به نام خدا

گزارش پروژه حل سودوکو

استاد درس: مهندس روشن فکر

میلاد اسرافیلیان

۹۷۱۳۰۰۷

- **نحوه فرموله بندی مسئله:**

در این مسئله هر خانه از جدول سودوکو را به عنوان یک متغیر در نظر میگیریم.

دامنه هر متغیر در ابتدای مسئله شامل دو بخش می شود:

- رنگ: شامل تمامی رنگ هایی که در ورودی گرفته شده اند.
- عدد: شامل تمام اعداد ۱ تا n که در ورودی داده شده است.

محدودیت ها:

هر متغیر(خانه از جدول) شامل چند نوع محدودیت است:

- محدودیت بین متغیر های هر سطر و ستون: که هر متغیر مقدار عددی متفاوتی باید با متغیر های سطر و ستون متناظر خود داشته باشد.
- محدودیت خانه های همسایه: هر متغیر باید رنگی متفاوت با خانه های همسایه خود داشته باشد و همچنین رنگ هر متغیر باید متناسب با عدد آن باشد.(به گونه ای که عدد بیشتر باید رنگ با اولویت بیشتر داشته باشد)

- **توضیح کلاس ها:**

تنها کلاس موجود کلاس cell می باشد که جهت مدل کردن هر خانه از جدول در نظر گرفته شده است شامل چهار field :

Number: عدد در نظر گرفته شده برای آن خانه (در صورت نبود عدد -۱)

Color: رنگ در نظر گرفته شده برای هر خانه(در صورت نبود #)

numberDomain: دامنه در نظر گرفته شده برای اعداد این خانه (از ۱ تا n)

colorDomain: دامنه در نظر گرفته شده برای هر خانه(در ابتدا شامل تمامی زنگ های وارد شده)

- **توضیح توابع:**

```
def initialState(number:int, colors:list, inputData:list):
```

جهت مدل کردن حالت اولیه و ایجاد جدولی از cell با استفاده از داده های وارد شده رنگ ها و سایز جدول.(تشکیل دامنه و اختصاص مقادیر به متغیر هایی که مقدار دهی شده اند)

```
def isComplete(state: list):
    'Check to see if all variables are assigned'
```

جهت تشخیص اینکه جدول در حالت داده شده تماما مقدار دهی شده است یا خیر.

```
def MRV(state: list):
    'Minimum Remaining Value Heuristic'
```

جهت پیاده سازی هیوریستیک کمترین مقدار باقی مانده در دامنه.
به این صورت عمل می کند که تمامی متغیر های مقدار دهی نشده را میگردد و کمترین مقدار دامنه(ضرب اندازه دامنه اعداد و رنگ ها) را بر می گرداند.

```
def checkRowConstraint(state: list, row:int):
```

چک کردن محدودیت هر خانه با تمامی خانه های هم ردیف.(تعداد محدودیت هایی که هر متغیر با باقی دارد را برمیگرداند)

```
def checkColumnConstraint(state: list, column:int):
```

چک کردن محدودیت هر خانه با تمامی خانه های هم ستون.(تعداد محدودیت هایی که هر متغیر با باقی دارد را برمیگرداند)

```
def checkColorConstraint(state: list, row: int, column: int):
```

چک کردن محدودیت رنگ هر خانه با تمامی خانه های همسایه.(تعداد محدودیت هایی که هر متغیر با باقی دارد را برمیگرداند)

```
def degree(state: list):
    'Returns a cell row and column with minimum remaining value and most constraints with unassigned variables'
```

جهت پیاده سازی هیوریستیک درجه. از بین تمام خانه هایی که کمترین مقدار هیوریستیک MRV را دارند شماره خانه با بیشترین محدودیت ها را بر میگرداند.

```
def checkValid(state:list , cellRow:int , cellCol:int , num:int, color:str):
    '''Checks and returns if the chosen number and color values are valid for assigning based on
    Row and Column repetition and color priority of its neighbors'''
```

جهت چک کردن اینکه مقادیر ورودی برای خانه [cellRow,cellCol] محدودیتی را نقض می کند یا خیر(یا دامنه سایر متغیر ها خالی می شود یا خیر)

```
def numberFC(state: list, row: int, col: int, number: int):
    'Forward Checking to remove invalid values from cells in the same row and column numberDomain'
```

جهت انجام forwardChecking خانه های هم سطر و هم ستون و حذف مقادیر غیر مجاز از دامنه آن ها.

```
def colorFC(state: list, row: int, col: int, number: int, color: str):
```

جهت انجام forwardChecking برای خانه های همسایه و حذف مقادیر رنگ و عدد غیرمجاز از دامنه آنها(با توجه به اولویت رنگ ها)

```
def assignValue(state: list, row: int, col: int, num: int, color: str):
```

مقدار دهی به متغیر در صورتی که دامنه سایر متغیر ها خالی نشود(با توجه به انجام forwardChecking)

```
def backtrack(state: list):
```

جهت پیاده سازی الگوریتم بک ترک به صورت بازگشتی بر روی لیست دو بعدی ورودی.

- **نمونه ورودی و خروجی برنامه:**

حل کامل جدول ۵*۵:
ورودی:

```
5 5
r g b y p
*# *# *# *# *#
*# *# *# *# *#
*# *# *# *# *#
*# *# *# *# *#
*# *# *# *# *#
```

خروجی:

```
3g 5r 1g 4r 2g
4r 1g 3r 2g 5r
1g 3r 2b 5r 4g
5r 2b 4g 1y 3b
2b 4g 5r 3g 1y
```

حل تست کیس صورت پروژه:
ورودی:

```
5 3
r g b y p
1# *b *#
*# 3r *#
*g 1# *#
```

خروجی:

```
1y 2b 3r
2b 3r 1g
3g 1b 2r
```