«به نام پروردگار»

گزارش پروژه سوم مبانی هوش مصنوعی

«حل مساله سودوکو به همراه زنگ آمیزی: CSP»

استاد: دکتر بهنام روشنفکر

دانشجو: محسن محمدیان

شماره دانشجویی: 9831502

بررسی اجمالی:

در این پروژه قرار هست برنامه ای نوشته شود که یک جدول سودوکو را حل نماید. منتهی باید رنگ آمیزی نیز در مساله دخیل است و هر خانه ی جدول علاوه بر آنکه در پایان عدد خواهد داشت، باید دارای رنگ نیز باشد به طوری که اعداد یک سطر و یا ستون مشابه هم نباشند و هیچ دو خانه ی مجاوری رنگ یکسان نداشته باشند.

علاوه بر این مساله برای رنگ ها اولویت نیز قائل هست و رنگی با اولویت بالاتر باید عددی بیشتر از همسایه های با رنگی با اولویت پایین تر داشته باشد.

فرموله سازی مساله:

برای فرموله سازی مساله باید سه پارامتر دامنه ها، متغیرها و محدودیت های بین متغیرها مشخص شود.

متغیرها:

متغیرهای ما در این مساله هر خانه از جدول خواهد بود. برای مثال اگر یک جدول 3*3 داشته باشیم، 9 متغیر خواهیم داشت.

دامنه ها:

دامنه های ما در این مساله یا میتواند در دو بعد باشد، یعنی رنگ و عدد را برای هر متغیر به صورت جداگانه درنظر بگیریم، یا یک بعد که این یک بعد با ضرب دکارتی همان دامنه رنگی و عددی یک متغیر به دست می آید.

$$D_{numeric} = \left\{d_{ij} = \{k\}\right\} : for\ k \in dimension\ of\ table$$

$$D_{colorful} = \left\{d_{ij} = \{k\}\right\} : for\ k \in color\ set$$

محدودیت ها:

در این مساله محدودیت ها به دو بعد محدودیت رنگ و عدد تقسیم می شوند و هر متغیر با متغیرهای هم ردیف و هم ستونش محدودیت عددی دارد یعنی مقدار عددی متغیرهایی که در یک سطر یا ستون هستند باید مخالف هم باشد. همچنین هر متغیر با متغیرهای همسایه اش محدودیت رنگی دارد و متغیرهای مجاور هم باید رنگی متفاوت داشته باشند.

یک محدودیت اولویت رنگ نیز داریم که متغیرهای مجاور یک متغیر، در صورتی که رنگ متغیر حاضر از متغیرهای مجاورش بیشتر باشد، متغیرهای مجاور باید عددی کمتر از عدد متغیر حاضر داشته باشند.

اگر بخواهیم به صورت نمادین نمایش دهیم خواهیم داشت:

$$
\begin{aligned}
Constraints = \{\ & alldiff(X_{11}, X_{12}, \dots, X_{1n}), \\
& alldiff(X_{21}, X_{22}, \dots, X_{2n}) \\
& alldiff(X_{31}, X_{32}, \dots, X_{3n}) \\
& \quad . \\
& \quad . \\
& \quad . \\
& alldiff(X_{n1}, X_{n2}, \dots, X_{nn}) \\
& alldiff(X_{11}, X_{21}, \dots, X_{n1}), \\
& alldiff(X_{12}, X_{22}, \dots, X_{n2}), \\
& \quad . \\
& \quad . \\
& \quad . \\
& alldiff(X_{1n}, X_{2n}, \dots, X_{nn})\}
\end{aligned}
$$

$$
\begin{aligned}
Color\ Constraints = \{\ & X_{i,j} \neq X_{i,j-1}, \\
& X_{i,j} \neq X_{i,j+1}, \\
& X_{i,j} \neq X_{i-1,j}\ , \\
& X_{i,j} \neq X_{i+1,j}\ \}
\end{aligned}
$$

توجه شود که محدودیت رنگ می تواند زیر مجموعه ی محدودیت عددی یک متغیر نیز درنظر گرفته شود.

شیوه پیاده سازی:

برای پیاده سازی از دو فایل پایتون استفاده شده است. یکی فایل csp.py که در آن کلاسCSP قرار دارد و شامل متغیرهایی نظیر محدودیت های عددی و رنگی متغیرها(خانه ها)، تعداد محدودیت هایی که متغیرها(خانه ها) دارند، state، متغیرهایی (خانه هایی) که تاکنون مقدار دهی نشده اند، دامنه عددی و رنگی متغیرها(خانه ها) و دامنه کلی هر متغیرمی باشد. (تقریبا همه ی متغیرهای موجود در این کلاس از type دیکشنری می باشند.

یک فایل main.py نیز داریم که تمامی توابع اینجا پیاده سازی شده است و الگوریتم در اینجا پردازش می شود.

توابع پیاده سازی شده:

:main()

این تابع، تابع اصلی ما می باشد و تمامی توابع در این تابع فراخوانی و handle می شوند.

:read_file()

این تابع مساله ورودی را از فایل text می خواند و state آغازین را تولید کرده و بر می گرداند.

:numeric_constraints_generator(dimension_of_table)

این تابع بعد جدول را گرفته و براساس آن محدودیت های عددی هرکدام از خانه های جدول را set می کند.

:colorfull_constraints_generator(dimension_of_table)

این تابع محدودیت های رنگی متغیرها را set می کند و در یک دیکشنری به نام ini_colorful_ constraints بر می گرداند.

:numeric_domain_generator(dimension_of_table)

این تابع براساس بعد جدول می آید و دامنه هر خانه یا متغیر را در دیکشنری num_domains مشخص می کند و بر می گرداند.

:color_domain_generator(dimension_of_table, txt_list)

این تابع بعد جدول و متن ورودی را می گیرد و براساس خط دوم متن ورودی (رنگ های ورودی) به هر متغیر دامنه ی رنگ اختصاص می دهد.

limiting_domain(dimension_of_table, start_state, num_domains, color_domains, ini_numeric_constraints, ini_colorful_constraints)

این تابع مانند الگوریتم ac-3 عمل می کند و باتوجه به ورودی، اندکی از محدودیت های قابل اعمال روی دامنه متغیرها را اعمال می کند تا جستجوی ما سریع تر به نتیجه برسد.

:create_num_of_constraints(ini_numeric_constraints, ini_colorful_constraints)

این تابع محاسبه می کند که هر متغیر در کل در چند تا محدودیت شرکت دارد و این مقادیر را برای هر متغیر در یک دیکشنری set می کند.

:combine_colorful_numeric_domains(num_domains, color_domain)

این تابع دامنه ی متغیرها را با ضرب دکارتی از دو بعد به یک بعد تبدیل می کند تا اختصاص دادن مقدار به آنها در خلال جستجو آسان تر باشد.

:mrv(domain_dic, unassigned_vars, num_of_constraints)

این تابع از میان متغیرهایی که مقدار دهی نشده اند، متغیری با دامنه انتساب کمتر را بر می گرداند، در صورتی که دو متغیر یا بیشتر اندازه ی دامنه ی یکسانی داشته باشند، فراخوانی تابع degree را بعنوان نتیجه بر می گرداند.

:degree(unassigned_vars, num_of_constraints)

این تابع متغیری را بر می گرداند که مقدار به آن انتساب نیافته باشد و در محدودیت های کمتری شرکت کرده باشد.

forward_checking(var_num_neighbors, var_color_neighbors, domain_dict, assigned_val)

این تابع با توجه به مقدار فعلی ای که به یک متغیر انتساب داده شده است، این مقدار را از دامنه ی همسایه های رنگی و عددی متغیر حذف می کند. یعنی رنگ منتسب یافته ی فعلی را از خانه های مجاور خانه ی فعلی وعدد انتساب یافته ی فعلی را از دامنه ی متغیرهای هم سطر و هم ستون با آن متغیر حذف می کند.

:backtrack(assignment_list, dimension_of_table, csp)

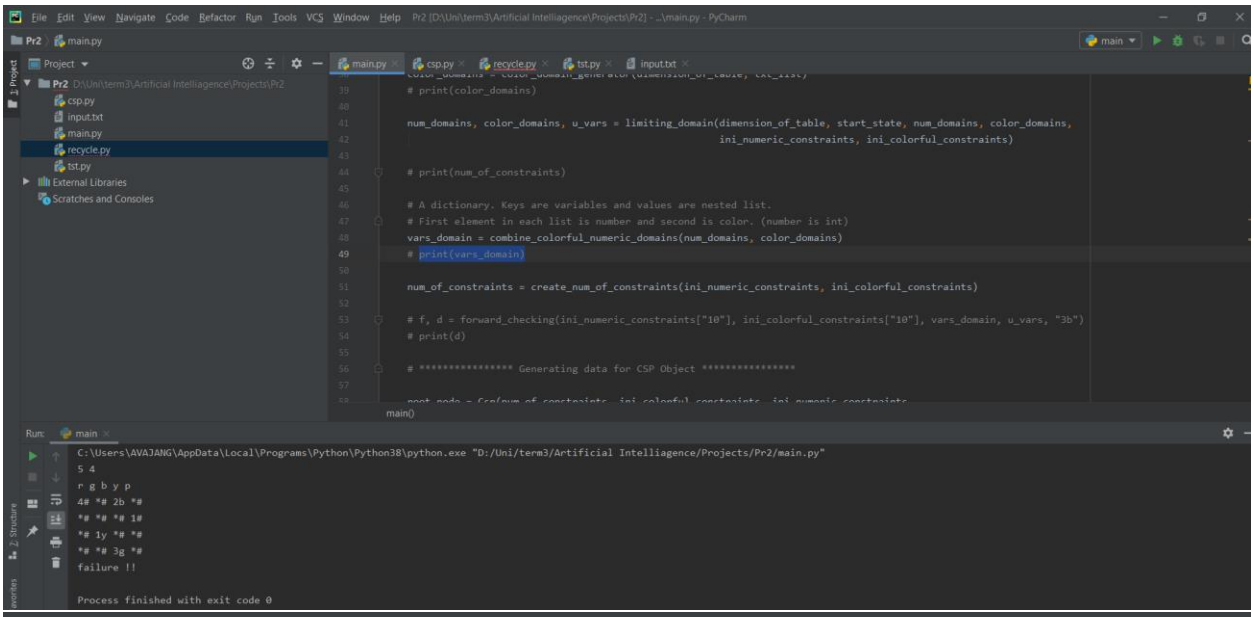
این تابع پیاده سازی الگوریتم backtrack می باشد و به این گونه کار می کند که ابتدا به وسیله تابع Mrv یک متغیر را از لیست متغیرهای بدون انتساب پیدا می کند. سپس در دامنه ی آن متغیر یک مقدار سازگار با متغیرهای انتساب یافته ی قبلی پیدا می کند و به آن نسبت می دهد و سپس تابع forward_checking فراخوانی می شود تا دامنه ها را محدود تر کند. در تابع forward_checking بررسی می شود که آیا دامنه ی متغیر بدون انتسابی تهی شده است یا خیر؛ اگر شده بود، تابع backtrack، "failure" بر می گرداند؛ اگرنه دوباره تابع backtrack فراخوانی می شود.

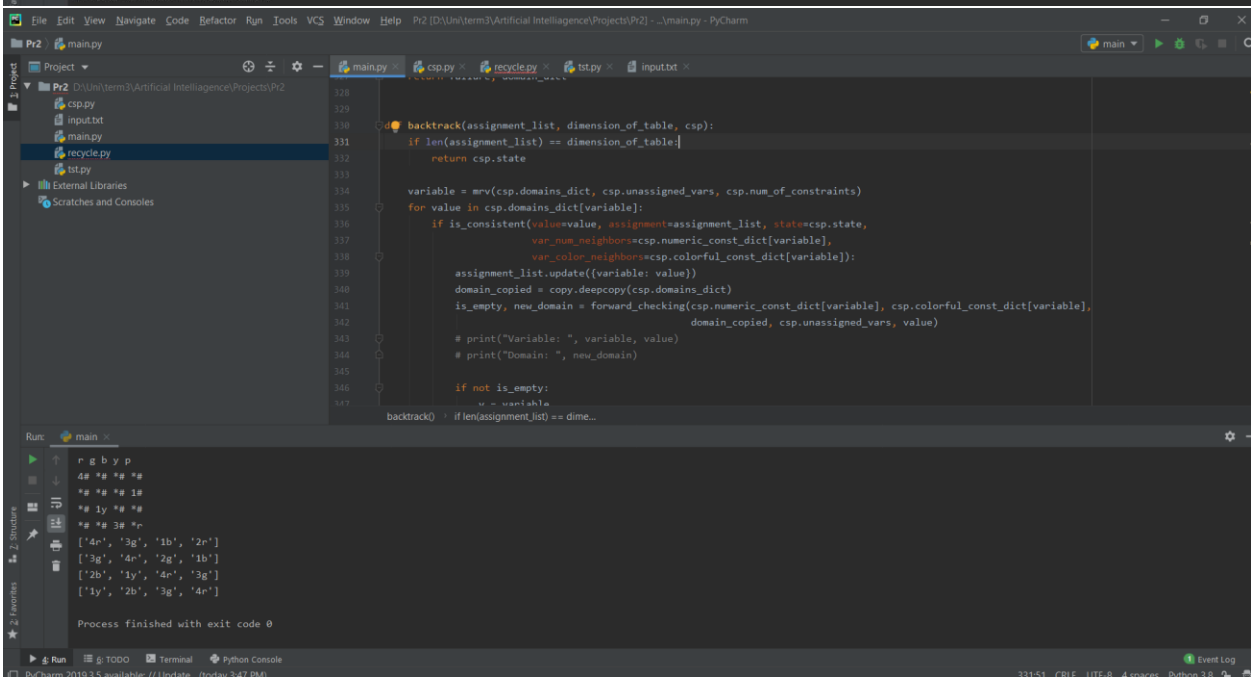
:is_consistene(vale, assignment, state, var_num_neighbors, var_color_neighbors)

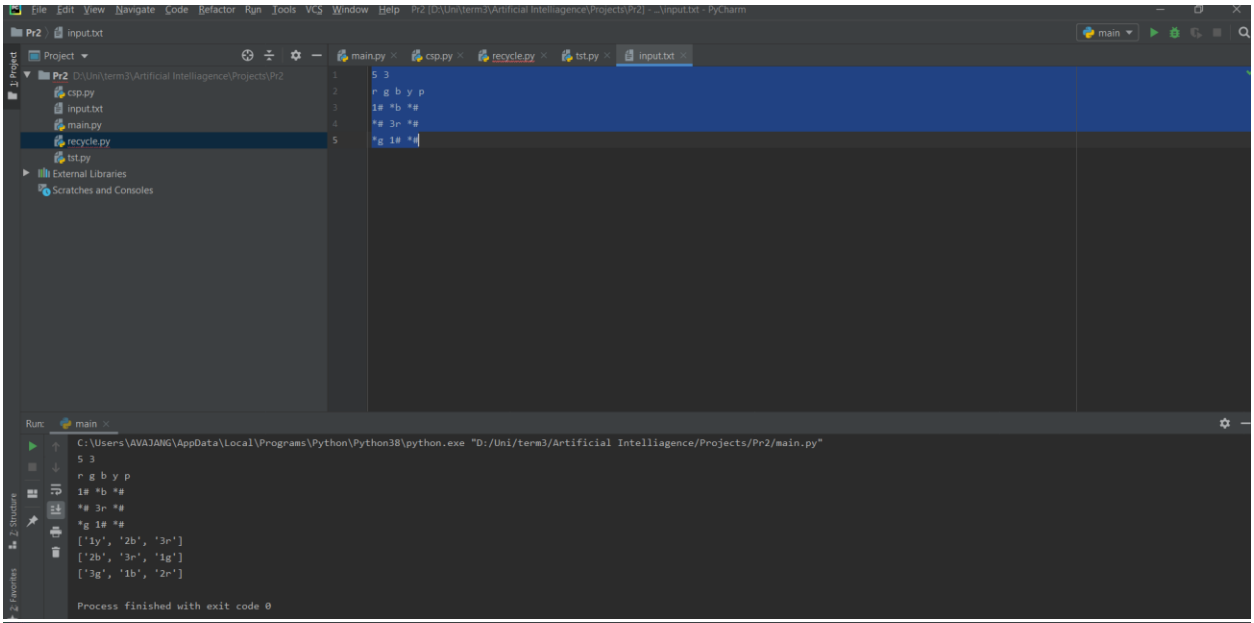
بررسی می کند که آیا مقداری که قرار است به متغیر فعلی نسبت داده شود، سازگار با متغیرهای انتساب یافته ی قبلی هست یا خیر. که در ورودی ما assignment لیست متغیرهای انتساب یافته و value مقداری که قرار است نسبت داده شود می باشد.

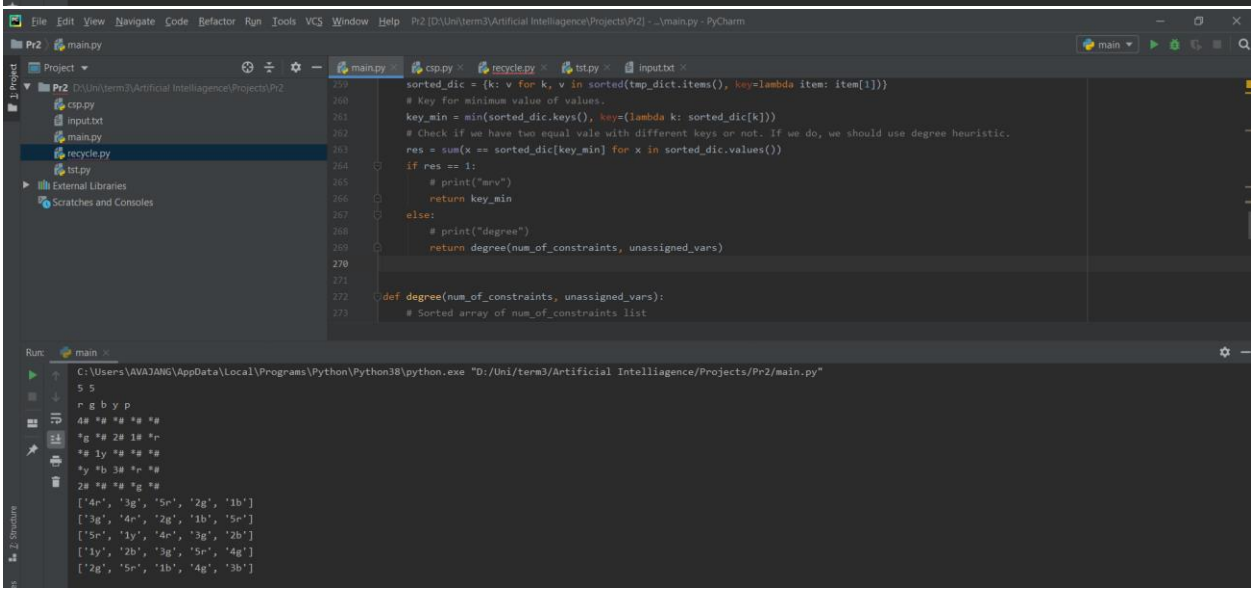
تمام سازگاری ها از قبیل محدودیت های عددی و رنگی و اولویت رنگ در این تابع به دقت بررسی می شود.

نتایج حاصل شده از برخی تست کیس ها در تصاویر زیر آمده است:









باسپاس از توجه شما