به نام خدا

گزارش پروژه دوم مبانی هوش

بهرام امامی

40231404

استاد درس:

دکتر جوانمردی

سوال 1: در فایل example.py، شروط سه گانه مسأله n وزیر به نحوه زیر آمده است:

الف) AllDifferent(): برای جلوگیری از قرار گرفتن دو یا چند وزیر در یک ستون مشترک است

ب) AllDifferent(q[i] + i for i in range(n)): برای جلوگیری از قرار گرفتن دو یا چند وزیر در یک قطر اصلی (قطری که از بالا چپ به پایین راست میرود) مشترک است.

ج) AllDifferent(q[i] - i for i in range(n)): برای جلوگیری از قرار گرفتن دو یا چند وزیر در یک قطر فرعی (قطری که از بالا راست به پایین چپ میرود) مشترک است.

سوال 2: دلیل انجام node-consistency : هدف اصلی انجام این مرحله، حذف مقادیر ناسازگار با unary checker از دامنه متغیر ها پیش از شروع حل مسأله است. این کار باعث میشود که حالت هایی که به بن بست میخورند و با شرایط اولیه ناسازگارند، مورد بررسی قرار نگیرند. اگر انجام نشود تمام این مسیر ها چک میشود و زمان جواب دهی افزایش میابد.

حالت 1: سودوکو مورد بررسی: peaceful sudoku

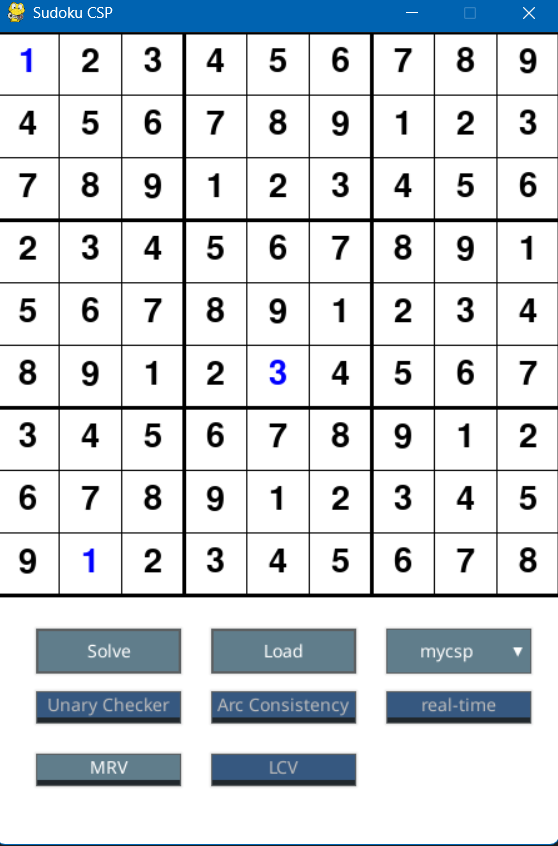
Unary checker: روشن

Arc consistency: روشن

MRV: خاموش

LCV: روشن

زمان: 



حالت 2: سودوکو مورد بررسی: peaceful sudoku

Unary checker: خاموش

Arc consistency: روشن

MRV: خاموش

LCV: روشن

زمان:

سوال 3: در DFS هر مسیری را تا انتها دنبال می‌کنیم، حتی اگر ناسازگار باشد؛ ولی در Backtracking، قبل از ادامه مسیر، در هر مرحله بررسی می‌شود که مقداردهی تا اینجا با قیدها سازگار هست یا نه. اگر ناسازگار بود، سریعاً به عقب برمی‌گردیم.

سوال 4: این شرط ها اینگونه به باینری تبدیل شده است که هر شرط را برای هر ترکیب دوتایی استیت ها بررسی میکنیم. مثلا برای بررسی عدم قرار گیری در یک ستون، میگوییم x1!= x2 , X1!= X3 , … , x8 != x9.

که تعداد این شرط ها میشود: n\*(n-1)/2

زمانی که شرط چندمتغیره از نوع AllDifferent یا مشابه آن باشد، می‌توان با ایجاد تمام ترکیب‌های دوتایی از متغیرها، همان مفهوم را پیاده‌سازی کرد.

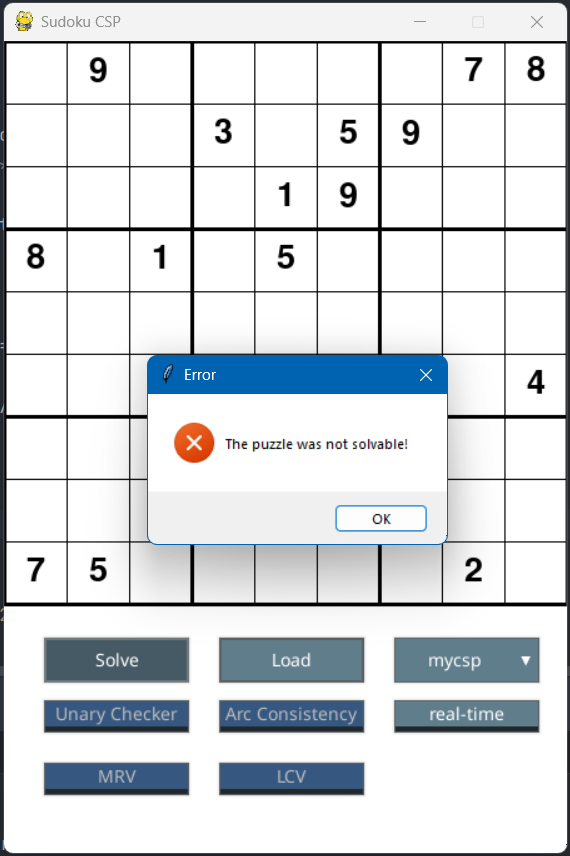
سوال 5: قبل از ورود به مرحله‌ی backtracking، با استفاده از AC-3، دامنه‌ی متغیرها بررسی و محدود می‌شود. اگر در این مرحله، دامنه‌ی یک متغیر خالی شود، یعنی مشخص می‌شود که هیچ مقدار سازگاری برای آن وجود ندارد. پس الگوریتم در همین مرحله متوقف می‌شود و اعلام می‌کند که هیچ جواب ممکنی وجود ندارد. در نتیجه، با صرف فقط چند چک ساده روی دامنه‌ها، به جواب نهایی می‌رسیم (بدون حل کامل).

امّا اگر از آن استفاده نکنیم، الگوریتم مجبور است وارد فرآیند کامل backtrack() شود. بسیاری از مسیرهای ناسازگار را امتحان می‌کند. زمان زیادی صرف مقداردهی‌های اشتباه می‌شود، تا در نهایت بفهمد هیچ ترکیبی جواب نمی‌دهد. این می‌تواند چندین برابر طول بکشد تا به همان نتیجه برسد.

بررسی زمان ها:real time خاموش است.

با Arc consistency: 

بدون Arc consistency: 

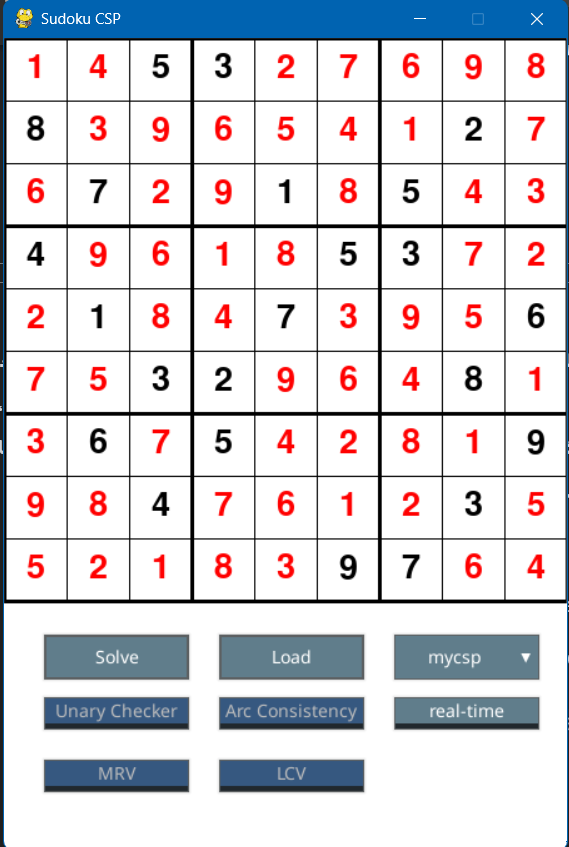


سوال 6: در این مسأله هر گره معادله یکی از 81 خانه این جدول 9\*9 است. برای یک خانه دلخواه باید مقادیر آن با مقادیر خانه های هم سطر، هم ستون، هم بلوک بررسی شود. که تعداد این خانه ها 8 تا در سطر، 8 تا در ستون، 8 تا در بلوک است. اگر از اشتراک این خانه ها چشم پوشی کنیم، در کل 20 تا داریم. یعنی هر گره ما با 20 گره دیگر ارتباط دارد و باید بررسی شود و یعنی 20 یال داریم از هر گره.

سوال 7: در این جا real time خاموش است.

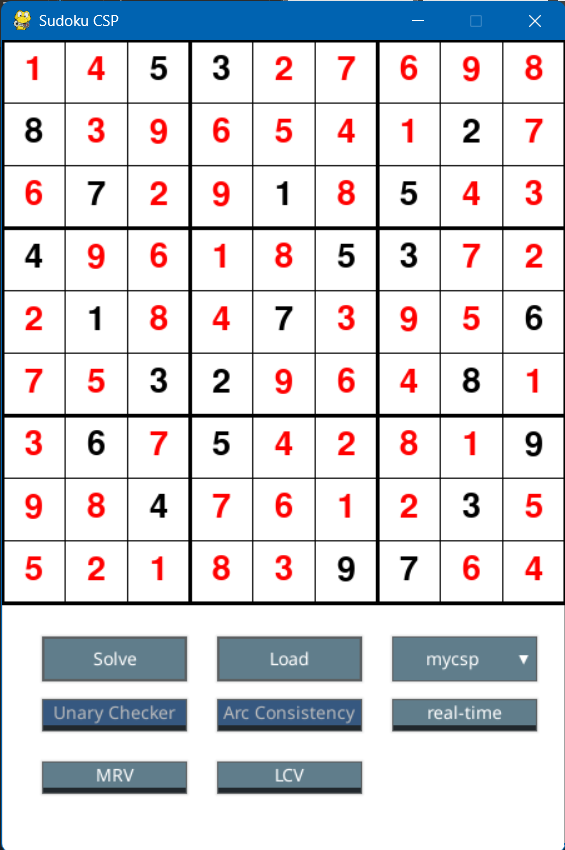
حالت 1: MRV و LCV روشن باشند:





حالت 2: MRV و LCV خاموش باشند:





در اینجا زمان تقریبا 2.5 برابر شد وقتی که از LCV و MRV استفاده نکردیم.

درخت جستوجو در این مسأله بسیار عمیق است. امّا اگر در هر مرحله انتخال بهتری داشته باشیم، میتوانیم سریع تر به جواب برسیم.

MRV متغیری را انتخاب میکند که دامنه کوچکتری داشته باشد که این کار باعث میشود که زودتر به ناسازگاری برسیم و مسیر را حذف کنیم و اینکه متغیرهای بحرانی زودتر مقدار بگیرند.

LCV مقداری را انتخاب می‌کند که کمترین محدودیت را برای همسایه‌ها ایجاد کند. یعنی اینکه مقدارهایی انتخاب می‌شوند که آزادی عمل بیشتری برای متغیرهای بعدی باقی می‌گذارند و عمق درخت جستجو کاهش می‌یابد.

در نهایت استفاده از هیوریستیک‌های MRV و LCV به‌طور مستقیم و قابل توجهی اندازه‌ی درخت جستجو را کاهش می‌دهند، و همین باعث می‌شود زمان حل به‌شکل چشمگیری کاهش یابد.

سوال8: در حالت حل با pycsp به سرعت جواب را داد. ولی وقتی از mycsp استفاده شد بسیار طول کشید تا جوال تولید شود.



در pycsp مدل با استفاده از کتابخانه‌ی PyCSP3 ساخته شده و حل آن توسط solverهای بسیار بهینه‌شده مثل Choco یا OR-Tools انجام می‌شود. این solverها شامل پیش‌پردازش‌های پیشرفته، نگهداری قیدها در ساختارهای خاص، و استراتژی‌های هوشمند حل هستند.

امّا در mycsp من از یک حل‌کننده‌ی دستی و ساده‌ی Backtracking استفاده کردم. درخت جستجوی بسیار بزرگی تولید می‌شود و به دلیل نبود pruning کافی، زمان حل بسیار زیاد می‌شود. حتی با MRV و LCV، سرعت همچنان بهینه نیست.

راه کار: استفاده از forward checking واقعی: اکنون ما فقط arc-consistency داریم؛ اما forward checking با pruning زودهنگام ترکیب می‌شود و بسیار سریع‌تر است. و یا افزودن constraint propagation به صورت نگهدارنده‌ی پشتیبانی‌ها. و یا استفاده از هیوریستیک های قوی تر مثل degree heuristic.

زمان mycsp: 



سوال 9: در این پروژه:

X: مجموعه‌ی متغیرها (خانه‌های سودوکو)

D: دامنه‌ی مقادیر برای هر متغیر (اعداد ۱ تا ۹)

C: مجموعه‌ی قیدها (مثل AllDifferent برای سطر، ستون، بلوک)

که در تمام سودوکو ها Xو C به هم یکسان بودند که X 81 خانه بود و C قیدهای شرطی یکسان نبودن در ستون و سطر و بلوک است.

در اینجا تفاوت در D است. هرچه قدر که تعداد خانه های ما از قبل مشخص تر باشد، D برای آن خانه ها یک عدد میشود. هرچه متغیر های کمتری از قبل مشخص شده باشند، دامنه متغیر های بیشتری کامل میماند و محاسبات طولانی تر و سخت تر میشوند. البته قرار گیری این متغیر ها در جدول نیز بسیار حیاتی است.