به نام خدا

شرح پروژه ارضای محدودیت

مدل سازی و توابع :

در این پروژه مدل سازی توسط سه کلاس cell, constraint, cps انجام گرفته است.

هر خانه جدول تحت عنوان یک متغیر در نظر گرفته شده که مقادیر 1 تا 9 برای آن مجاز باشد و این موضوع در کلاس cell پیاده سازی شده است ، هر cell یک مقدار سطر و ستون، مقدار گرفته شده که میتواند 0 یا 1 یا مقدار دیگری بین 1 تا 9 پس از حل باشد و یک تایپ دارد که 0 به معنای خانه های خالی برای پر کردن و 1 به معنای خانه های از پیش پرشده میباشد و علاوه بر آنها یک لیست از اعداد 1 تا 9 به عنوان مقادیر مجاز در ابتدای برنامه برای هر خانه ست خواهد شد و یک استک از نوع لیست برای گرفتن کپی قبل از اعمال forward checking قرار دارد.

محدودیت ها در کلاس constraint چک میشوند و در این کلاس ما سه تا تابع به نام های countAssigned و findConstraint و isConstraint داریم که در ادامه به توضیح هر کدام خواهیم پرداخت:

تابع اول تعداد خانه هایی که در هر سطر و ستون از پیش مقدار داده شده اند را میشمرد، مثلا در مثال موجود در پی دی اف داک پروژه این تابع برای سطر اول مقدار 5 را برمیگرداند یعنی تمام خانه های سطر اول پر هستند و هرکدام از مقادیر شمارش شده را در خانه ی مربوط به همان سطر یا ستون در آرایه مربوط به خودش ثبت میکند.

تابع دوم هم مقداری که در هر سطر و ستون باید برابر شود را پیدا میکند که مثلا در ستون اول باید جمع ستونی مقادیر چه مقداری شود و این مقدار در کدام خانه قرار دارد و این دیتاها هم در ارایه های مربوط به سطر و ستون ذخیره خواهند شد.

تابع سوم هم همانطور که از اسمش پیداست یک خانه را میگیرد و بررسی میکند که مقداری که به این خانه نسبت داده شده است آیا محدودیت هارا نقض میکند یا خیر.

کلیه توابع مربوط به هیوریستیک ها در کلاس csp قرار گرفته اند.

در این کلاس ابتدا الگوریتم cspBacktracking اجرا میشود و در ابتدای ان فقط یک بار تابع ac3 صدا زده میشود و بعد از آن تابع MRVSelection برای انتخاب متغیر با کمترین مقدار باقی مانده صدا زده میشود و بعد از آن LCVOrdering هست که این خانه انتخاب شده را میگیرد و روی تک تک اعضای مانده در لیست مقادیر مجازش این بررسی را میکند که با گرفتن هر مقدار چند مقدار از لیست خانه های دارای محدودیت با آن حذف خواهد و در نهایت همان لیست مقادیر را با ترتیب متفاوتی از مقادیر باقی مانده بر حسب کمترین مقدار حذفی شمرده شده مرتب میکند و بازمیگرداند.

سپس در یک حلقه این مقادیر را منتصب میکنیم و اگر محدودیتی نقض نمیشد آنگاه یک کپی از مقادیر باقی مانده این خانه میگیریم و بعد تابع forwardchecking را صدا میزنیم و به صورت سطری و ستونی کلیه خانه های مرتبط با خانه انتخابی را بررسی کرده و اگر مقدار انتخابی را داشتند از لیست آنها حذف میکنیم و سپس بررسی میکنیم که تا به الان چه از نظر سطری چه ستونی جمع مقادیر خانه های تا به الان مقدار گرفته شده چقدر است و بعد از مقداری که با برابر ان باشد کم میکنیم و نام آنرا useless مینامیم و سپس از لیست متغیر های مقدار نگرفته در همان سطر و ستون مرتبط با متغیر انتخابی مقادیر بالاتر از useless را حذف میکنیم و البته قبل از همه این حذفیات از لیست مقادیر مجاز یک کپی در استک ذخیره میکنیم و بعد از fc اگر متغیری برابر 0 شد سایز لیستش ، آنگاه تابع undoForwarchecking را صدا میزنیم و همه خانه هایی که در fc دستخوش تغییر شده بودند را به حالت قبل از اعمال fc برمیگردانیم.

در نهایت هم که جواب پیدا شود آنرا چاپ میکنیم.

برنامه برای مپ 5*5 و 7*7 جواب را به طور کامل پیدا میکند ولی برای مپ 9*9 خیلی زمان زیادی لازم دارد و بعد از اعمال ac3 سرعت پیدا کردن جواب در مپ های گفته شده افزایش یافت و ابتدا متغیر هایی که دامنه آنها محدود به گرفتن 2 یا 3 عدد میشد مقدار گرفتند و روند حل مسئله سریعتر شد.

جواب مثال 7*7 پس از گذشت 3 دقیقه و 45 ثانیه :

```
solution is find
-1 25 32 21 29 33 36
21 1 2 3 4 5 6
27 2 9 1 3 4 8
29 3 1 4 8 6 7
30 4 8 2 6 1 9
30 6 7 5 1 9 2
39 9 5 6 7 8 4
```

جواب مثال 5*5 پس از گذشت 1 ثانیه :

```
solution is find
-1 23 16 10 -1
14 9 1 4 3
16 6 7 2 1
14 8 3 1 2
-1 8 5 3 -1
```