گزارش پروژه CSP

اعضاء گروه

- علی مجاهد (۹۸۱۲۷۶۲۵۵۴)
- محیا احسانی مهر (۹۸۱۲۷۶۲۳۲۷)

مدل سازی

در این پروژه در ابتدا تلاش شد تا بازی و قوانین آن را در قالب کلاس هایی در پکیج game توسعه دهیم. بعد از آن فارغ از مسئله ای که حل می کنیم و به صورت جنریک بلاک های توصیف کننده یک مسئله CSP را که برابر است با CSP (مسئله)، Variable Constraint, Domain در پکیج csp.problem ایجاد کردیم.

هر مسئله ای که بخواهد به فرم CSP درآید باید از کلاس انتزاعی CSP ارث بری داشته باشد و متد های انتزاعی آن را override کند.

توجه کنید که در کل پروژه منظور از PROBLEM_T نوع کلاس مسئله (خارج از فرم csp)، VAR_T نوع متغیر، DOMAIN_T نوع مقادیر مجاز برای متغیر ها است.

در این روش ما فرض می کنیم که تمام متغیر های ما مقادیری از یک جنس بگیرند.

نوع PROBLEM_T در مسئله ما برابر MagnetPuzzleBoard است.

متغیر های ما هر قطب یک آهنربا است و تایپ آنها Pole است و مقادیر مجاز آن ها PoleContent.

برای تبدیل مسئله پازل آهنربا به فرم csp کلاس MagnetPuzzleCSP که از کلاس CSP ارث میبرد را ایجاد کرده ایم.

در ابتدا الگوریتم Backtrack عادی را در کلاس BacktrackAlgorithm در پکیج csp.algorithm پیاده سازی کردیم که به شکل رندوم متغیر هایی را انتخاب می کند و یک مقدار را برای آن انتخاب می کند و به صورت بازگشتی برای متغیر بعدی این عمل را انجام می دهد و درصورتی که یکی از شرط ها دیگر برقرار نباشد از همانجا backtrack میکند و یک مرحله به عقب باز می گردد.

بعد از آن در کلاس HeuristicBacktrackAlgorithm در همان پکیج با استفاده از heuristic ها و الگوریتم های inference توانستیم تا حد زیادی روند حل کردن را بهبود ببخشیم.

برای انتخاب یک متغیر ابتدا از هیوریستیک MRV استفاده می کنیم به این صورت که متغیری بدون مقداری را برمیگردانیم که کمترین تعداد مقادیر مجاز را دارد. در صورتی که همه متغیر ها تعداد مقادیر مجاز برابری داشته باشند null برمیگردانیم. در این حالت برای از بین بردن حالت تساوی از DegreeHeuristic استفاده می کنیم به این شکل که متغیری را برمیگردانیم که در تعداد شرط بیشتری شرکت دارد.

برای گرفتن ترتیب مقادیر برای تست باید از کلاس MagnetPuzzleLCVHeuristic استفاده کنیم. به این صورت که این کلاس با در نظر گرفتن شرایط فعلی برد و مقادیر مجاز فعلی برای متغیر آن ها را به شکل یک لیست مرتب شده برمیگرداند.

از الگوریتم های Inference الگوریتم Forward checking و MAC (Maintaining Arc Consistency بر پایه AC۳) نوشته شده. که می توانید الگوریتم مورد نیاز خود را در کانستراکتور کلاس HeuristicBacktrackAlgorithm ست کنید. هر دو الگوریتم با توجه به توضیحات موجود در کتاب (بخش ۶.۳.۲ Interleaving search and inference) نوشته شده و باتوجه به نیاز های مسئله تغییرات جزئی اعمال شده.

الگوریتم AC3 به عنوان Propagation کارساز نیست زیرا در این مسئله به ازای هر مقدار برای یک متغیر در ابتدای کار قطعا یک مقدار معتبر در متغیر همسایه ی او وجو دارد .

این الگوریتم در مسئله آهنربا نیز به عنوان Inference کارساز نبود و حتی در صورتی که تست کیس اول با استفاده از forward checking در کمتر از ۰.۹ ثانیه حل شد اما با این الگوریتم به جواب نرسیدیم.

در ضمن، برای اجرای برنامه، لازم است کلاس Test ران شود.