز به چنین Boxای برنخوردیم، آخرین Boxای که محاسبه می‌کنیم شامل تمام برنامه‌های مجاز و ممکن خواهد بود.



شکل ۳-۳ شیوه محاسبه کردن تمام برنامه‌های ممکن و مجاز برای یک مجموعه درس ورودی سه تایی

```
Box CreateBoxForOfferedCourse(OfferedCourse offeredCourse)
{
   Box box = new Box();

   offeredCourse.OfferedCourseRows.ForEach(r =>
   {
     if (r.Color != RED)
     {
        Row row = new Row();
        row.Columns.Add(r);
        box.Rows.Add(row);
     }
   });

   return box;
}
```

تابع CreateBoxForOfferedCourse یکی از دروس موجود در مجموعه دروس ورودی را بعنوان آرگومان گرفته Box متناظر با آن را ایجاد کرده، برمی‌گرداند.

```
List<Box> Validate(List<Box> boxes, bool examCollideCheck = false)
{
    if (boxes.Count == 1 && boxes[0].Rows.Count == 0)
       return null;
    List<Box> bxs = new List<Box>();
    bxs.Add(boxes[0]);

    bool con0 = true;
    for (int i = 1; i < boxes.Count; i++)
    {
      var b1 = bxs[0];
      var b2 = boxes[i];

      Box newBox = new Box();

      for (int j = 0; j < b2.Rows.Count; j++)
```

```
      {
        for (int k = 0; k < b1.Rows.Count; k++)
        {
           bool con = true;

           Row newRow = new Row();


           for (int m = 0; m < b1.Rows[k].Columns.Count; m++)
           {
             bool collide = DoTheyCollide(b1.Rows[k].Columns[m]
                                            .GoalVersionOfferedCourseRow
                                             .TimeAndSitesAndExam
                                              .TimeAndSites,
                                         b2.Rows[j].Columns[0]
                                            .GoalVersionOfferedCourseRow
                                             .TimeAndSitesAndExam
                                              .TimeAndSites);

             if (collide){
                con = false;
                break;
              }

             if (examCollideCheck &&
                 DoTheyExamCollide(b1.Rows[k].Columns[m]
                                     .GoalVersionOfferedCourseRow
                                       .TimeAndSitesAndExam.Exam,
                                   b2.Rows[j].Columns[0]
                                     .GoalVersionOfferedCourseRow
                                        .TimeAndSitesAndExam.Exam)){
                   con = false;
                   break;
              }

             newRow.Columns.Add(b1.Rows[k].Columns[m]);
             }

             newRow.Columns.Add(b2.Rows[j].Columns[0]);

             if (con) newBox.Rows.Add(newRow);
          }
        }

        if (newBox.Rows.Count == 0){
          con0 = false;
          break;
        }
        else bxs[0] = newBox;
      }

      if (con0) return bxs;
      else return null;

}
```

تابع Validate به عنوان آرگومان یک لیست از Boxها را گرفته، با بررسی دو به دو آنها، در آخر اگر برنامه(های) ممکن و مجازی وجود داشته باشد، همه برنامه‌های ممکن و مجاز را محاسبه کرده و برمی‌گرداند و درغیراینصورت null خروجی تابع خواهد بود.

```
Validate(List<OfferedCourse> offeredCoursesList, bool examCollideCheck = false)
{
  List<Box> lst = new List<Box>();
  for (int i = 0; i < offeredCoursesList.Count; i++)
    lst.Add(CreateBoxForOfferedCourse(offeredCoursesList[i]));
  return Validate(lst, examCollideCheck);
}
```

این تابع Validate، مجموعه دروسی که می‌خواستیم از اول این بحث، برنامه‌های ممکن و مجاز را برایشان محاسبه

کنیم به عنوان ورودی گرفته Boxهای متناظر آنها را ایجاد کرده در لیستی ذخیره می‌کند سپس با فراخوانی تابع Validateای که قبل‌تر بررسی‌اش کردیم که لیستی از Boxها را به عنوان ورودی می‌گیرد، همه برنامه‌های ممکن و مجاز را در صورتی که حداقل یک برنامه ممکن و مجاز وجود داشته باشد برمی‌گرداند و چنانچه چنین برنامه‌ای وجود نداشته باشد null به عنوان خروجی تابع در نظر گرفته می‌شود.

۲-٦-٤ بخش اصلی الگوریتم

الگوریتم پیشنهاد برنامه(های) ترمی به عنوان ورودی به ترتیب لیستی از امتیاز دروس،لیستی از درس‌های سفید، لیستی از درس‌های سبز، حداقل واحد مجاز برای برنامه ترمی، حداکثر واحد مجاز برای برنامه ترمی، مجموع تعداد واحدهای درس‌های سبز، مجموع امتیازات دروس سفید، ساختار توصیف کننده برنامه ترمی مصوب وزارت علوم که با اطلاعات مناسب پر شده است، کارت اعتباری دانشجو، همه برنامه‌های ممکن و مجاز برای دروس سبز، حالت بررسی تصادم امتحانات و تایم اجرا الگوریتم را به عنوان ورودی دریافت می‌کند.

الگوریتم ابتدا از میان درس‌های سفید ورودی، به صورت تصادفی با احتمال حاصل از محاسبه **امتیاز درس ضرب در صد تقسیم بر مجموع امتیازات دروس سفید**، درس‌هایی را جهت اخذ شدن انتخاب می‌کند. اگر مجموع واحدهای دروس انتخاب شده در این مرحله به علاوه مجموع واحدهای دروس سبز، در محدوده مجاز حداقل و حداکثر واحد مجاز یک برنامه ترمی قرار نداشت این مرحله تکرار می‌شود. در غیر اینصورت الگوریتم بررسی می‌کند آیا با اخذ مجموعه دروسی شامل درس‌های سبز و درس‌های سفیدِ انتخابیِ فعلی، آیا ساختار در وضعیت معتبری قرار دارد یا خیر؟ اگر پاسخ منفی باشد الگوریتم مجدداً باید دنبال مجموعه دروس سفید جدیدی برای بررسی بگردد اما اگر پاسخ مثبت بود الگوریتم همه برنامه‌های ممکن و مجاز از مجموعه شامل دروس سبز و سفید انتخابی فعلی را در صورتی که وجود داشته باشند، می‌یابد. پس از آن هر کدام از این برنامه‌ها به نمونه‌ای به نام `_ChoosedWeeklyProgramManager`، درخواستی برای قرار گرفتن در لیست نهایی می‌دهند. `_ChoosedWeeklyProgramManager` وظیفه دارد برای هر برنامه‌درخواستی ارزشی براساس تابع ارزیابی در نظر بگیرد. به علاوه وظیفه دیگر آن گزینش تعداد مشخصی از بهترین برنامه‌های غیر تکراری یافته شده است. الگوریتم همه این تکرارها را در مدت زمان مشخصی انجام می‌دهد و پس از آن، تعداد مشخصی از بهترین برنامه ترمی‌هایی را که یافته باز می‌گرداند.

```
Algo(double[] courseScore, List<OfferedCourse> whiteCourses, List<OfferedCourse> greenCourses,
int minUnits, int maxUnits, int greenCoursesUnits, double whiteCoursesTotalScore,Curriculum
curriculum, CreditCard studCredit,List<Box> GreenCoursesBoxes, bool examCollideCheck, int
timeout)
{
 var _ChoosedWeeklyProgramManager= new AlgorithmTopWeeklyProgramManager();

 Execute.For(timeout){

   Random random = new Random();
   int CurrentSelectedUnits=0;
   bool[] Selected;

   List<int> takenCoursesId = new List<int>();

   while (true){
    while (true){
     do{
        Selected = new bool[whiteCourses.Count];
```

```
        CurrentSelectedUnits = 0;
        takenCoursesId.Clear();

        //init and assigning values to Selected array
        for (int i = 0; i < Selected.Length; i++){

           var offeredCourse = whiteCourses[i];
           var course = whiteCourses[i].Course;

           var pc = random.Next(100);

           if (CurrentSelectedUnits + course.Units + greenCoursesUnits >
               maxUnits) continue;

           if (pc <= courseScore[course.Id] * 100.0 /
               whiteCoursesTotalScore){

                Selected[i] = true;
                CurrentSelectedUnits += course.Units;
                takenCoursesId.Add(course.Id);
           }
        }

      }while (minUnits > CurrentSelectedUnits + greenCoursesUnits);

      greenCourses.ForEach(gc => takenCoursesId.Insert(0, gc.Course.Id));
      //two step validation
      bool o = IsValidState(curriculum,studCredit,takenCoursesId);
      if (o) break;
    }

   List<Box> boxes = new List<Box>();
   GreenCoursesBoxes.ForEach(b => boxes.Add(b));
   for (int i = 0; i < Selected.Length; i++){
      if (Selected[i]){
         Box b = CreateBoxForOfferedCourse(whiteCourses[i]);
         boxes.Add(b);
      }
   }

   List<Box> res = Validate(boxes, examCollideCheck);

   if (res != null) _ChoosedWeeklyProgramManager.TryAddNewWeeklyProgram(res[0].Rows);

  }
 }
 return _ChoosedWeeklyProgramManager;
}
```

همان‌طور که دیدیم الگوریتم بالا ورودی‌های خاصی دارد که به نظر نمی‌رسد مهیا کردن همه آنها توسط مشتری(کسی که از الگوریتم استفاده می‌کند یا آن را فراخوانی می‌کند) به صورت مستقیم کار معقولی باشد. در واقع نیز قرار نیست همه این ورودی ها را مشتری تهیه کند بلکه وظیفه مشتری فقط تهیه ورودی‌های ضروری است، در این قسمت ما شبه‌کد دیگری ارائه می‌کنیم که در آن اطلاعات مورد نیاز برای الگوریتم، با توجه به ورودی‌های ضروری مهیا شده از طرف مشتری تولید می‌شود و سپس با فراخوانی آنچه در شبه‌کد قبلی دیدیم در یک بازه زمانی مشخص، سعی می‌کنیم تعدادی برنامه ترمی مناسب به کاربر پیشنهاد دهیم.

```
MainAlgo(List<OfferedCourse> inputs, Curriculum curriculum
         ,CreditCard studCredit, int minUnits, int maxUnits
         , bool examCollideCheck, int timeout){
  double[] courseScore = new double[curriculum.Courses.Count];

  double numberOfGreenUnits = 0;

  List<OfferedCourse> greenCourses = new List<OfferedCourse>();

  //white courses that have at least one white row
  List<OfferedCourse> whiteCourses = new List<OfferedCourse>();
```

```
  int greenUnits = 0;
  for (int i = 0; i < inputs.Count; i++){
     var offeredCourse = inputs[i];

     if (offeredCourse.Color == Green){
        greenCourses.Add(offeredCourse);
        greenUnits += offeredCourse.Course.Units;
     }
     else if (offeredCourse.Color == WHITE){
         for (int r = 0; r < offeredCourse.OfferedCourseRows.Count; r++){
            var offeredCourseRow = offeredCourse.OfferedCourseRows[r];
            if (offeredCourseRow.Color == WHITE){
              whiteCourses.Add(offeredCourse);
              break;
            }
         }
      }

      var course = offeredCourse.Course;

      courseScore[course.Id] += (course.Units * 2 - 1);

      for (int c = 0; c < course.PrerequisiteCourses.Count; c++){
         var preCourse = course.PrerequisiteCourses[c];

         if (!preCourse.IsPassed && preCourse.IsAvailable()){
           if (preCourse.NumberOfFailed > 1)
               courseScore[preCourse.Id] += (course.Units * 2 - 1) * 0.20;
           else
               courseScore[preCourse.Id] += (course.Units * 2 - 1) * 0.25;
         }
       }

       for (int c = 0; c < course.RequisiteCourses.Count; c++){
          var reqCourse = course.RequisiteCourses[c];

          if (!reqCourse.IsPassed && reqCourse.IsAvailable())
             courseScore[reqCourse.Id] += (course.Units * 2 - 1) * 0.20;
        }

   }

   double whiteCoursesTotalScore = 0;

   whiteCourses.ForEach(i =>{whiteCoursesTotalScore +=
                        courseScore[i.Course.Id];});

   List<Box> greenCoursesBoxes = new List<Box>();

   greenCourses.ForEach(gc =>{
                numberOfGreenUnits += gc.Course.Units;
                Box b = CreateBoxForOfferedCourse(gc);
                greenCoursesBoxes.Add(b);
   });
   if (greenCoursesBoxes.Count > 0){
      List<Box> res = Validate(greenCoursesBoxes, examCollideCheck);
      if (res != null){
         greenCoursesBoxes.Clear();
         greenCoursesBoxes.Add(res[0]);
      }
   }
  return  Algo(courseScore, whiteCourses, greenCourses, minUnits,
          maxUnits, greenUnits, whiteCoursesTotalScore, curriculum
          ,studCredit, greenCoursesBoxes, examCollideCheck,timeout);

}
```

امتیاز دادن به درس‌ها بستگی به شرایط مسئله یا نظر طراح، می‌تواند متفاوت باشد. ساده‌ترین راهکار این است که امتیاز همه درس‌ها را یکسان درنظر بگیریم در این صورت همه درس‌ها برای انتخاب شدن در برنامه شانس برابری

خواهند داشت اما بدون توضیح بیشتر این راهکار خوبی به نظر نمی‌رسد.در عوض استفاده از یک **تابع اکتشافی**، جذاب‌تر بنظر می‌رسد. تابع اکتشافی ما سعی می‌کند معیاری برای خوب بودن یک درس ارائه کرده و احتمال انتخاب شدن درس‌های بهتر را بیشتر کند و این درحالی است که برای درس‌های بدتر هم شانس انتخاب شدن قائل می‌شود. همانطور که از شبه‌کد هم قابل برداشت است ما امتیاز هر درس را به صورت زیر تعریف کردیم:

```
+ ( 1 - 2 * تعداد واحد درس )
( 0.25 * تعداد واحدهای دروس موجود پاس نشده‌ای که این درس در حال حاضر پیشنیاز آنها است )
( 0.20 * تعداد واحدهای دروس موجود پاس نشده‌ای که در حال حاضر این درس همنیاز آنها است ) +
```

در طراحی این تابع فرض ما این اینست که پیشنیازی و همنیازی دروس، قانونی پایه‌ای است که برای همه برنامه‌های مصوب وزارت‌علوم خواه برنامه‌های فعلی باشد خواه برنامه‌های آینده ثابت و موجود است و به همین دلیل وابسته کردن فرمول بالا به این قانون، ایرادی ندارد. به علاوه معیاری که ما برای امتیاز دهی در نظر گرفتیم مثل این است که بگوئیم **درسی بهتر است که تعداد واحدهای بیشتری داشته باشد و با اخذ یا گذراندن آن بتوان تعداد دروس بیشتری را قابل اخذ کرد**. اما همان‌طور که گفته بودیم این فرمول بسته به سلیقه طراح می‌تواند متفاوت باشد.

تا به این جا بخش‌های مهمی از الگوریتم را تشریح کردیم اما هنوز معیاری برای ارزش‌گذاری برنامه‌های ترمی ارائه نکردیم. در رابطه با این موضوع نیز باید گفت معیاری که برای این ارزش‌گذاری انتخاب می‌شود بسته به سلیقه طراح می‌تواند متفاوت باشد.

معیار اصلی‌ای که ما در اینجا از آن استفاده کردیم، حداقل فاصله بین کلاس‌ها است. البته این تنها معیاری نیست که در نظر گرفتیم چرا که در این صورت برنامه‌هایی که تعداد واحد بیشتری دارند در مقایسه با برنامه‌های با تعداد واحد کمتر تقریبا هیچ شانسی برای گزینش پیدا نمی‌کنند و این درحالی است که کاربر حداقل و حداکثر واحدی را برای برنامه‌ترمی پیشنهادی مشخص می‌کند. بنابراین معیاری که از آن استفاده کردیم را به صورت غیر رسمی می‌توان **حداقل فاصله بین کلاس‌ها به ازای هر واحد** در نظر گرفت.

```
CalculateWeeklyProgramValue(List<OfferedCourseRow> lst)
{
   double res = 0;
   int units = 0;

   var rows = new SortedList<int,TimeAndSite>();
   foreach (var item in lst){

    if (item.GoalVersionOfferedCourseRow.TimeAndSitesAndExam
         != null && item.GoalVersionOfferedCourseRow
           .TimeAndSitesAndExam.TimeAndSites != null){

            foreach (var timeAndSite in
                          item.GoalVersionOfferedCourseRow.
                            TimeAndSitesAndExam.TimeAndSites){

                   rows.Add((((int)timeAndSite.Day) * 60 * 24 +
                             timeAndSite.StartTime.Hour * 60
                             + timeAndSite.StartTime.Minute)
                             , timeAndSite);
             }
    }
   units += item.OfferedCourse.Course.Units;
  }

  for (int i = 1; i < rows.Count; i++){
```

```
   int dt = (((int)rows.Values[i].Day) * 60 * 24 +
             rows.Values[i].StartTime.Hour * 60
             + rows.Values[i].StartTime.Minute) -
             (((int)rows.Values[i - 1].Day) * 60 * 24 +
             rows.Values[i - 1].FinishTime.Hour * 60
             + rows.Values[i - 1].FinishTime.Minute);
   res += dt;
  }
           return (res * -1.0 * 1.3) / units;
}
```

هر چه ارزش محاسبه شده برای یک برنامه ترمی کمتر منفی باشد آن برنامه بهتر است در واقع اینطور در نظر گرفتیم که بهترین حالت وقتی است که تمام وقت دانشجو در طول هفته برای خودش باشد.

۳-۷ ارائه ترتیبی قابل اخذ برای سطرهای یک برنامه ترمی

در فصل اول در رابطه با سیستم گلستان گفتیم، در حال حاضر در هنگام انتخاب واحد در این سیستم شما باید واحدها را به ترتیبی ثبت کنید که همواره برنامه ترمی شما در وضعیت معتبری قرار داشته باشد یعنی برای مثال جهت اخذ درسی که همنیازی دارد ابتدا باید همنیاز آن درس را وارد کرده سپس خود درس را وارد کنید. حالا که الگوریتم پیشنهاد برنامه ترمی را معرفی کردیم وقت آن رسیده بررسی کنیم چگونه می‌توانیم سطرهای یک برنامه ترمی را با ترتیبی قابل اخذ برای سیستم گلستان مرتب کنیم.

راه‌حل بسیار ساده است، تنها کافیست یک گراف بر مبنای پیشنیازی و همنیازی مجموعه دروس موجود در برنامه ترمی ایجاد کنید و سپس با اجرای الگوریتمی که مرتب سازی توپولوژیکی روی این گراف انجام دهد، شما به یک ترتیب قابل اخذ برای آن دروس دست خواهید یافت.

۳-۸ خلاصه

در این فصل بحث را با معرفی ساختار توصیف کننده برنامه درسی مصوب وزارت علوم شروع کردیم. پس از معرفی این ساختار نقض‌ها و چالش‌هایی که ساختار معرفی شده با آنها مواجه بود را بررسی کرده و خط مشی که در این خصوص اتخاذ کردیم را بیان نمودیم. در ادامه الگوریتم‌های مربوط به ایده کاستن را که در فصل قبل با آنها آشنا شدیم شرح دادیم و در طول این کار با مفاهیمی مانند درس‌ها و سطرهای رنگی آشنا شدیم. بعد از این مرحله به الگوریتم اصلی پروژه خود یعنی الگوریتم پیشنهاد برنامه‌های ترمی رسیدیم و پس از تشریح زیرساخت‌های آن به تشریح خود الگوریتم پرداختیم. در آخر نیز راهکاری برای دستیابی به ترتیبی قابل اخذ برای دروس یک برنامه ترمی پیشنهاد دادیم.

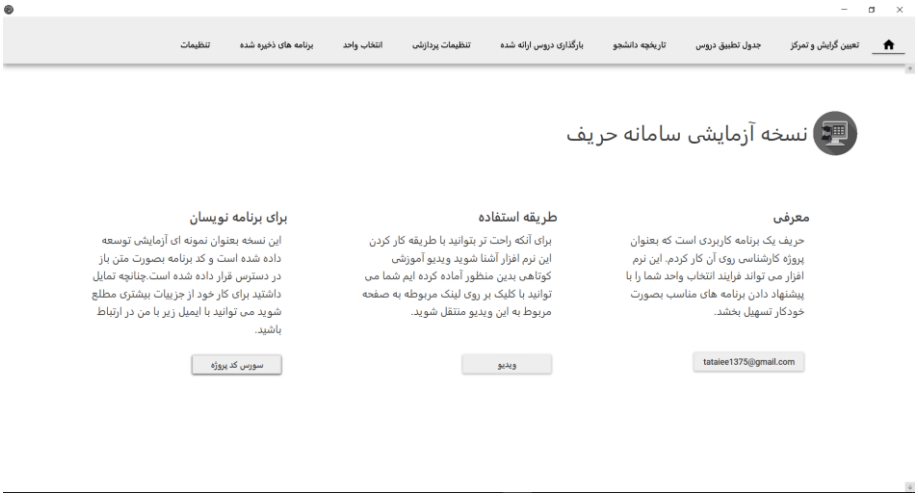
فصل چهارم:
نسخه آزمایشی و نتایج آزمایشات

٤-١ مقدمه

در این فصل ابتدا شما را با یک نسخه آزمایشی توسعه داده شده بر مبنای برنامه درسی مصوب سال ۹۲ وزارت علوم برای رشته مهندسی کامپیوتر آشنا خواهیم کرد. سپس با اشاره به تعدادی از برنامه‌های ترمی پیشنهادی توسط نسخه آزمایشی که آنها را در طول آشنایی با این نسخه، طرح می‌کنیم و برخی از نکات و توضیحات را در رابطه با نتایج حاصل از آزمایشات بیان می‌کنیم.

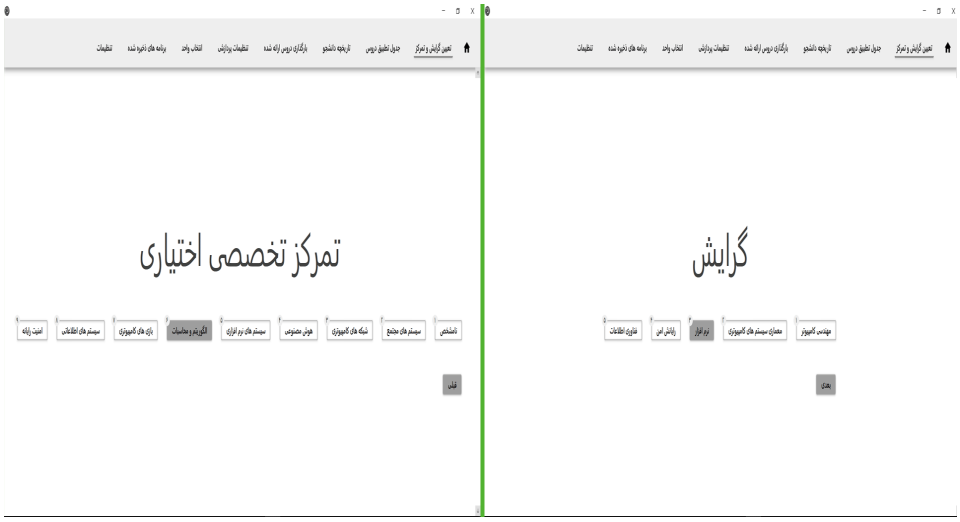
٤-٢ آشنایی با نسخه آزمایشی

نسخه آزمایشی، برنامه‌ای است که به عنوان نمونه با توجه به آنچه در فصل‌های قبل بیان کردیم توسعه داده شده است. این نسخه بر مبنای برنامه درسی سال ۹۲ مصوب وزارت علوم برای رشته مهندسی کامپیوتر با زبان سی شارپ پیاده‌سازی شده است. در ادامه شما را با این نسخه آزمایشی آشنا می‌کنیم.



شکل ٤-١. تصویری از صفحه خانه نرم‌افزار

برنامه به جز صفحه خانه که صرفا جنبه نمایشی دارد شامل بخش‌هایی به نام‌های تعیین گرایش و تمرکز، جدول تطبیق دروس، تاریخچه دانشجو، بارگذاری دروس ارائه شده، تنظیمات پردازشی، انتخاب واحد، برنامه‌های ذخیره شده و تنظیمات می‌باشد.



شکل ٤-٢ تصویری از قسمت‌های تعین گرایش و تمرکز تخصصی

در شکل ۴-۲ بخش تعیین گرایش و تمرکز نشان داده شده است. در تصویر سمت راست کاربر گرایش تخصصی نرم‌افزار و در تصویر سمت چپ کاربر تمرکز تخصصی الگوریتم و محاسبات را انتخاب کرده است.

شناسه	عنوان درس	عنوان درس در دانشگاه مقصد	کد درس در دانشگاه مقصد
۰	اندیشه اسلامی یک - مبدا ومعاد	اندیشه اسلامی یک - مبدا ومعاد	۱۶۱۱۰۰۱
۱	اندیشه اسلامی دو - مبدا ومعاد	اندیشه اسلامی دو - مبدا ومعاد	۱۶۱۱۰۰۲
۲	انسان در اسلام	انسان در اسلام	-
۳	حقوق اجتماعی و سیاسی در اسلام	حقوق اجتماعی و سیاسی در اسلام	-
۴	فلسفه اخلاق	فلسفه اخلاق	-
۵	اخلاق اسلامی	اخلاق اسلامی	-
۶	آیین زندگی - اخلاق کاربردی	آیین زندگی - اخلاق کاربردی	۱۶۱۱۰۰۷
۷	عرفان عملی اسلامی	عرفان عملی اسلامی	-
۸	انقلاب اسلامی ایران	انقلاب اسلامی ایران	۱۶۱۱۰۰۹
۹	آشنایی با قانون اساسی جمهوری اسلامی ایران	آشنایی با قانون اساسی جمهوری اسلامی ایران	-
۱۰	آشنایی با اندیشه سیاسی امام خمینی	آشنایی با اندیشه سیاسی امام خمینی	-
۱۱	تاریخ فرهنگ و تمدن اسلامی	تاریخ فرهنگ و تمدن اسلامی	-
۱۲	تاریخ تحلیل صدر اسلام	تاریخ تحلیل صدر اسلام	۱۶۱۱۰۱۳
۱۳	تاریخ امامت	تاریخ امامت	۱۶۱۱۰۱۴
۱۴	تفسیر موضوعی قرآن	تفسیر موضوعی قرآن	۱۶۱۱۰۱۶
۱۵	تفسیر موضوعی نهج البلاغه	تفسیر موضوعی نهج البلاغه	۱۶۱۱۰۱۷
۱۶	زبان فارسی	زبان فارسی	۱۱۱۱۰۱۷
۱۷	زبان انگلیسی	زبان انگلیسی	۱۱۱۱۰۱۸
۱۸	تربیت بدنی یک	تربیت بدنی یک	۱۱۱۱۰۳۷

شکل ٤-٣ تصویری از بخش جدول تطبیق دروس

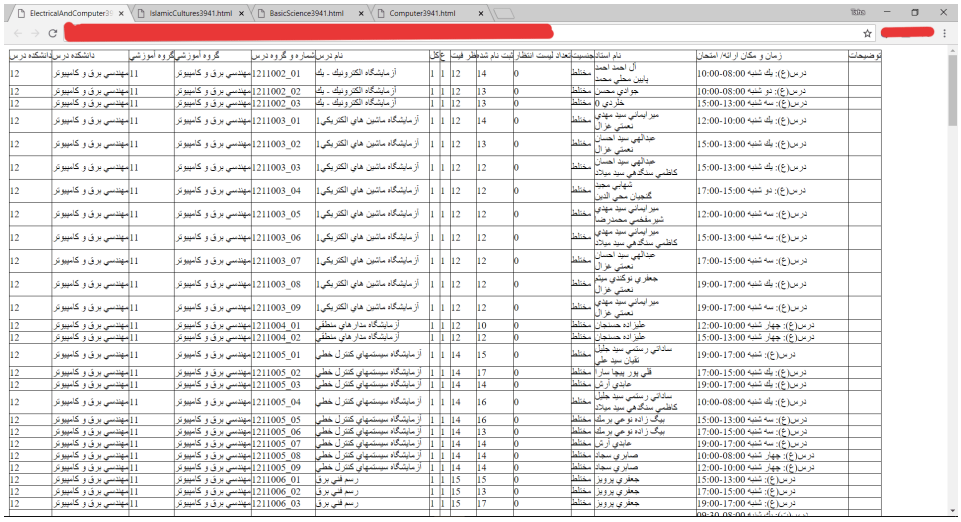
در شکل ۴-۳ بخش جدول تطبیق دروس نرم‌افزار نشان داده شده است و داده‌های آن بر اساس اطلاعات دانشگاه صنعتی نوشیروانی بابل پر شده است. خوب است به این نکته اشاره کنیم که جدول تطبیق دروس به تنهایی نمی‌تواند ضامن سازگاری این نسخه در دانشگاه‌های دیگر باشد چرا که همه دانشگاه‌ها خود را ملزم به رعایت پیشنیازها و همنیازهای برنامه درسی مصوب وزارت علوم نمی‌دانند بنابراین برای آنکه بتوانید این نسخه را در دانشگاهی که چنین شرایطی دارد استفاده کنید، باید داخل کد برنامه، قسمتی که ساختار توصیف کننده برنامه درسی مصوب وزارت علوم توصیف شده‌است، پیشنازها و همنیازها را بر اساس شرایط دانشگاه مورد نظر اصلاح کرد.

شکل ٤-٤ تصویری از بخش تاریخچه دانشجو

در بخش تاریخچه دانشجو درس‌های سبز، درس‌هایی هستند که دانشجو آنها را گذرانده است. درس‌های زرد به این معنی هستند که دروسی را که این درس پیشنیاز آنها است می‌توان با آن همنیاز کرد(مثلا دانشجویی که دوبار درسی را رد شده است طبق قوانین فعلی مجاز به انجام چنین کاری می‌باشد) و سایر درس‌ها سفید هستند.

شکل ٤-٥ تصویری از بخش بارگذاری دروس ارئه شده

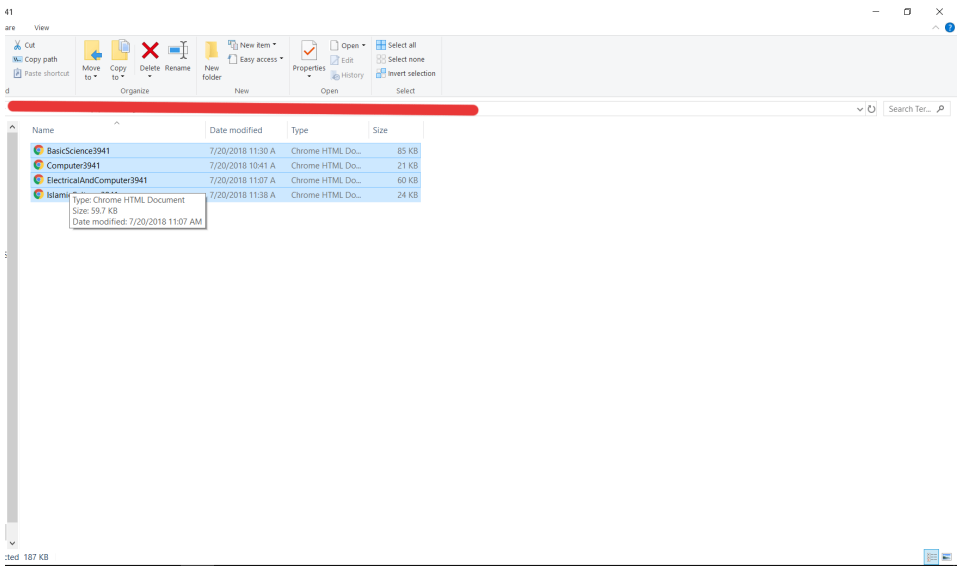
در بخش بارگذاری دروس ارائه شده، می‌توانید فایل‌های مربوط به برنامه دروس ارائه شده در ترم توسط دانشگاه را که در سیستم گلستان قابل دریافت است برای پردازش بارگذاری کنید.



دانشکده درس	دانشگاه درس	گروه آموزشی	گروه آموزشی	شماره گروه درس	نام درس	ظرفیت	ثبت نام شده	ظرفیت	جنسیت	تعداد لیست انتظار	نام استاد	زمان و مکان ارائه امتحان	توضیحات
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211002_01	آزمایشگاه الکترونیک ۱ - یک	1	12	14	مختلط	0	آل احمد یایین مجتبی محمد	درس(ع): یک شنبه 10:00-08:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211002_02	آزمایشگاه الکترونیک ۱ - یک	1	12	13	مختلط	0	جوادی محسن	درس(ع): دو شنبه 10:00-08:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211002_03	آزمایشگاه الکترونیک ۱ - یک	1	12	13	مختلط	0	خلوتی ف	درس(ع): سه شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_01	آزمایشگاه ماشین های الکتریکی۱	1	12	14	مختلط	0	میرایمانی سید مهدی نعمتی عزال	درس(ع): یک شنبه 12:00-10:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_02	آزمایشگاه ماشین های الکتریکی۱	1	12	13	مختلط	0	عبدالهی سید احسان نعمتی عزال	درس(ع): یک شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_03	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	عبدالهی سید احسان کاظمی سنگدهی سید میلاد	درس(ع): یک شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_04	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	شهابی محمد گنجیان محی الدین	درس(ع): دو شنبه 17:00-15:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_05	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	میرایمانی سید مهدی شیرمقصی محمدرضا	درس(ع): سه شنبه 12:00-10:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_06	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	میرایمانی سید مهدی کاظمی سنگدهی سید میلاد	درس(ع): سه شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_07	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	عبدالهی سید احسان نعمتی عزال	درس(ع): سه شنبه 17:00-15:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_08	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	جعفری نوکندی میثم نعمتی عزال	درس(ع): یک شنبه 19:00-17:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211003_09	آزمایشگاه ماشین های الکتریکی۱	1	12	12	مختلط	0	میرایمانی سید مهدی نعمتی عزال	درس(ع): سه شنبه 19:00-17:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211004_01	آزمایشگاه مدار های منطقی	1	12	10	مختلط	0	علیزاده حسنجان	درس(ع): چهار شنبه 12:00-10:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211004_02	آزمایشگاه مدار های منطقی	1	12	12	مختلط	0	علیزاده حسنجان	درس(ع): چهار شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_01	آزمایشگاه سیستمهای کنترل خطی	1	14	15	مختلط	0	سادائی زمستی سید جلیل نقیان سید علی	درس(ع): شنبه 19:00-17:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_02	آزمایشگاه سیستمهای کنترل خطی	1	14	17	مختلط	0	قلی پور بیجا سارا	درس(ع): یک شنبه 17:00-15:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_03	آزمایشگاه سیستمهای کنترل خطی	1	14	14	مختلط	0	حایدی آرش	درس(ع): یک شنبه 19:00-17:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_04	آزمایشگاه سیستمهای کنترل خطی	1	14	16	مختلط	0	سادائی زمستی سید جلیل کاظمی سنگدهی سید میلاد	درس(ع): یک شنبه 10:00-08:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_05	آزمایشگاه سیستمهای کنترل خطی	1	14	16	مختلط	0	بیگ زاده نو عی برمله	درس(ع): سه شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_06	آزمایشگاه سیستمهای کنترل خطی	1	14	13	مختلط	0	بیگ زاده نو عی برمله	درس(ع): سه شنبه 17:00-15:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_07	آزمایشگاه سیستمهای کنترل خطی	1	14	14	مختلط	0	حایدی آرش	درس(ع): سه شنبه 19:00-17:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_08	آزمایشگاه سیستمهای کنترل خطی	1	14	14	مختلط	0	صابری سجاد	درس(ع): چهار شنبه 10:00-08:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211005_09	آزمایشگاه سیستمهای کنترل خطی	1	14	14	مختلط	0	صابری سجاد	درس(ع): چهار شنبه 12:00-10:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211006_01	رسم فنی برق	1	15	15	مختلط	0	جعفری پرویز	درس(ع): شنبه 15:00-13:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211006_02	رسم فنی برق	1	15	13	مختلط	0	جعفری پرویز	درس(ع): شنبه 17:00-15:00	
12	مهندسی برق و کامپیوتر	11	مهندسی برق و کامپیوتر	1211006_03	رسم فنی برق	1	15	17	مختلط	0	جعفری پرویز	درس(ع): شنبه 19:00-17:00	

شکل ٤-٦ تصویری از نمونه فایل‌های قابل دستیابی در سیستم گلستان

ابتدا باید برنامه‌های مورد نیاز خود را در سیستم خود به صورت فایل‌های **اچ‌تی‌ام‌ال**ای ذخیره کنید. برای مثال ما در اینجا چهار فایل مربوط به دروس ارائه شده در ترم توسط برق و کامپیوتر، کامپیوتر، علوم پایه و معارف اسلامی را ذخیره کردیم.



شکل ٤-٧ تصویری از فایل‌های ذخیره شده مربوط به دروس ارائه شده در ترم توسط دانشگاه

سپس همه فایل‌های مورد نیاز خود را انتخاب کرده با drag and drop آنها را به داخل شکل ابر مانند موجود در بخش بارگذاری دروس ارائه شده بکشید و رها کنید و در آخر بر روی گزینه تهیه فایل هدف کلیک کنید.

شکل ٤-٨ تصویری از بخش بارگذاری دروس ارائه شده پس از تهیه موفقیت آمیز فایل هدف

چنانچه فایل هدف با موفقیت تولید شود بخش کشیدن و رها کردن به رنگ سبز در می‌آید.

شکل ٤-٩تصویری از بخش تنظیمات پردازشی

در بخش تنظیمات پردازشی، یک فرم تعبیه شده است که طی آن برخی از اطلاعات مورد نیاز برای اجرای الگوریتم را مشخص می‌کند. در قسمت ترم می‌توانید مشخص کنید به عنوان دانشجوی ترم چندم می‌خواهید انتخاب واحد کنید. در قسمت حداقل و حداکثر واحد می‌توانید مشخص کنید برنامه‌هایی که می‌خواهید الگوریتم به شما پیشنهاد دهد، حداقل و حداکثر چند واحد باشد (اگر دقیقا تعداد خاصی واحد را مد نظر دارید کافی است حداقل و حداکثر واحد را روی همان واحد مد نظرتان تنظیم کنید). فیلد بعدی مشخص می‌کند برای اعتبارسنجی درس‌های سبز حداکثر چقدر زمان بر حسب میلی ثانیه در نظر گرفته شود. فیلد زمان پردازش الگوریتم پیشنهاد برنامه ترمی مشخص می‌کند پردازش الگوریتم بر حسب میلی ثانیه چقدر طول بکشد. فیلد حداکثر تعداد برنامه‌های پیشنهادی مشخص می‌کند الگوریتم حداکثر چند برنامه ترمی به شما پیشنهاد دهد. گزینه فیلتر کردن سطرهای بدون ظرفیت مشخص می‌کند آیا سطرهایی که ظرفیت آنها پر شده است فیلتر شوند یا خیر. گزینه بررسی تداخل زمانی امتحانات مشخص می‌کند آیا علاوه بر تداخل

نداشتن کلاس‌ها تداخل نداشتن زمان امتحانات نیز بررسی شود یا خیر. در آخر با توجه به جنسیت می‌توانیم سطرهایی که از نظر جنسیت برای شما مجاز نمی‌باشند را فیلتر کنیم.

شکل ٤-١٠ تصویری از بخش انتخاب واحد

در بخش انتخاب واحد درس‌هایی که امکان اخذ آنها را دارید به همراه سطرهایی که ممکن است آنها را اخذ نمایید نمایش داده می‌شود. نام این دروس به همراه کد آنها در سمت راست این بخش نمایش داده می‌شوند. بر روی هر درس که کلیک کنید، رنگ دکمه آن درس خاکستری می‌شود و سطرهای آن به نمایش در می‌آید.

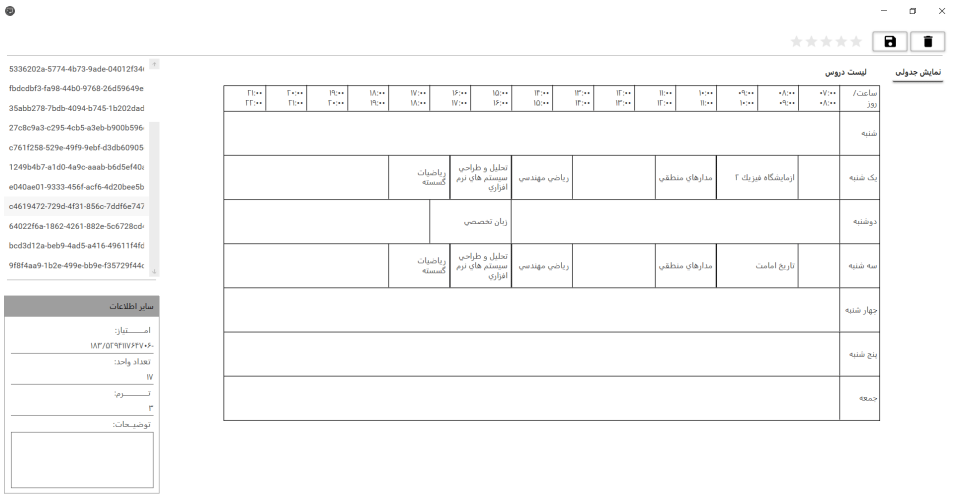
در بالای صفحه سمت راست، سه مربع قرار گرفته‌اند. با کلیک بر روی مربع اول از سمت راست می‌توانید مشخص کنید درس سبز باشد(در برنامه های پیشنهادی حتما این درس وجود داشته باشد) یا خیر(بسته به شرایط درس بتواند اخذ شود یا نشود). با کلیک بر روی مربع قرمز رنگ می‌توانید همه سطرهای سفید رنگ یک درس غیر سبز را به رنگ قرمز درآورید. و با مربع کناری می‌توانید همه سطرهای قرمز یک درس را سفید کنید. علاوه بر این با دوبار کلیک بر روی هر سطر می‌توانید در صورت مجاز بودن، رنگ سطر را از سفید به قرمز یا از قرمز به سفید تغییر دهید. سطرهای قرمز به این معنی هستند که در برنامه‌های پیشنهادی الگوریتم نباید این سطرها حضور داشته باشند. در بخش مربوط به فیلتر زمانی می‌توانید رنگ کردن سطرها را با شیوه‌ای متفاوت انجام دهید. برای مثال فرض کنید به دلایلی مثلا کار نیمه وقت نمی‌خواهید در زمان‌های خاصی کلاس داشته باشید در این صورت می‌توانید بازه‌های زمانی و روزهایی که نمی‌خواهید کلاس داشته باشید را در این بخش مشخص کرده و رنگ آنها را به کمک این ابزار قرمز کنید. اگر از بخش انتخاب واحد به بخش دیگری منتقل شوید یا نرم‌افزار را بسته مجدادا اجرا کنید اطلاعات این بخش از دست خواهد رفت چنانچه می‌خواهید در مراجعات بعدی به این بخش اطلاعات شما ذخیره شده باقی بماند(تغییراتی که در رنگ سطرها و درس‌ها داده‌اید) می‌توانید بر روی گزینه موجود در قسمت ذخیره‌سازی کلیک کنید تا با این کار اطلاعات شما ذخیره شود. پس از خاتمه اجرا می‌توانید برنامه‌های پیشنهادی را با کلیک بر روی دکمه سمت چپ همین قسمت مشاهده کنید.

وقتی تنظیمات مورد نظر خود را انجام دادید، می‌توانید از قسمت پیشنهاد برنامه درسی، بر روی دکمه سمت راست آن

کلیک کنید تا الگوریتم پیشنهاد برنامه‌های ترمی اجرا شود.

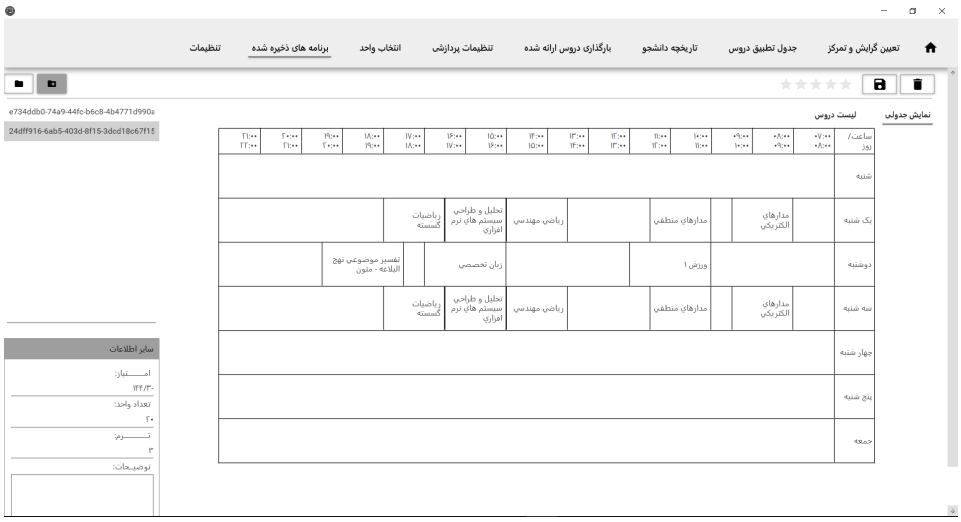
شکل ٤-١١تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش لیست دروس

الگوریتم‌های پیشنهاد شده توسط الگورتیم با دو شیوه **نمایش جدولی** و **لیست دروس** قابل نمایش هستند. در شیوه لیست دروس، درس‌ها با ترتیبی قابل اخذ برای سیستم گلستان نمایش داده می‌شوند.



شکل ٤-١٢ تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش جدولی

در شیوه نمایش جدولی هم، برنامه پیشنهادی در قالب جدول برنامه هفتگی به نمایش در می‌آید.



شکل ٤-۱۳تصویری از بخش برنامه‌های ذخیره شده

در بخش برنامه‌های ذخیره شده، می‌توانید هم به برنامه‌های پیشنهادی در آخرین اجرای الگوریتم و هم بـه برنامـه‌هـایی که برای خود ذخیره کرده‌اید دسترسی پیدا کنید.

شکل ٤-١٤ تصویری از بخش تنظیمات

علاوه بر آنچه تا به اینجا گفته شد، این نسخه، بخشی به نام تنظیمات دارد که از طریق آن می‌توانید تنظیمات مربوط بـه نرم‌افزار را انجام دهید.

٤-۳ نتایج حاصل از آزمایشات

در شکل‌های ۴-۱۲ و ۴-۱۳ دو نمونـه از برنامـه‌هـای ترمـی پیشـنهادی توسـط الگـوریتم مشـاهده مـی‌شـوند. در اینجـا نمونه‌های بیشتری را ارائه نمی‌کنیم اما در مورد برخی از نکات در رابطه با نمونه‌هایی که با آزمـایش بـرای دوره تحصـیلی یکی از دانشجویان در هر ترم بدست آمده، بحث می‌کنیم.

برنامه‌های پیشنهاد شده معمولا در عمل به گونه‌ای بودند که، روز یا روزهایی در برنامه هفتگی آزاد می‌ماند(کلاسی نداشته باشند).

اغلب در عمل وقتی یک محدوده حداقل و حداکثر واحدی برای الگوریتم انتخاب می‌کنیم بیشتر جواب‌هایی که پیدا می‌شوند متمایل به یکی از واحدها هستند مثلا اگر حداقل و حداکثر واحد را به ترتیب ۱۷ و ۱۹ در نظر گرفته و تعداد برنامه پیشنهادی را ۱۵ در نظر بگیریم در ۱۵ برنامه پیشنهادی تعداد زیادی از برنامه‌ها مربوط به یکی از واحدها مثلا ۱۸ هستند و برنامه‌های پیشنهادی از سایر واحدها در اقلیت قرار می‌گیرند. در فصل قبل وقتی معیاری را برای ارزیابی ارزش هر برنامه ترمی معرفی می‌کردیم خاطر نشان کردیم که می‌خواهیم برنامه‌های پیشنهادی صرفا برای یک واحد خاص نباشند که این امر محقق شد و همینکه تعدادی اقلیت از واحدهای دیگر هم در برنامه‌ها می‌آیند نشانگر همین موضوع است. با این حال این اقلیت بودن خود شک برانگیز است چرا که این سوال را پیش می‌آورد که در اکثریت چشمگیر قرار گرفتن یکی از واحدها در برنامه‌های پیشنهادی به چه دلیل است؟ بنظر می‌رسد پاسخ را باید در سه احتمال جستجو کرد. اول آنکه ممکن است معیار ارزیابی ارائه شده موجب این امر شده باشد. دوم اینکه ممکن است خود الگوریتم و شیوه گزینش آن موجب چنین امری شود. سوم این احتمال وجود دارد که در عمل محدودیت‌هایی که توسط برنامه دروس ارائه شده توسط دانشگاه در ترم و شرایط دانشجو اعمال می‌شوند واقعا ما را در وضعیتی قرار می‌دهند که تعداد بخصوصی از واحد برای اخذ در آن ترم مناسب تر باشد. ممکن هم هست که ترکیبی از این موارد باعث چنین امری شده باشند. به هر حال تا به اینجا من نتوانسته‌ام نتیجه گیری کنم که دلیل این اتفاق چیست و آیا اصلا این خوب است یا نه.

علاوه بر آنچه تا به اینجا گفته شد خوب است یادآور شویم که الگوریتم پیشنهاد برنامه‌های ترمی براساس معیاری که طراح، آن را مناسب می‌دانسته توسعه یافته ولی شکی نیست که دیدگاه‌ها و سلایق مختلف می‌توانند معیارهای دیگری را برای یک برنامه، خوب بدانند. در آخر ارزیابی من از برنامه‌های پیشنهادی این است که فکر می‌کنم براساس معیارهایی که من برای یک برنامه ترمی، خوب می‌دانم؛ الگوریتم برنامه‌های بسیار خوب و همچنین متنوعی را در نتیجه آزمایشات انجام شده ارائه داده است.

٤-٤ خلاصه

در این فصل ابتدا با یک نسخه آزمایشی توسعه داده شده آشنا شدیم و بخش‌های مختلف آن را به همراه چگونگی بکار بردنشان توضیح دادیم. سپس به تشریح نتایجی که از برنامه‌های پیشنهاد شده توسط الگوریتم بدست آمده بودند پرداختیم.

فصل پنجم:
نتیجه‌گیری‌ها و پیشنهادها

۵-۱ نتیجه‌گیری‌ها

در این پروژه پس از مطالعه‌ی برنامه‌های درسی مصوب وزارت علوم به تحلیل و طراحی جهت ارائه راهکاری عملی و کارآمد برای پیشنهاد برنامه‌های ترمی جهت انتخاب واحد دانشجویان مبادرت کردیم. سپس نسخه‌ای آزمایشی در همین جهت پیاده‌سازی شد که در عمل توانست برنامه‌های ترمی مناسبی را در زمانی کوتاه و قابل تنظیم ارائه دهد. سعی شد تا قسمت تحلیل و طراحی اولیه، بسیار دقیق و جامع باشد که به نظرم این امر محقق شد با این وجود اما ساختاری که می‌خواستم به عنوان توصیف کننده برنامه‌های درسی ارائه دهم در حالت کلی شکست خورد و ساختار به ساختار توصیف کننده برنامه‌های درسی وزارت علوم محدود شد. اما در نهایت فکر می‌کنم پروژه خوبی بود هرچند نیاز به کار بیشتری دارد.

۵-۲ پیشنهادها

برخی از پیشنهادهایی که برای ادامه این پروژه می‌توان ارائه داد را در ادامه لیست کرده‌ایم:

- ارائه ساختار توصیف کننده برنامه‌های درسی برای حالت عمومی و یا اصلاح همین ساختار فعلی
- افزودن سایر اولویت‌های باید و شاید
- گنجاندن امکان اصلاح پیشنیازها و همنیازهای دروس در برنامه کاربردی
- توسعه یک برنامه کاربردی که تولید ساختار توصیف کننده برنامه‌های درسی را به صورت ویژوال تسهیل نماید.
- اضافه کردن امکاناتی جهت افزودن اسکریپت‌ها یا تکه برنامه‌ها در برنامه کاربردی با این ایده که هر شخص بتواند معیارهای خود در مورد برنامه ترمی خوب، توصیف کند.
- اضافه کردن امکان انتخاب واحد خودکار که در آن دانشجو برنامه‌ای از برنامه‌های پیشنهادی را انتخاب کند و برنامه کاربردی به صورت خودکار پس از تائید توسط دانشجو آن برنامه را در سیستم برایش اخذ نماید.