نکات فصل اول:

در عمل انسان گونه سیستمی هوشمند هست که مانند انسان عمل کند یعنی شبیه سازی کند در مورد دوم نحوه تفکر و استدلال ذهن انسان را باید شبیه سازی کنیم. دقت کن در عمل انسان گونه باید تماما مانند انسان شبیه سازی شود و رفتار کند. در عمل عقلایی ممکن هست دانش ما کامل نباشد نسبت به محیط صرفا بر اساس دانش فعلی بهترین تصمیم را بگیرد . GPS میاد دقیقا رفتاری که انسان در مغز دارد برای حل یک مسئله را گام به گام شبیه سازی کند. تست تورینگ کامل برای قابلیت های فیزیکی هست و برای هوشمندی لازم نیست. برنامه ای وجود دارد که هر سوال قابل حلی را به زبان منطقی تبدیل کند. استنتاج و تفکر یکی از راه های رسیدن به عمل عقلایی هست. مزیت دیگر عمل عقلایی این هست که اون عقلانیت یک مفهوم ریاضی دارد و حالت بهینه قابل محاسبه است. عامل منطقی باید ویژگی های تست تورینگ را داشته باشد. اعمال انعکاسی مثل پلک زدن میتوانند منطقی برآورد شوند. هر چیزی عامل نیست مثلا اینکه فقط یک سری اعداد را جمع کند و میخواهیم که یک سری اعمال غیر بدیهی انجام دهد. دقت کن به این هر مسئله بستگی دارد که چه چیزی عامل هست یا نیست صرفا با اون ویژگی های اولیه سنسور و اکشن نمیتوان گفت عامل هست. بزرگترین چالش این هست که برنامه ی عامل ما هر لحظه فقط یک درک به عنوان ورودی میگیرد در صورتی که تابع عامل ما مجموعه از ادراک هست و این چالش اصلی ما هست یک راه این هست که بیایم به صورت یک جدول تشکیل بدهیم و بگذاریم درون برنامه عامل همان تابع عامل جدول شده را، این رویکرد نیاز به حافظه بسیار بزرگی دارد و منطقی نیست. و دوست داریم عامل ما خودش یاد بگیرد نمیخواهیم از قبل همه چیز درست را بهش بدهیم. عامل را بر اساس چیز هایی که دارد قضاوت میکنیم. عامل ما دانای کل نیست بر اساس دانش فعلی بهترین تصمیم را میگیرد. و عامل همچنین باید سعی کند دانش خودش را بیشتر کند و یاد بگیرد. دقت شود ویژگی های مثل مشاهده پذیری و ... ویژگی های task environment هست ویژگی مسائل هوش مصنوعی یعنی اون 4 تا PEAS را تعریف کنیم بعد راجب مشاهده پذیری ها صحبت کنیم و منظورمان نیست مثلا محیط فیزیکی چطور هست و دقت کن مثلا بر اساس سنسور های عامل میتوانیم بگوییم محیط چه قدر مشاهده پذیر هست. مثلا اینکه بگیم محیط یک جارو برقی چه ویژگی دارد سوال خوبی نیست چون مثلا سنسور های آن را مشخص نکردیم. مبتنی بر انسان برای تحقیق مناسب نیست مبتنی بر منطق مناسب هست. از نظر تئوری میشود هر مسئله به زبان منطق را حل کرد. در شرایط خاص اعمال انعکاسی عقلایی محسوب میشوند.

نکات فصل دوم:

ارتباط با سایر عامل ها عقلایی است. در محیط چند عامله تصادفی عمل کردن عقلایی است برای اینکه حریف را گیج کنی. عامل مبتنی بر سودمندی یک عامل عقلایی است. دقت کن تنها راهش مبتنی بر سودمندی بودن نیست برای عامل عقلایی بودن. دقت کن در محیط قطعی میدانیم نتیجه اکشن چی هست حالا یک حالت هست که میشود قطعی یا چند حالت که میشود غیر قطعی ولی در محیط نا شناخته اصلا نمیدانیم نتیجه اکشن چی هست و باید حتما انجام بدهیم تا بفهمیم به چه حالت یا حالاتی میرویم. مثلا نا شناخته بودن یعنی دکمه بالا در یک بازی ویدیویی یعنی این دکمه رو بزنی نمیدانی چه اتفاقی قرار هست بیفتد. همه این ویژگی ها با هم قابل رخ دادن هستند و مستقل از همدیگر هستند و هیچ تاثیری رو همدیگر ندارند. واسه گسسته یا پیوسته بودن به زمان دقت کن و واسه ترتیبی یا اپیزودیک ببین حالت فعلی به تاریخچه ربطی دارد یا نه. دقت کن ویژگی های محیط مستقل از اکشن های عامل هست. عامل عقلایی ما را به حالاتی میبرد که در آنها معیار کارایی مسئله در آن بیشترین هست یا معیار کارایی ما. معیار کارایی بر اساس حالت مسئله باشد نه حالت عامل چون اونطوری هر کی از خودش راضی تر باشد امتیاز بیشتر میگرد مثلا اگر اتاق تمیز بود 1 امتیاز مثبت. برای عقلایی بودن یک عامل باید PEAS تعریف شود. هر چه قدر دانش عامل از محیط و قدرت اکشن هاش و سنسور هاش قوی تر باشد بعلاوه معیار کارایی کار ها برای ما آسان تر هست. رفتار عامل با تابع آن مشخص میشود که دنباله را به اکشن مپ میکند. اگر سخت افزار ثابت باشد، هر برنامه عامل دقیقا یک تابع عامل را پیاده سازی میکند. برای هر تابع عامل نمیتوان برنامه ای طراحی کرد مثلا نیاز به حافظه بینهایت باشد. تابع عامل یک مفهوم انتزاعی و ریاضی هست. در عامل واکنشی ساده فقط بر اساس درک فعلی اکشن را انجام میدهیم و برای این محیط باید کاملا مشاهده پذیر باشد و حالات قبلی را نیاز به ذخیره سازی ندارند. عامل واکنشی مبتنی بر مدل یک حالتی درونی نگه میدارند واسه قسمت هایی که نمیتوانند ببینند که این یک حدس هست برای اپدیت این حالت یک مدل میسازیم که کار مدل این هست که بگوییم محیط چگونه تغییر میکند که از 2 طریق یک اینکه محیط چگونه تغییر میکند و 2 اینکه اکشن ما چه تاثیری در این محیط دارد. در 2 عامل قبلی صرفا یک سری شرط داشتیم که اگر این بود اینکار را بکن. اما در عامل مبتنی بر هدف یک هدفی هم داریم و سعی میکنیم به آن برسیم در 2 تای قبلی باید به ازای هر حالت یک سری شرط بنویسیم و اگر مثلا محیط یک مقدار تغییر کند همه قوانین باید از اول نوشته شود اما در مبتنی بر هدف بر اساس هدف اکشن را تعیین میکنند چون باعث میشود یک سری محاسبات بیشتری انجام بدهیم کارایی را از دست میدهیم چون باید به آینده هم فکر کنیم و یک مقداری اکشن از دست میرود. در مبتنی بر سود میگوییم هدفی بهتر است که ما احساس رضایت بیشتری بکنیم. دقت کن عامل عقلایی ربطی به عامل مبتنی بر سودمندی ندارد هر 4 تا عامل میتواند عقلایی باشد چه واکنشی ساده چه مبتنی بر مدل چه مبتنی بر هدف و چه مبتنی بر سودمندی میتوانند عقلایی باشند. عنصر یادگیری دانش عامل را بروز میکند بر اساس فید بکی که نقاد بهش میدهد که عامل چه قدر دارد درست کار میکند خود نقاد از یک سری از استاندارد های کارایی استفاده میکند تا مقایسه کند مثلا اگر بدون راهنما بپیچیم به چپ بقیه فحش بدهند نقاد این را متوجه میشود که بد هست. اگر محیط کاملا مشاهده پذیر باشد نیازی به حافظه نداریم چون در هر لحظه کل محیط را میبینیم و نیازی به ذخیره سازی چیزی نیست. گاهی خوب هست در محیط های رقابتی به شکل تصادفی عمل کنیم. پویا مثل شطرنج. شناخته بودن ربطی به PEAS ندارد به دانش عامل بستگی دارد. وضعیت محیط از ناشناخته به شناخته در طی زمان تبدیل میشود. واکنشی مبتنی بر مدل میاد یک حافظه در نظر میگیرد و از محیط اطلاعات جمع میکند به این میگوییم مدل. دقت کن هر عامل میتواند عنصر یادگیری هم داشته باشد. دقت کن خصوصیات محیط ربطی به کارایی که ما میکنیم ندارد یعنی کار ما باعث نمیشود محیط پویا شود. عامل مبتنی بر مدل بخاطر بحث مشاهده ناپذیر بودن هست یعنی اون بخشی از جهان را که نمیبینی بیای و مدل کنی. انعکاسی یعنی همان شرط عمل و همه دانش عامل را میتوانیم به این صورت بنویسیم. انعکاسی ساده نمیتواند در نیمه مشاهده پذیر کاری بکند البته برای همه محیط ها نیست کلا نمیتوانیم بگوییم. عقلانی بودن را بر اساس امکانات موجود میسنجیم نه ویژگی های محیط. دقت کن اگر ما محیطی داشتیم که چند تا هدف توش بود و اهداف با هم در تناقض بودند، پس باید از عامل های مبتنی بر سودمندی استفاده کنیم. دقت کن کسی که با زبان طبیعی صحبت میکند چون به ذهن طرف دسترسی نداریم تا منظور واقعی او را متوجه شویم نیمه مشاهده پذیر هست. استنتاج کردن یکی از ویژگی های پاس کردن تست تورینگ هست نه همه آن چندین ویژگی هست. عامل معقول نتیجه واقعی را نمیداند بر اساس دانشی که دارد انتخاب میکند کاری را که فکر میکند بهتر هست. هدف گرا چون به آینده هم فکر میکند کارایی آنها افت میکند. تک عامله تصادفی سازی را برای فرار از حلقه های بینهایت استفاده میکند و بهتر هست زیاد از تصادفی سازی استفاده نکنیم و بگذاریم در چند عامله. ندانیم حریف چه کار میکند تاثیری در قطعیت محیط ندارد. اگر حافظه n بیت باشد حداکثر دو به توان n تا دنباله ادراک را نشان بدهیم پس بیشتر از آن نمیتواند بشود و دلیلی ندارد. مشاهده پذیری به سنسور ها ربط دارد و اگر ثابت باشد مشاهده پذیری تغیری نمیکند. شناخته بودن در طول زمان تغییر میکند، اول ناشناخته بعد شناخته. تابع عامل همیشه یک تابع قطعی نیست چون اتفاقا میخواهیم عامل ما رفتار تصادفی داشته باشد و قابل پیشبینی نباشد. دقت کن هر چه قدر عامل بیشتر نیاز داشته باشد تحلیل کند کارایی کمتری دارد. مبتنی بر مدل حافظه دارد که واکنشی ساده ندارد پس کارا تر هست.

نکات فصل سوم:

در جستجو گرافی فقط گره با اولویت بالاتر را نگه میداریم اگر دو گره یکسان داشتیم که در فرانتیر بودند و اکسپند نشده بودند. در گراف سرچ هر گره حداکثر یکبار گسترش داده میشود و بیشتر از آن نخواهد بود. ولی در درختی ما گره تکراری داریم. اگر یک گره ای در explored set بود حتی دیگر تولید نمیشود در جستجو گرافی. اگر گره را تولید کنی ولی هنوز گسترش ندادی و در ادامه دوباره میرسی که دوباره همون گره تولید شده و میخواد بیاد تو فرانتیر در BFS کم عمق تر یا قبلی را نگه میدارد DFS جدید تر را نگه میدارد براش عمق مهم هست در UCS میاد کم هزینه ترین و بهتر را نگه میدارد. الگوریتم BFS یک ویژگی دارد این هست که تست هدف در زمان تولید انجام شود ولی برای سایرین این درست نیست و دیگر بهینه نخواهد بود. BFS صف هست هر کی زودتر بیاد زودتر هم میره و ترتیب ورود با ترتیب گسترش یکسان هست. DFS در درختی کامل نیست اگر فضای حالت دور نداشته باشد کامل هست در گرافی هم کامل هست چون دور ندارد و بهینه نیست. بدترین مرتبه DFS هست چون تا M رفته. در UCS و بعد در A\* هر گره ای که برای گسترش انتخاب میکنیم مسیر رسیدن بهینه به آن پیدا شده است.

جستجو دو طرفه تا جایی میرود که frontier ها گره مشترک داشته باشند. حالت هدف نباید داشته باشیم باید تست هدف داشته باشیم. اول باید هدف را فرموله کنیم و حالات مسئله را بدست بیاوریم پس هر حالت بودن در یک حالت هست همه چیز بر اساس هدف هست اولین کار فرموله کردن هدف همواره قبل از فرموله کردن مسئله هست. دقت کن منظور از تست هدف در زمان گسترش یعنی به ازای هر فرزند که تولید کردی درجا چک کن هدف هست یا نیست. دقت کن تست هدف در زمان گسترش یعنی قبل اینکه گسترش پیدا کنه تست هدف میشه. اگر نگفت جستجو درختی یا گرافی پیشفرض بگذار جستجو درختی. Dfs اونی که عمق بیشتری دارد را نگه میدارد در گرافی برای گره تکراری. ترتیب گسترش با مسیر هدف فرق دارد ترتیب گسترش همه گره های که explored شدن هست ولی مسیر هدف از شروع تا هدف هست. دقت کن در UCS اگر دو تا گره تکراری هزینه یک اندازه ای داشتند اولی را فقط نگهدار. UCS قطعا همه گره ها با هزینه کمتر از C\* را گسترش میدهد مانند A\* که اون هم با هیورستیک همین کار را میکرد. پیش فرض الگوریتم ها تست هدف در زمان گسترش هست. BFS اونی را بدست میاورد که کمترین تعداد اکشن را دارد. BFS درختی و گرافی فرقی نمیکند. DFS درختی ممکن هست به جواب نرسد و کامل نباشد. فاصله منهتن میشود مجموع فاصله افقی و عمودی تا هدف و یک هیورستیک سازگار و خوب هست. اگر در یک حالت باشیم و دیگر حالت قبلی را نبینیم DFS درختی جواب میدهد و اون خونه تغییر میکند و گیر نمیکند و اون خانه را دوباره نمیبینیم. BFS درختی وقتی هزینه یال ها یکسان باشد همیشه بهینه هست. BFS اگر هزینه یال ها مساوی و مثبت باشد. برای DFS که کلا هیچ تضمینی نیست حتی اگر مثبت باشد هزینه ها و برای UCS هزینه یال ها باید کران مثبت داشته باشد و افزایشی باشد برای جواب بهینه.

بخش دوم

دقت کن کامل بودن a\* ربطی به هیورستیک ندارد اگر صرفا کران بالای آنها مثبت باشد یعنی هزینه یال ها افزایش باشد کافی هست ممکن هست به جواب بهینه نرسد فقط. اگر هیورستیک سازگار باشد مقادیر f وقتی میایم پایین غیر نزولی هست، هر الگوریتمی که تضمین بهینه بودن بکند قطعا مقدار گره بزرگتر مساوی a\* تولید میکند. اگر هیورستیک قابل قبول باشد اونی را برمیداریم که بیشتر تخمین میزند یعنی با تعداد کمتری گره به جواب میرسیم. پیشفرض درختی هست مگر اینکه بگوید گرافی. برای چک کردن قابل قبول بودن کمترین هزینه مسیر را از همان گره تا هدف بگیر. در SMA باید هیورستیک قابل قبول باشد و همچنین جواب باید تو حافظه ای که میدهیم جا شود. اگر نتواند مسیر بهینه را جا کند بین مسیر های موجود بهترین را انتخاب میکند. دقت کن database میاد یک نمونه کوچکتر شده را حل میکند و محدود تر میکند و بعد از آن هیورستیک بدست آمده برای مسئله اصلی هم قابل قبول هست. a\* قابل قبول نباشد تضمینی نمیکند مسیر بهینه را پیدا کند. دقت کن هیورستیک دادن ببین مقداری میتواند از h\* بیشتر شود تا آن را رد کنی یا نه.

نکات فصل چهارم:

دقت کن برای حالات مسئله 2 نوع فرموله سازی داریم یکی فرموله سازی حالت کامل که مثلا در 8 وزیر هر 8 وزیر روی صفحه شطرنج هستند و یک حالت دیگر فرموله سازی افزایشی هست که به ازای هر گام که جلو میرویم مثلا یک وزیر در صفحه بگذاریم و اول کل صفحه شطرنج خالی هست. بعد حالت های مسئله باید تابع هدف تشخیص کنیم تا یک ارزیابی باشد برای هر حالت مثلا همون جفت وزیر هایی که همدیگر را تهدید نمیکنند. تپه نوردی تصادفی یک آزمایش برنولی هست. یعنی به ازای p اگر درست باشد به ازای 1-p هم میتواند اتفاق نیفتد. مثلا برای دیدن اولین 6 در پرتاب تاس به طور متوسط 6 تا پرتاب نیاز داری که میشود 1/6 میشود تعداد متوسط برای دیدن 6 . اگر دما را به اندازه کافی کاهش دهیم به احتمال 1 به بهینگی سراسری میرسیم. Or انتخاب های ما هست مثل قبلا ولی and انتخاب های محیط هست، چون محیط هم دخیل هست هر برگ این درخت قطعا هدف هست. یعنی محیط ما را به هر جایی ببرد باید مطمئن باشیم که هدف هست. گره های and باید شامل تمام خروجی هایی ممکن باشد. جستجو بدون مشاهده پذیری یعنی اینکه محیط برود داخل حالت باور یعنی باور کند بلاخره در یکی از حالات هست. محیط کاملا مشاهده پذیر و محیط غیر قابل مشاهده از یک جنس هست ولی مسائل جزئی مشاهده پذیر و غیر قطعی این شکلی نیستند، و شرطی میشود چون یک چیز هایی از محیط دریافت میکنیم. دقت کن اون غیر قطعی هم آفلاین هست یعنی محاسبات از قبل میشود انجام داد آنلاین ها یک دسته دیگر هستند کلا. ضریب رقابتی: نسبت هزینه در جستجوی آنلاین نسبت به جستجو آفلاین. دقت کن فرق پرتو محلی این هست که به حالت بهتر میرود لزوما شاید نسبت به حالت فعلی هم بهتر نباشد. تصادفی یکنواخت همان توزیع یکنواخت هست یعنی به احتمال یکسانی همه عضو ها را انتخاب کنیم یعنی 100 تا عضو داریم احتمال هر کدام 1/100 باشد. جمعیت اولیه ژنتیک باید به صورت رندوم باشد. ژنتیک تا کی میتوانیم ادامه بدهیم ؟ ماکزیمم تعداد نسل یعنی 100 نسل فقط یا به هدف برسیم. الگوریتم گرادیان خودش یک نوع الگوریتم تپه نوردی هست و از آن سریعتر نیست. کلا تپه نوردی با پرتو محلی معادل نیست و این 2 تا را نمیتوان با هم مقایسه کرد. در تپه نوردی ممکن است تمام جمعیت فعلی بدتر از تمام جمعیت تو حالت قبلی باشند. تپه نوردی میگفت که از حالت فعلی باید بهتر باشد ولی همچین چیزی را راجب پرتو محلی نداریم پرتو صرفا میگوید بهترین ها را انتخاب کن. تپه نوردی حافظه یکسان دارد. A\* بهینه هست و UCS هم بهینه هست و برتری نسبت به همدیگر ندارند. البته که اگر هیورستیک سازگار باشد با گسترش کمتر مساوی گره به حالت هدف میرسیم. اگر نگفت A\* گرافی یا درختی پیش فرض را درختی بگذار یعنی گره تکراری هم میشود. مسیر با ترتیب گسترش گره ها فرق دارد. اگر گرافی را اجرا کنیم مسیر گزینه 1 به جواب میرسد. هیورستیک داخل گره ها هست و ما تابع هدف محلی را همان هیورستیک در نظر میگیریم. تابع جانشین همان تابع حالت تولید همسایه ها هست. تعداد همسایگی تابع جانشین الف؟ 8 ضربدر 2 چون 8 تا نود داریم که هر کدام میتوانند دو رنگ مشکلی و خاکستری بگیرند، اون حالت برای بهینه سراسری نیست و ربطی به جانشین ندارد. اگر الف باشد مشخص هست که بهینه محلی هست چون یعنی خودش از سایر حالات بهتر هست و تو دقت کن رنگ هر کدام از حالات را عوض کنی مقدار تابع هدف کمتر میشود. انتسابی که بدهیم تمام حالات بدتر میشود یعنی حالت فعلی بهینگی محلی هست. بهینه سراسری ربطی به تابع جانشین ندارد. با تابع ب انگار در شانه گیر کردیم و بهینگی محلی نیست زیرا بعد از یک حرکت دیگر میتوانیم به بهینگی سراسری برسیم. دقت کن سوال باید بگوید که تست هدف مساوی دما برابر صفر نیست چون اگر باشد اصلا دیگر الگوریتم تمام میشود و هم ارز با چیزی نیست. هم ارز بودن با random walk در الگوریتم ژنتیک تنها در صورتی هست که n=1 باشد تا از مرحله انتخاب و crossover عبور کند و فقط در مرحله جهش که کاملا تصادفی هست تغییر میکند. اگر k بینهایت باشد دیگر برترین ها معنی نمیدهد. حتی با هیورستیک کاملا درست و دقیق هم نمیتوانیم بگوییم تپه نوردی حتما به جواب میرسد. دقت کن در ذوب شبیه سازی شده گره به صورت تصادفی تولید و انتخاب میشود. دقت کن loge، یک هست. دقت کن اگر B انتخاب شود که هیچ انتخاب میشود پس دیگر شانسی برای A وجود ندارد خود A هم شانس 1/3 را دارد ولی اگر C انتخاب شود باز هم A شانس دارد چون احتمال C کمتر از 1 و ½ هست که خوب این باعث میشود باز A انتخاب شود دقت کن 1/3 ضربدر ½ اول برای C هست و 1/3 بعدی برای A هست. این حالت تا ابد ادامه دارد. دقت کن لازم نبود این محاسبات را بکنی از نظر ذوب شبیه سازی شده گره A و B به یک اندازه ارزش دارند اما گره C به اندازه ½ ارزش دارد پس ارزش A, B به اندازه 2 برابر C هست پس احتمال هم که میدانیم همه باید برابر 1 شود پس میشود 2/5 سهم A, B. دقت کن اگر K برابر بینهایت یا N باشد و N هم کل حالات مسئله باشد شبیه به BFS و شبیه به هم هستند و طبیعتا به جواب میرسیم. دقت کن در گزینه بعدی 1000 یک حد بالا هست و از اونجایی که مسئله ما کمینه سازی هست و طی چندین گام حتی اگر از بیشترین مقدار یعنی 999 شروع کنیم به 1 میرسیم پس حتما جواب داریم و تا ابد جابجا شویم چون بلاخره ممکن هست به یک local minimum or flat برسیم. تست بعدی: به احتمال 1/n از حالت شماره 1 مسئله را شروع میکنیم مشخص هست دیگر چون تصادفی هست هر کدام احتمال 1/n دارد. و دقت کن چون مرتب تابع هدف 1 به 1 بیشتر میشود هیچ موقع گیر local maximum نمیشویم. بنابراین حتما n-1 یال داریم و چون مسیر خوبی هست همیشه در شروع تپه نوردی اول به جواب میرسیم و شروع مجدد نیاز نداریم، و NF مساوی 0 هست و P مساوی 1 هست چون دقیقا به جواب میرسیم و ما فقط NS را داریم حساب میکنیم همین برای حالت دوم و همین برای حالت سوم و همین برای حالت n-1 که با یک گام به جواب میرسیم. تست بعدی: دقت کن باید از توزیع برنولی استفاده کنی و توزیع دو جمله ای استفاده کنی که مقدار p و متغیر هر کدام 2/15 خواهد بود. پس همانطور که میدانیم مقدار expected در متغیر های توزیع دو جمله ای میشود n\*p، که n میشود 4 و p هم 5/15 یا همان 1/3 هست. مشخص هست دیگه این گزینه ما هر بار احتمال دارد به احتمال 1/3 اولین نفر را ببینیم 4 بار انجام میدهیم پس میشود 4 ضربدر 1/3، Y` بزرگتر مساوی 1 یعنی اینکه حداقل 1 بار انتخاب شود، اینجوری فکر کن احتمال اینکه حداقل 1 بار انتخاب شود چه قدر هست؟ 1 منهای احتمالی که اصلا انتخاب نشود. تست بعدی: دقت کن 3 بار جستجو کردیم ممکن هست در همان جستجو اول به جواب برسیم که میشود p و ممکن هست در دومی به جواب برسیم که میشود احتمال اینکه در اولی به جواب نرسیم و در دومی به جواب برسیم پس (1-p)\*p و برای سومی هم همینطور. فروشنده دوره گرد: یک گراف بسته همه یال ها را ملاقات کند و بین مسیر های مختلف کمترین مسیر را انتخاب کند بر اساس طول یال ها. یعنی دور همیلتونی با کمترین هزینه را پیدا کنیم. در این مسئله هر حالت مسئله یک ترتیبی از نقاط هست که ملاقات کردیم تابع هدف دنبال کمترین مسیر ممکن هستیم. فاصله دو نقطه در جزوه آمده است. همسایگی چی؟ جای دو تا نقطه را با هم عوض کن تا یک حالت جدید تولید کنی. چند تا حالت دارد؟ انتخاب 2 از 6 تا دیگه چون جای هر کدام از این 6 تا نقطه را عوض کنی یک حالت جدید میدهد و این مسئله بزرگی هست چون هر راس 15 تا همسایه دارد و بزرگ هست. برای جابجایی هم گفتیم 2 تا نقطه را هم بدست بیاور تابع هدف را چطوری حساب میکنی؟ مسیر ها را جمع میزنی فقط. مثلا تپه نوردی هر 15 تا را تولید و کمترین هزینه را برمیدارد. اگر بخواهیم از الگوریتم های فصل قبل استفاده کنیم میتوانیم از فرموله سازی افزایشی استفاده کنیم و یک به یک راس اضافه میکنیم پس میتوانیم از سرچ ها استفاده کنیم هیورستیک هم پس داریم از هیورستیک MST استفاده میکنیم که درخت پوشای کمینه هست. سوال شریف: 5 فاکتوریل چون فرموله سازی کامل هست و فقط یکی پر معنا ترین هست. همسایگی خوب این هست که جای 2 تا کلمه را با هم عوض کنیم. حالت درست یک جایگشتی از 5 تا کلمه هست پس نمیتوانی اینطوری cross over کنی که این ترم، این ترم است برای همین یک روش دیگر این هست که بیایم جای این لغات را در جمله پیدا کنیم و به همون ترتیب اون یکی والد قرار بدهیم. دقت کن cross over ها میتواند متفاوت باشند.

**نکات فصل پنجم:**

دقت کن که الگوریتم minimax اول عمق رفتار میکنه ولی کامل هست و به برگ میرسد اگر دور باشد گیر میکند پیچیدگی حافظه ایش خوبه ولی زمانی نه. مقادیر الفا بتا پدر به فرزندان به ارث میرسد. بعد از بدست آوردن مقدار برگ برمیگردیم بالا. الفا همیشه میخواد زیاد شه و بتا همیشه میخواد کم شه. دقت کن جز مقدار گره ریشه بقیه گره ها میتوانند مقادیر متفاوت داشته باشند. در شانسی دقت کن حریف قطعی هست انتخاب ما شانسی هست. بعد هم میانگین وزن ها را میگیریم. دقت کن در max آلفا تغییر میکند و در min بتا و در min الفا ثابت است و همچنین در max بتا ثابت هست. دقت کن الفا نمیتواند کمتر از مقدار فعلی که گرفته شود همینطور بتا نمیتواند بیشتر از مقدار فعلی که گرفته شود.

Min میخواهد مثبت بینهایت را هعی کم کند و max میخواهد منفی بینهایت را مرتب زیاد کند. حریف میخواهد با کمترین انتخاب سودمندی ما را کم کند و ما میخواهیم با انتخاب بیشترین مقدار سودمندی خود را بیشتر کنیم. هرس هم فهمیدی دیگه ما 2 را دیدیم از 3 کمتر هست پس مشخص هست بقیه هر چه قدر هم بیشتر باشد تاثیری ندارد چون حریف در نهایت همین 2 را برمیدارد. ولی 14 هرس نمیشود چون 14 از 3 بزرگتر هست اگر d2 ما 1 بود چون حریف همان 1 را برمیداشت d3 هرس میشد یعنی شاخه های پایینی تاثیری در جواب های نهایی ندارند پس چک نکنیم. شرط هرس در گره min این هست که اون مقدار کمتر از الفا باشد و در گره max این است که اون مقدار بزرگتر از بتا باشد. الفا بهترین انتخاب گره مکس بود و بتا بهترین انتخاب گره مین بود. مشکلات قطع جستجو یکی این بود که یکی حرکات غیر ساکن بود یعنی چون ما تا یک عمقی حرکات را پیش بینی میکردیم و دیگر نگاه نمیکردیم ممکن بود بعد اون یک حرکتی حریف بزند که چون ما قطع کردیم آنرا ندیده ایم. مشکل دوم این بود که اثر افق هست یعنی یک کار بد را مجبوریم انجام بدیم ولی میتوانیم وقوع آنرا به تاخیر بیندازیم. دقت کن رفتی تا پایین مقدار نهایی را واسه گره مشخص کنی. فلسفه هرس این است که اگر از زیری ها نمیتوانی بالایی را بهتر کنی هرس کن.

Expect minimax میاد بیشترین سودمندی ممکن بیشتر از هر استراتژی دیگر را بدست میاورد. هرس چیز هایی را هرس میکند که ریشه را نمیتواند تغییر بدهد. v که گرفتی تا ته برو بالا حتی اگه اپدیت کردی یا نکردی چه الفا چه بتا رو. دقت کن minimax یک الگوریتم جستجو هست چون میخواهد راه حل بدهد. برش آلفا بتا هم همین هست چون خروجی میدهد که چیکار کنیم. دقت کن اگر گره شانس را داشته باشیم حتما باید مقادیر کران بالا و پایین را بدانیم چون میانگین میگیرند. هرس آلفا بتا هم نداریم ولی ایده آن را داریم. دقت کن سمت چپ ترین زیر شاخه هرس نداریم چون یک ممکن هست اون یکی برگ هم مقدار min را تغییر بدهد و هم از سایر شاخه ها بدست نیاوردیم. تو شانسی ها فکر کن در هر لحظه این عدد میتواند بالایی ها را تغییر بدهد یا نه به محض اینکه فهمیدیم تغییر نمیدهد هرس میکنیم. در تست ضرب دقت کن اگر در عدد منفی ضرب کنیم دیگر درست نیست چون ترتیب ها فرق میکند ولی عدد مثبت مشکلی ندارد. دقت کن حریف بهینه عمل نکند ممکن هست استراتژی بهتری از minimax وجود داشته باشد.

دقت کن minimax را با فرض zero-sum بودن انجام میدهیم و گرنه جواب غیر بهینه میدهد چون بر اساس zero-sum بودن جواب میدهد. دقت کن مقداری که در برگ مینویسیم صرفا برای ریشه هست که max سعی میکند اون مقدار را بیشینه و min سعی میکند اون مقدار را کمینه کند. ریشه max هست و دقیقا مقدار همین در برگ هست. دقت کن min یعنی حداکثر سود حریف معادل حداقل کردن سود ما هست واسه همین min هست. دقت کن در گره های شانس حتما باید کران پایین و بالا را ارزش گذاری کنیم چون داریم میانگین میگیریم، و باید حد پایین و بالا مقادیر سودمندی را بدانیم. اگر حریف حرکات اشتباه انجام دهد ممکن است مینی مکس بهترین استراتژی نباشد و سیاست بهتری وجود داشته باشد. دقت کن در تست کنکور مهندسی آیتی 98 مهندسی گزینه سوم اکثرا درست هست وقتی درست نیست که ما چند تا استراتژی بهینه داشته باشیم و در یکی هرسی بکنیم که آن جز گره های بهینه یک استراتژی دیگر باشد. گزینه 4 ام هم همین هست یعنی اکثرا درست هست مگر چند تا استراتژی بهینه داشته باشیم. برای کشیدن درخت ببین چه اکشن هایی خارج میشود آنها را به صورت یال خارج کن. کلا یال ها اکشن ها هستند هر جا نتیجه اکشن را ندانیم گره شانس میگذاریم. در گره های شانسی هرس به این شکل هست که تو باید ببینی مقدار این گره هایی که میخواهی ببینی تاثیر در مقدار ریشه دارد یا ندارد اگر قرار هست بین از یک طرف منفی 4 ببینی و از طرف دیگر بیشترین حد مثبت 5 هست مقدار اون را حساب کن ببین اگر بیشتر از 1 شد و باعث تغییری در بالاتر شد که هرس نکن ولی اگر نشد کلا بزن هرس کن. دقت کن حتی اگر حریف بهینه نباشد بازی مینی مکس یک اطلاعاتی میدهد با اینکه این شکلی طراحی نشده باشد و دقت کن که اون کمترین سودمندی یک حد پایین خواهد بود. بهترین ترتیب این هست که کمترین مقدار برای مین ها اول و بیشترین مقدار برای ماکس ها اول. دقت کن هیچوقت الفا بیشتر از بتا نمیشود چون قبلا هرس شده است پس اشتباه هست. دقت کن مقادیر گره های میانی بعد از هرس ممکن هست متفاوت باشد فقط ریشه هست که تغییر نمیکند. دقت کن اگر گره های مین را با گره های شانس عوض کنی مقدار ریشه بیشتر میشود چرا چون اونا دارند میانگین میگیرند و مشخص هست بیشتر میشود. دقت کن وقتی m توی توان باشد دیگر خطی بر حسب m نمیشود. در گره های min حاصل بعد از هرس یک عدد بزرگتر مساوی اون مقدار v واقعی هست و در ماکسیموم ها بعد از هرس یک عدد کوچکتر مساوی اون مقدار واقعی v هست. دقت کن ما هیچوقت کمتر از مینی مکس واقعی نمیشویم چون میانگین اون هست و اتفاقا بعد از هرس اون عدد بیشتر هست. آلفا هعی بیشتر میشود و هیچوقت کم نمیشود و بتا هعی دارد کم میشود و هیچوقت زیاد نمیشود. M عمل ممکن یعنی ضریب انشعاب m هست. واسه همین سوال گزینه 4 اگر میگفت اعداد منفی هستند چون یک حد بالا داشتیم پس جواب غلط میشد. دقت کن در شانسی ها وقتی احتمال هر یال را نداند چیز کن میانگین بگیر.

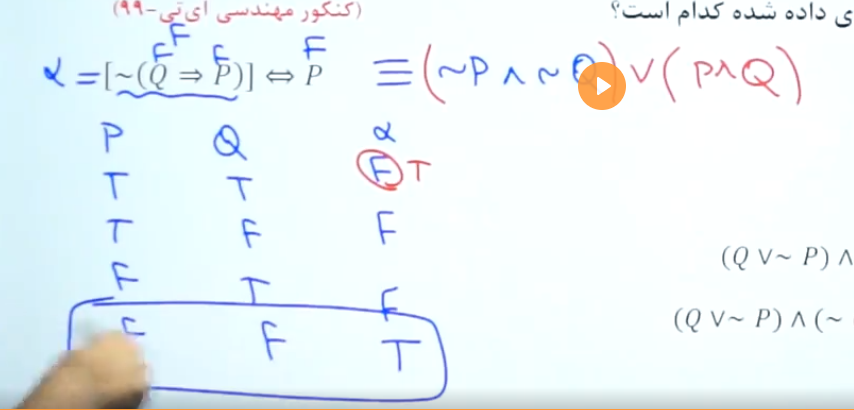
**نکات فصل پنجم- ارضای محدودیت CSP:**

ترتیب مقدار دهی اهمیتی ندارد در مسائل CSP، مرتبه زمانی عمق اول dm هست و همین تو مسئله CSP هست چون این شکلی پر میکنیم و میخواهیم این را کم کنیم. دقت کن اگر قرار باشد دنبال همه جواب ها باشیم ترتیب مقدار دهی بی مورد هست ولی وقتی فقط دنبال یک جواب هستیم چون میخواهیم فقط به آن برسیم پس ترتیب مقدار دهی مهم هست و LCV میگوید بیا به اونی بده که کمترین موارد را از دامنه سایر حالات کم میکند. هر یال حداکثر D بار سازگار یا پردازش میشود. ترتیب بررسی یال ها تاثیری در نتیجه نهایی AC-3 ندارد، اگر دامنه AC-3 دامنه یک حالت خالی شد یعنی جواب نداریم قبل سرچ هست به صورت پیش پردازش. در AC-3 در اول در صف تمام یال های گراف حضور دارند، اما واسه MAC این شکلی نیست چونکه به ازای اون مقادیر که یک سر مقدار گرفته دارند فقط یال های آنها را تو صف میگذارد بعد دقت کن MAC از فوروارد چکینگ قوی تر هست تو همین MAC هم همه اونایی که به یک گره ای متصل باشند بعد یک مقدار از دامنه آن حذف شده باشد دوباره به صف اضافه میکنیم، دقت کن به ازای یال های موجود در صف سازگار میکنیم. دقت کن اگر انتساب سازگار دوتایی نباشد که بشود به یک انتساب سازگار 3 تایی برسیم این باعث نقض در سازگار 3 تایی نمیشود چرا که اصلا چیزی وجود ندارد که بخواهد باعث از بین رفتن اون سازگاری شود ولی 2 سازگار نیست. مسئله میتواند 2 سازگار باشد ولی 3 سازگار نباشد، مسئله میتواند نه 2 سازگار باشد نه 3 سازگار. در محلی ها باید فرموله سازی حالت کامل انجام بدهیم. اگر گراف محدودیت درخت باشد یعنی دور نداشته باشد در زمان خطی میتوانیم حل کنیم در درختی ها سازگاری باید از پایین به بالا انجام شود درخت را هم حتما باید ریشه دار کنیم کلا میخواهیم یک ترتیب توپولوژیکی بدست بیاوریم یعنی یک گره انتخاب کن تمام یال های خارج شده از آن را بکش بعد برو به فرزندان اون رو بکن ریشه همین کار را برای اون بکن بعد از این مرحله میایم سازگار میکنیم از پایین به بالا یعنی از اون گره هایی که متصل به برگ ها هستند بعد بالایی های آنها این الگوریتم backtracking نداریم و همیشه مقداری در دامنه متغیر ها هست که سازگار باشد. پس ایده درخت خوب هست پس ما میتوانیم گراف ها را هم به درخت تبدیل کنیم. یک ایده این هست که یک یا چند تا گره برداریم آنها را حذف کنیم و حاصل را که تبدیل به یک درخت شده است سازگار کنیم که بهش cycle cutset گفته میشود. یک ایده دیگر این بود که یک سری ابر گره تعریف میکردیم و هر ابر گره یک تیکه از مسئله هست. و ابر گره ها باید مقادیر سازگار داشته باشند. دقت کن صفر سازگاری نداریم. یک سازگاری داریم که واسه یک گره هست مثلا a دوست ندارد قرمز باشد. اگر حتی یک یال 2 سازگار نباشد یعنی کل مسئله 2 سازگار نیست. در روش پرش به عقب یا back jumping وقتی دامنه یک متغیر تهی شد برمیگردیم به آخرین متغیری که در conflict set آن وارد شده است و یک مقدار دیگر را برای آن چک میکنیم، و کلا وارد یه شاخه دیگه میشویم و همه متغیر ها یک نوع دیگر مقدار دهی میشوند. تو AC-3 وقتی دامنه یک متغیر خالی شد مسئله جواب ندارد و ناسازگار هم هست. دقت کن AC-3 با MAC فرق میکند. در AC-3 اگر دامنه تهی شود کلا جواب نداریم ولی در MAC فقط میگوییم صرفا در آن شاخه ما جواب نداریم. البته طراح ها فکر میکنند به این بستگی دارد در چه مرحله ای باشیم اگر در پیش پردازش باشیم کلا جواب ندارد و اگر در حال اجرا باشیم و همزمان با سرچ، روی یک شاخه فقط جواب ندارد و به عقب برمیگردیم. ترتیب متغیر ها گفتند که مثلا کدام رنگ آمیزی شود دو تا هیورستیک MRV و LCV را داریم. در MRV اگر چند تا گره اولویت یکسان داشتند بعدش سراغ درجه میرفتیم. در تست 94 ای تی میایم اول 4 را برمیداریم چرا چون کمترین دامنه را نسبت به سایرین دارد اگر دامنه همه یکسان باشد. دقت کن سازگاری یال همان کارهای فوروارد چکینگ را انجام میدهد و چیز های بیشتری هم حذف میکند. دقت کن اگر به یکی مقدار دادی دیگه یال های که منتهی به اون هستند و داری یک متغیر بهش نسبت میدی را دیگر اضافه نکن چون باید یک سرش مقدار متغیر نگرفته باشد مثلا در تست سال 95 اگر به q3 در رابطه با q1 مقدار بدهی و از دامنه q3 چیزی حذف کنی دیگر q1 به q3 را اضافه نمیکنی بلکه دو تای دیگر را اضافه میکنی. در همین تست دقت کن اونایی که همین اول حذف کردی جز فوروارد چکینگ بودند نه AC-3 اون بنفش ها منظور هست و قرمز ها توسط AC حذف شدند. کلا همه اونایی که یک سر متغیر گرفته یک سر نگرفته اضافه کن بعد شروع کن سازگار کردن. همیشه اول فوروارد چکینگ کن چون باقی روش ها حداقل فوروارد چکینگ را انجام میدهند اگر دامنه خالی نشد یعنی ناسازگاری نداشتیم. ALLDIFF دادند اینطوری چک کن که چند تا متغیر داریم و هر کدوم چند مقدار متمایز میتوانند بگیرند یعنی در تست دوم 96 ما 3 تا متغیر داریم ولی کلا 2 تا مقدار متمایز وجود دارد. دقت کن path consistency هم در این سوال قابل پیدا کردن ناسازگاری بود. اگر یک عدد ثابت d2 در n ضرب شود یعنی خطی هست بر حسب n، چون توان n 1 هست و n هم تعداد متغیر هاست. در گزینه ای تی دقت کن گزینه دوم خطی نیست نسبت به d بلکه مربعی هست نسبت به d. اگر درخت شد تو زمان خطی حل میشود. یافتن تجزیه درختی هم نمایی و ان پی هارد هست و خطی نیست و پیدا کردنش مسئله سختی هست. سعی کن از جایی حرکت کنی که بیشترین محدودیت را داری با اینکه فرقی ندارد از کجا شروع کنی. دقت کن وقتی strongly آوردند یعنی از k تا 1 باید سازگار باشد حتما نمیشود 2 سازگار نباشد ولی 3 سازگار باشد. همه جواب خواستی اهمیتی ندارد انتخاب ترتیب مقدار. اما برای ترتیب متغیر ها این قضیه فرق میکند دو تا سیاست داریم MRV OR LCV، یکی از سیاست های آن fail first هست یعنی سعی میکنیم زودتر شکست بخوریم که MRV هم هست این برای ترتیب متغیر ها هست. اما برای ترتیب مقدار دهی به متغیر ها سعی میکنیم اون مقادیری را انتخاب کنیم که شکست ما را دور تر کند و به موفقیت نزدیک تر کنید یعنی fail last یا LCV. ما n تا متغیر داریم که هر کدام d تا مقدار میتوانند بگیرند پس میشود d^n نه برعکس. خطی بر حسب n و مرتبه 2 بر اساس d. این موارد واسه تست 99 هست. cycle cutset یعنی دور نداشته باشد و درخت باشد. اینقدر باید حذف کنیم تا 3 تا گره نماند چونکه دور میشود باید 2 تا بماند. پس اندازه cycle cutset باید n-2 باشد چون 7 تا هم باشد باز دور داریم. اگر گراف محدودیت ما یک گراف کامل باشد، ایده تبدیل به درخت جواب خاصی به ما نمیدهد. پس تو AC بیا دو به دو بگیر سازگار کن با هر ترتیب دلخواه. در فوروارد چکینگ اولا بدون در نظر گرفتن هیچ چیزی فقط دامنه ها را بنویس بر اساس اون مقادیری که تا الان مقدار گرفته اند. node consistency همان 1 سازگاری هست. conflict set میشود آخرین متغیری که مقداری را از دامنه متغیر مورد نظر حذف کرده است که معمولا همسایه آن هست. دقت کن لزومی ندارد به صرف همسایه بودن یکی جز conflict set باشد بر اساس محدودیت ها باید گفته شود. هر یال حداکثر d بار سازگار میشود. فوروارد چکینگ در طول اجرا هست و صرفا با متغیر های همسایه کار دارد و با AC-3 معادل نیست. AC-3 فقط مقادیری را حذف میکند که مطمئنیم جز جواب ها نیستند. دقت کن حتی اگر کل جواب ها را بخواهیم باز ترتیب متغیر ها مهم هست برای هرس بیشتر. در گراف فقط با یک دور با حذف یک راس و بدست آمدن cycle cutset میتوانیم در زمان چند جمله ای حل کنیم و c=1 هست اندازه cycle cutset. گراف کامل d^n هست. دقت کن اونی را که حذف میکنیم به ازای d^1 حل میکنیم یعنی به ازای هر کدام حل میشود. یعنی 1 بار به اونی که قرار هست حذف شود مقدار میدهیم ممکن هست درخت حاصل تهی شود بعد باید نسبت به مقدار دومی که به اون گره حذفی میدهیم اجرا کنیم ضریب d برای همین هست و دقت کن مکعبی هست نسبت به d نه مربعی. چند تا عقب گرد خواهیم داشت؟ حداکثر d بار چون به ازای هر کدوم مقدار میدهیم ببینیم جواب دارد یا خیر. تالیفی: حتی اگر یک متغیر دامنش خالی شود یعنی جواب نداریم. AC-3 اجرا کردی اندازه دامنه بزرگتر از 1 شد ممکن هست جواب نداشته باشد و دقیقا 1 شد جواب دارد.

LCV میگفت اون مقداری را نسبت بده که کمترین محدودیت را برای سایر متغیر ها ایجاد کند. از جستجو محلی بخواهیم استفاده کنیم حتما باید انتساب اولیه بدهیم. Min conflicts میاد بین متغیر هایی که شرط را نقض کرده اند به صورت رندوم انتخاب میکرد، بعد بهترین مقدار ممکن را نسبت میدهد. دقت کن اون قضیه دامنه دقیقا یک مسئله جواب دارد واسه دودویی هست. فوروارد چکینگ بدون اتلاق رنگ خاصی فقط دو به دو چک و دامنه مشخص میکند همین. دقت کن وقتی دامنه بشود 1 و AC-3 نتواند ناسازگاری کشف کند بقیه الگوریتم ها هم مثل 3 سازگاری یا PATH و ALLDIFF نمیتوانند ناسازگاری کشف کنند. وقتی جوابی وجود دارد یعنی ناسازگاری نداریم که روشی بخواد آن را کشف کند. ممکن هست گرافی 2 سازگار نباشد اما اعمال 2 سازگاری هم ناسازگاری را کشف نکند و دامنه ای خالی نکند. دقت کن وقتی میگیم از نظر درجه وقتی MRV جفت مساوی شدن بره سراغ درجه، درجه متغیر هایی که هنوز مقدار نگرفته اند ملاک هست.

**عامل های منطقی:**

هر مدل یک انتساب یک مقداری به متغیر ها هست. به ازای n نماد ما 2n مدل داریم دقت کن نقیض یک نماد وقتی خودش هم باشد یک نماد جدا محسوب نمیشود. M alpha یعنی مدل هایی که توش الفا درست هست. الفا entail میکند بتا را یعنی از درستی الفا درستی بتا را نتیجه بگیریم. اینکه الفا نتیجه بدهد بتا را تو همه مدل ها منظور ما هست و تو همه مدل ها چک میکنیم اما اینکه بگیم که آیا مدل الفا درست هست یا نه میگیم تو چه مدلی میگی. کامل بودن یعنی اینکه اگر الفا از پایگاه دانش قابل استنتاج بود بگوید بله و اگر نبود مهم نیست چه بگوید. صحیح بودن یعنی اگر قابل استنتاج نبود نیاد بگه قابل استنتاج هست. دقت کن نماد شرطی واسه یک جمله خاص و یک مدل خاص هست و entail اصلا جز منطق نیست و اصلا همچین عملگری را نداریم واسه همین هست تو همه مدل ها بررسی میکنیم و جوابش بله یا خیر هست ولی شرطی میگیم تو چه مدلی مد نظرت هست. اگر الفا آنگاه بتا یعنی همان شرطی در همه مدل ها درست باشد یعنی ولید باشد یا تاتولوژی باشد میگیم الفا نتیجه میکند بتا را یا الفا entail میکند بتا را و اگر الفا نتیجه کند بتا را یعنی الفا آنگاه بتا ولید هست یعنی به ازای همه مدل ها ارزش درست دارد. دقت کن تهی بودن فالس باعث شد که اون فالس هر چیزی را نتیجه کند. نقیض ارضا ناپذیری میشود ارضا پذیری نه تاتولوژی. تو رزولوشن همزمان نمیتوانی دو تا نقیض را با هم بزنی و به تهی برسی. رزولوشن روی کلاز ها که or یک سری لیترال هست اعمال میشود. ولی برای الگوریتم رزولوشن باید پایگاه دانش به فرم CNF یا and، or ها یا ترکیب عطفی باشد یا and یک سری کلاز. و هر جمله ای را میتوان به فرم CNF تبدیل کرد. با رزولوشن میتوانیم تاتولوژی بودن یک مجموعه را ثابت کنیم چطوری نات الفا رو روش رزولوشن بزنیم تا به تهی برسیم در نتیجه ارضا ناپذیری آن اثبات میشود. هورن یعنی در کلاز ها حداکثر یک لیترال مثبت باشد و میتواند نباشد ولی عبارات معین میگویند حتما یک لیترال مثبت باید باشد یعنی هر عبارت معین هورن هست ولی هر هورنی عبارت معین نیست. زنجیره سازی جلو رو و عقب رو در زمان خطی میتوانند حل کنند در صورتی که رزولوشن تو زمان نمایی میتواند انجام دهد مخصوصا اگر 3 تایی باشد. Fact : اون تک لیترال ها هستند. جلو رو چک میکند اگر سمت چپ قابل ارضا باشد سمت راست را اضافه میکند. زنجیره سازی هم رو هورن هم رو عبارات معین کامل هست. برای رزولوشن اول اون اند ها را حذف کن اون 4 تا کار بکن یک سری اور بشود. جای رزولوشن بیا به گزینه طوری مقدار بده که الفا نات شود بعد ببین با همون مقادیر به صورت سوال به نات پایگاه دانش میرسی یا نه اگر رسیدی یعنی پایگاه دانش نتیجه میدهد الفا را. دقت کن جملاتی که تو پایگاه دانش هستند ارزش درست دارند. اگر تناقض بدست بیاد یعنی ما از نادرستی الفا نادرستی پایگاه دانش را نتیجه کرده ایم یعنی چی یعنی اینکه پایگاه دانش نتیجه میکند الفا را. از روی شرطی سعی کن به تناقض برسی. دقت کن جای اونا که نیستند مقداری بده که سعی کنی اون عبارت های پایگاه دانش درست شود. اگر نتوانی به هیچ تناقضی برسی و پایگاه دانش را نادرست نکنی یعنی پایگاه دانش آن را استنتاج نمیکند. برای تاتولوژی دنبال مدلی باش که آن را غلط کند. BC به حقایق نیاز دارد چون میخواهد بر اساس هدف به حقایق برسد. سرعت BC از FC بیشتر هست. ویژگی یکنوایی این هست که اگر یک جمله به پایگاه دانش اضافه کردی جملات قبلی هنوز قابل استنتاج هستند و درست هستند. در تست بعدی دقت کن چون KB عوض شده از قبلی نمیتوانی بگی A^B قابل نتیجه گیری هست. and دو تا چیز آنها را نتیجه میدهد برای همین جملات پایگاه دانش توسط پایگاه دانش نتیجه میشوند. در گزینه 4 دقت کن چون a, b از پایگاه دانش قابل نتیجه گیری هستند or آنها هم قابل نتیجه گیری هست. منطق مرتبه اول هم ویژگی همنوایی را دارد. در FC, BC باید عبارات هورن و عبارات معین باشند. هورن حداکثر یک لیترال مثبت دارد. دقت کن ما نات یک کلاز را در رزولوشن به پایگاه دانش اضافه میکنیم نه فرم CNF آن را. دقت کن لزومی ندارد پایگاه دانش تناقض نداشته باشد بلکه میتواند هم ارز با فالس باشد و هر چیزی را استنتاج کند. پایگاه دانش و حقایق باید به فرم CNF باشند. دقت کن جملات تک لیترالی نیستند. دقت کن مجموعه ای فکر کن چون اجتماع آنها افتاده داخل گاما پس هر کدام از آنها هم داخل گاما هستند. یه طور دیگه هم میشه فکر کرد یعنی اینکه تو هر مدلی که گاما درسته الفا اور بتا هم درست هست پس از این میشه نتیجه گرفت تو هر مدلی که گاما درسته الفا هم درسته و تو هر مدلی که گاما درسته بتا هم درسته. دقت کن اگر الفا یا بتا داخل گاما باشند و اون یکی نباشد قطعا اشتراک این 2 چون جز الفا یا بتا هست جز گاما هم هست. دقت کن DPLL دقیقا میاد مقداری که دادی را اعمال میکند روی بقیه بعد DPLL میاد از Unit clause و pure literal استفاده میکند اولی یعنی یک کلاز تکی باشد که نداریم در این تست و دومی میگوید یک لیترال فقط به شکل منفی یا فقط به شکل مثبت اومده باشد تا بتوانیم با آن تکلیف یک کلاز را مشخص کنیم که چون هیچ کدام از اینها امکان پذیر نیست میرویم به متغیر دوم مقدار بدهیم که b هست دقیقا کن به a دادیم و اول نات الفا را چک کردیم و جملات درست در اومدند و از سایر جملات اون الفا حذف شد چون خودش فالس بود. برای این 2 تا هیورستیک از چپ به راست بیا. یونیت کلاز میگوید فقط اون مقدار دهی را بررسی کن که کل عبارت را ترو میکند پس باید به سی مقدار فالس بدهیم که نات آن ترو شود. یونیت کلاز دقیقا میگوید اون شاخه از درخت که قرار هست به سی بدهی ترو و ارزشی برای تو ندارد را اصلا مقدار نده. ولی چون به خاطر این مقدار دهی فالس به سی تو جمله بعدی به مشکل میخوری یعنی تو این شاخه جواب ندادی و باید عقب گرد کنی به بی و مقدار ترو بدهی اگر تو مقدار بی ترو هم نتوانی پیدا کنی میروی برای الف چک کنی. دقت کن pure literal هم عین یونیت کلاز عمل میکند. جایی که به هیورستیک برسی درخت یک شاخه میشود. DPLL مسئله ارضا پذیر چک میکند، دقت کن اگر اون جمله ارضا ناپذیر KB ^ NOT Q را بدهیم و الگوریتم شکست بخورد و نتواند مدل درستی برای آن پیدا کند میفهمیم که KB میدهد Q را یا entail میکند. دقت کن DPLL فقط دنبال یک مدل هست که درست باشد تا بگوید این ارضا پذیر هست و ارضا پذیری را چک میکند نه ولید بودن و DPLL ولید بودن نمیدهد. اضافه کردن جملات جدید باعث تناقض و entail نشدن جملات قبلی نمیشود. تو منطق گزاره ای باید برای تک به تک خانه ها قوانین بنویسیم. اگر رو دو تا الفا و بتا رزولوشن بزنیم حتما یعنی کلاز هستند و نقیض هم داشته اند و or آنها حتما ترو میشود. True فقط عبارت همیشه درست را نتیجه میکند. دقت کن به ازای تمام جملات باید آن چیزی که entail میشود درست باشد تا true آن را entail کند اگر یکی پیدا کردی که نزد پس غلط هست. وقتی رو دو تا عبارت رزولوشن میزنیم یعنی از درستی اون دو تا عبارت گاما را نتیجه میگریم یعنی از اند اونها گاما را نتیجه میگیریم ولی برعکس آن درست نیست. اگر بگیم الفا entail نمیکند گاما را یعنی الفا برابر با false نبوده چون فالس هر چیزی را نتیجه میکند. ارضا ناپذیر یعنی اون عبارت برابر هست یا هم ارز هست با فالس پس اگر قرار باشد هم ارز با فالس باشد هر چیزی که آن را entail میکند هم باید فالس باشد. اگر دو تا عبارت تو همه عبارت ها فالس باشند یعنی ارضا ناپذیر باشند اور آنها هم ارضا ناپذیر هست. دقت کن که شرطی برای ولید بودن هست اگر الفا نتواند گاما را entail کند به این معنی نیست که شرطی آنها ارضا ناپذیر هست فقط به این معنی هست که شرطی آنها همیشه درست نیست همیشه ولید نیست و ممکن هست بعضی جا ها درست باشد ولی همیشه غلط نیست. ولی الفا entail میکند بتا را یعنی شرطی آنها مساوی هست با ولید. پس نمیتواند گاما آن را entail نکند یک چیز همیشه درست True را و هر جمله ای entail میکند True را. دقت کن در تست بعدی مثلا برای جمله اولی ما برای A,B هم که داخل اون جمله نیستند یک انتساب دلخواه داریم یعنی به ازای هر حالت از اون 3 حالت یک مقداری هم برای A,B داریم دو به توان 2 تا چون دو تا لیترال و هر کدام دو تا مقدار میتوانند بگیرند پس میشود 4 تا پس ما 3 تا حالت 4 تایی داریم. دقت کن به ازای تمام حالاتی که اون جملات ترو میشوند حساب میکنیم. دسته بندی میکنیم و میبینیم تو مثلا 3 حالت چطوری ترو میشوند. دو شرطی باید ارزش دو طرف یکسان باشد. یعنی جفت غلط یا جفت درست. دقت کن الگوریتم رزولوشن کامل هست نه خودش به این معنی نیست یک جمله بدهیم هر چیزی قابل استنتاج باشد بدهد نه. پس تمام عبارت های قابل استنتاج را نمیتوانیم بدست بیاوریم و همچنین چون فی ممکن هست هر چیزی باشد لزومی ندارد بتوانیم اصلا رزولوشن بزنیم ممکن هست اصلا نقیض هم نباشند. تفسیر همان انتساب مقادیر هست. همیشه یک انتسابی هست که بتوان نصف کلاز ها را ارضا کرد حتی اگر ارضا ناپذیر باشد نصف یا بیشتر. قطعا تعداد مدل های نات فی از خود فی بیشتر هست. در تست ای تی 98 دقت کن اگر فی 1 و 2 در یک مدل خاص درست باشند یعنی ارزش آنها یکسان هست ارزش نات آنها هم یکسان هست اما با هم، هم ارز نیستند اگر تاتولوژی بودند یعنی به ازای تمام مدل ها درست بودند با هم، هم ارز هم بودند. وقتی تو یک مدل ارزش یکسان دارند تو نات آن مدل هم ارزش یکسان دارند. Unit clause propagation یعنی به ازای هر یونیت کلازی که مقدار میدهی به طوری هست که هعی یونیت کلاز دیگر از کلاز ها تولید میشود و مقدار میدهیم میرویم جلو و این دقیقا شبیه forward chaining هست. چون forward chaining هم از فکت ها شروع میکند به حرکت کردن. ارضا پذیری معادل ولید بودن نیست. 2-sat در زمان چند جمله ای قابل حل هست و برای همه نمایی نیست و میتوانیم رزولوشن بزنیم. همه مسائل SAT قابل تبدیل به یک CSP هستند و زیر مجموعه ای آن هستند. اگر برای 3 تا متغیر که هر کدام 2 تا مقدار میتوانند بگیرند 3 تا درست باشد و کل میشود 2 به توان 3 یعنی هشت تا پس 5 تا غلط هست. دقت کن 32 تا حالت داشتیم برای X چون 5 تا متغیر داشت حالا 24 تاش درست بود پس باقی مانده آن غلط هست که میشود 8 تا. کلا تو این حالت سعی کن جملات را تا حد ممکن بشمری. تو سوال سال 97 اینطوری فکر کن دنبال مدلی باش که سعی کند کل جمله را سازگار کند اگر این کار را کردی دیگر غیر قابل ارضا نیست قطعا و بعد یه مدل پیدا کن که غلط باشد تا ثابت کنی همیشه ولید نیست. دقت کن اولش سعی کردیم یک مورد پیدا کنیم کل جمله غلط شود ولی نتوانستیم یعنی جمله دست کم ارضا پذیر هست. دقت کن چون نتوانستیم اثبات کنیم که ارضا ناپذیر هست پس ولید هست چرا چون تنها حالتی که فالس میشد را بررسی کردیم ولی نتوانستیم به آن برسیم. دقت کن تو سوال 99 سعی نکن فقط رزولوشن بزنی بیا این کار کن یه مقداری به اون جملات بده ببین ترو میشود یا نمیشود اگر شد یعنی به تهی نمیرسیم قطعا چرا اینکار رو میکنیم چون اگر انتسابی پیدا کنیم که ارضا ناپذیر باشد یعنی رزولوشن بزنیم قطعا به تهی میرسیم چه وقتی این اتفاق میفتد وقتی کل جمله را نتوانیم ترو کنیم و به تناقض برسیم. دقت کن چون اند دارد بین جملات و به شکل کلاز هم هستند میتوانیم راحت به CNF تبدیل کنیم و رزولوشن بزنیم. اگر ترو شد تهی نمیشود. نتیجه رزولوشن روی دو تا کلاز هورن قطعا هورن هست. چون حداکثر یک دونه لیترال مثبت داشتیم و قطعا اون هم حذف شده است و باز حداکثر یک لیترال مثبت داریم. روی یک عبارت معین و هورن رزولوشن بزنیم حاصل هورن میشود. کلاز معین کلاز هورن هم هست. نتیجه رزولوشن 2 تا عبارت معین هم حتما کلاز معین هست چرا چون 2 تا عبارت معین حداکثر 2 تا لیترال مثبت دارند و بزنی یک لیترال مثبت باقی میماند. هم ارزی یعنی جملات تو جاهای یکسان درست باشند.



دقت کن entail کردن جز جملات منطقی نیست و از لحاظ منطقی معنایی ندارد پس نمیتوانی از روی entail بگی تاتولوژی هست. تاتولوژی بودن برای جمله منطقی هست نه که f entail alpha تاتولوژی باشد این جمله صرفا میگوید به ازای همه الفا ها درست هست. دقت کن kb and not alpha یک جمله همیشه نا درست هست و برعکس آن تاتولوژی هست. اگر برای یک مدل هم الفا درست باشد هم بتا اونوقت alpha entail beta. ای تی 1400: دقت کن باید یک مقدار ناسازگار را حذف کنیم و وقتی حذف کردیم اصلا دیگر اون شاخه را چک نمیکند. دقت کن وقتی توسط forward checking فالس حذف میشود دیگر اون شاخه اصلا تولید نمیشود تو درخت DPLL یا هر درختی که داریم. دقت کن از تناقض حرکت میکنیم جملات گزینه ها را فالس میکنیم تا ببینیم آیا پایگاه دانش دچار تناقض میشود یا خیر.

تالیفی:

سوال اول: فرمت به این شکل هست که اگر به اولی ها فالس بدهیم بعد از اون تا ابد باید ترو بدهیم دقت کن اول فالسی که بدهیم ترو باید کلا بدهیم یا کلا فالس ندهیم و کلا ترو بدهیم این مسئله از مرتبه n^2 خواهد بود.

دقت کن اگر الفا و بتا تحت شرطی هر دو ارضا پذیر باشند یعنی تو دو تا شرطی مختلف نمیشود گفت اینها با هم، هم ارز هستند. هم ارز یعنی یک چیز باشند دقیقا. دقت کن اگر آنگاه هم لزوما درست نیست چون دو تا چیز مختلف هستند و یکی نیستند اگر یکی بودند اگر آنگاه درست بود. در همان تست گزینه سوم دقت کن میتوانیم از اند الفا و بتا نتیجه بگیریم که اور گاما و اون یکی درست هست و entail میکند. دقت کن در گزینه بعدی چون تو هر مدلی که بتا درست باشد آنگاه بتا اور گاما هم درست هست.

دقت کن اگر توانستی روی 2 تا عبارت رزولوشن بزنی یعنی clause هستند و اگر یک چیزی حاصل شد در واقع اون حاصل a and b بوده است و a b نه تنها کلاز هستند بلکه دو تا لیترال نقیض هم داشتند که توانستیم حذف کنیم و رزولوشن بزنیم. و دقت کن or این 2 تا که توانستیم رزولوشن بزنیم قطعا چیز درستی هست. دقت کن false همه چیز را entail میکند. دقت کن true ممکن هست حاصل را entail نکند.

منطق مرتبه اول:

نماد های ثابت و تابعی و متغیر ها جز ترم ها هستند. قانون کلی همیشه با نماد شرطی میاد یعنی سور عمومی همیشه با شرطی میاد. هر متغیر به نزدیکترین سور خود برمیگردد. در سور عمومی اگر فقط یکی بود و به طور کلی بود یکی پیدا کنیم که نقض کند دیگر ولید نیست ولی برای دو تا سور عمومی که دو طرف اگر آنگاه هستند کلا قضیه فرق میکند. اگر سور عمومی با اند بیاد جمله خیلی قوی تر از آن چیزی میشود که میخواستیم بنویسیم و مناسب نیست. دقت کن اگر سور وجودی با شرطی بیاد خیلی ضعیف میشود و منظور ما نیست دقت کن این به این معنی نیست هر جا این دو تا ترکیب رو دیدی خط بزنی نه ممکن هست سور عمومی بیاد و اون اند جز ترکیب جملات باشد. شرط های دیگر را مثلا یکی نیستند و حداکثر دو تا برادر دارد با اند نشان بده. تو منطق مرتبه اول هم سعی کن اگر دنبال جملات درست بودند سعی کن جایگذاری کنی با ترو فالس یعنی سعی کن جمله شرطی سمت چپ درست و سمت راست غلط باشد ببین میتوانی یا نه. وقتی سور داشتی داخل پرانتز قبل شرطی بود به احتمال خیلی زیاد تاتولوژی هست و سور ها بیرون بودند تاتولوژی نیستند.

استنتاج در منطق مرتبه اول:

دقت کن الگوریتم ها در منطق گزاره ای تصمیم پذیر بودند یعنی بلاخره متوقف میشدند یا جواب پیدا میکردند یا میگفتند نه ولی در منطق مرتبه اولی هیچ الگوریتمی وجود ندارد که متوقف بشود برای قسمت اول وجود دارد یعنی اگر قابل استلزام باشد میگوید بله ولی برای قسمت خیر مشکلی که داریم این هست نمیتواند و متوقف نمیشود مثلا رزولوشن در منطق مرتبه اول. برای حذف سور وجودی باید یک نماد جدید به نام اسکولم را بیاوریم و جای آن بیاوریم تا سور وجودی حذف شود اسکولم یک نماد ثابت هست. در سور عمومی وقتی حذف شود جملات حاصل هم ارز هست ولی واسه سور وجودی همچین چیزی درست نیست. هم ارز بودن باید هم الف entail میکند ب را هم ب entail کند الف را، در حذف سور وجودی از نظر استنتاج یکسان هستند اگر چه هم ارز نیستند.

استنتاج در منطق مرتبه اول:

اگر دیتالوگ داشته باشیم استنتاج در منطق مرتبه اول تصمیم پذیر هست ولی در حالت کلی این شکلی نیست. زنجیره سازی عقب به اتمام نمیرسد.

رزولوشن کامل هست و تهش به تهی میرسیم. دقت کن حتما باید به CNF تبدیل شود پس or های بین کلاز رو اون کلاز رو باید پخش کنی حتما. تو یکسان سازی وقتی با یک متغیر یکسان کردی بعدش اگر اون متغیر تغییر کرد به یک چیز دیگه یکسان شد قبلی ها هم باید اپدیت شوند. تابع باید یکسان باشد یک متغیر بود نباید در اون یکی جز پارامتر تابع باشد و دقت کن دو تا تابع باید یکسان باشد. دقت کن که در یک رزولوشن برای یکسان سازی نباید دو تا مقدار متفاوت بدهی اما میتوانی برای دو بار رزولوشن زدن دو تا مقدار بدهی برای هر رزولوشن فقط یک مقدار.

رزولوشن کامل هست یعنی هر چیزی که ما نتیجه میگیریم آن هم نتیجه میگیرد. برای استنتاج در منطق مرتبه اول یا همه باید به منطق گزاره ای تبدیل شوند حتما نه ولی یک راه این هست و بهینه نیست یا به فرم CNF هر چیزی غیر این غلط. یک متغیر با تابعی که اون متغیر توش هست نمیتوان یکسان سازی کرد. FC میاد BFS میزند و BC میاد DFS میزند برای اینها هم یکسان سازی میکنیم. تو FC سمت چپ ارضا میکنی سمت راست میدی تو KB. BC از سمت راست چک میکند چپ رو اضافه میکند. FC برای KB به شکل عبارت معین کامل هست دقت کن چه منطق مرتبه ای اول چه منطق گزاره ای فقط روی عبارات معین کامل هست. دقت کن تو یکسان سازی هر جا دیدی باید جایگذاری کنی نه فقط یکجا. نظیر به نظیر از چپ یکسان کن. استاندارد سازی قبل یکسان سازی انجام میشود یعنی متغیر هایی که با هم فرق دارند ولی اسم یکسان دارند را جا به جا کن. دنبال اون برو که بتونی این تابع را حذف کنی و دقت کن یک ثابت را نمیتوان با یک تابع یکسان کرد. اگر نتوانستی یعنی استنتاج نمیکند. FC میاد modes ponens میزند و روی عبارات معین کامل هست.