دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

مبانی و کاربردهای هوش مصنوعی

پروژه دو

رادین شایانفر

زمستان ۱۳۹۹

پروژه دو

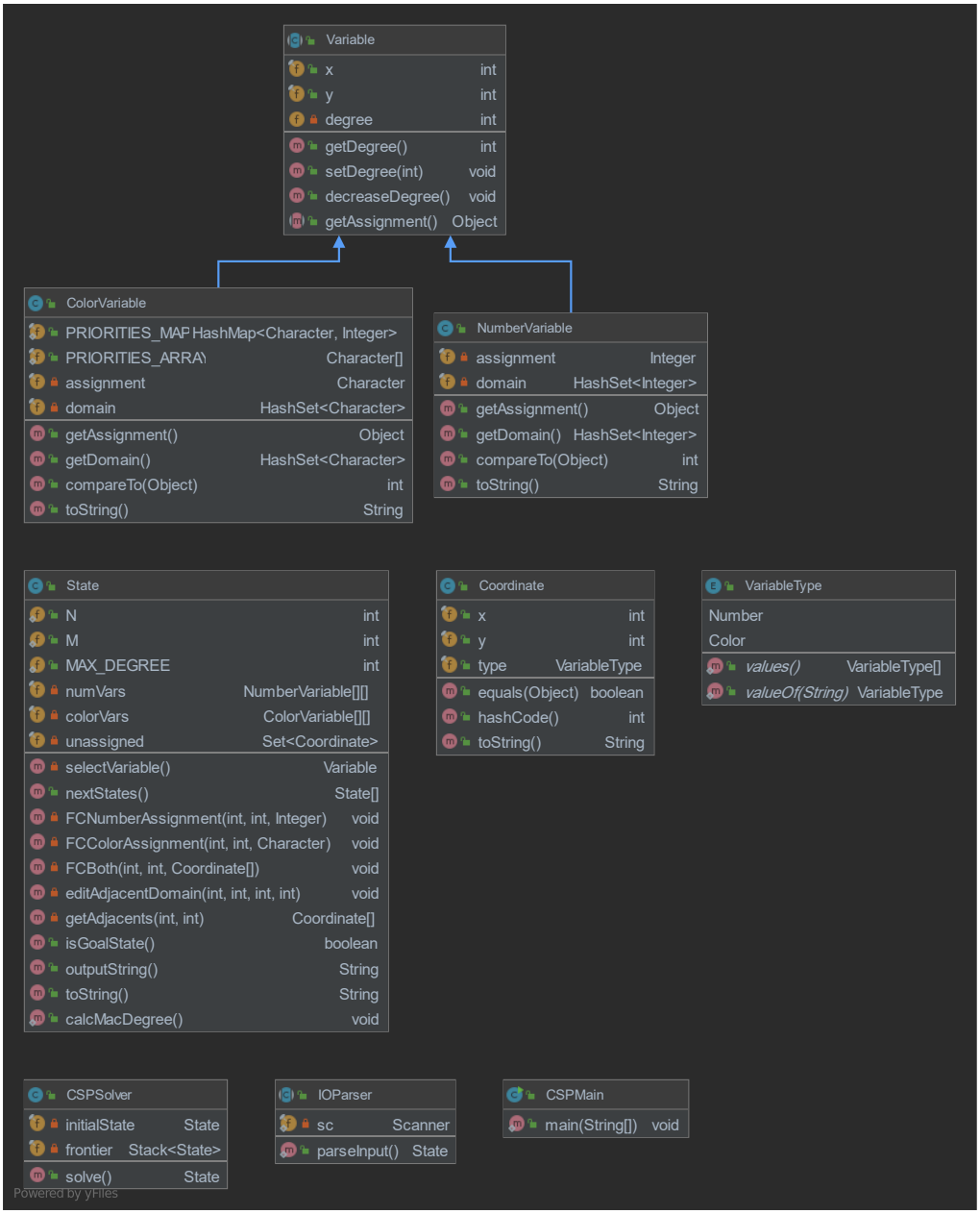
شرح کد مسئله:

شرح هر یک از کلاس‌ها برای حل مسئله در ادامه آمده است.

- کلاس Variable: یک کلاس انتزاعی (abstract) برای نگهداری اطلاعات کلی متغیرهای مسئله است که توسط کلاس‌های NumberVariable و ColorVariable توسعه می‌یابد.
- کلاس NumberVariable: این کلاس برای نگهداری اطلاعات مربوط به دامنه و مقدار احتمالی نسبت داده شده به متغیرهای عددی استفاده می‌شود.
- کلاس ColorVariable: این کلاس برای نگهداری اطلاعات مربوط به دامنه و مقدار احتمالی نسبت داده شده به متغیرهای رنگی استفاده می‌شود.
- کلاس State: برای ذخیره‌سازی هر حالت از مسئله استفاده می‌شود. شرح بخش‌های مختلف این کلاس در ادامه آمده است.
  - آرایه numVars: یک آرایه ۲ بعدی $n \times n$ از اشیای کلاس NumberVariable که متغیرهای عددی جدول در هر حالت (state) را نگهداری می‌کند.
  - آرایه colorVars: یک آرایه ۲ بعدی $n \times n$ از اشیای کلاس ColorVariable که متغیرهای عددی جدول در هر حالت (state) را نگهداری می‌کند.
  - مجموعه unassigned: متغیرهای بدون مقدار حالت (state) در این مجموعه نگهداری می‌شوند تا در هر مرحله از اجرای الگوریتم backtrack یک متغیر از این مجموعه توسط هیوریستیک‌ها انتخاب شود.
  - متد selectVariable: در این متد یک متغیر بدون مقدار توسط هیوریستیک MRV (و در صورت مساوی بودن توسط Degree) برای مقداردهی انتخاب می‌شود.
  - متد nextStates: این متد ابتدا یک متغیر بدون مقدار به کمک متد selectVariable() انتخاب می‌کند و سپس یک حالت (state) به ازای هر مقدار ممکن از دامنه متغیر انتخاب شده می‌سازد و برمی‌گرداند.
  - متدهای FCNumberAssignment، FCColorAssignment، FCBoth و editAdjacentDomain: پس از مقداردهی به هر متغیر بسته به نوع متغیر (عددی یا رنگی بودن آن) تابع FCNumberAssignment یا FCColorAssignment برای اجرای الگوریتم Forward Checking صدا زده می‌شود. در هر دوی این توابع پس از اعمال برخی اصلاحات دامنه‌ها که مخصوص به آن متغیر خاص است، تابع FCBoth صدا زده می‌شود. در این تابع نیز توسط تابع editAdjacentDomain مقادیر دامنه‌ی متغیرهای کناری در صورت لزوم اصلاح می‌شود.

- متد getAdjacents: مختصات خانه‌های مجاور هر خانه را به شکل آرایه‌ای از کلاس Coordinate برمی‌گرداند.

- کلاس Coordinate: این کلاس برای نگهداری مختصات و جنس هر متغیر استفاده می‌شود.
- کلاس CSPSolver: در این کلاس الگوریتم backtrack برای جست‌وجو در فضای مسئله و یافتن پاسخ آن پیاده‌سازی شده است.
- کلاس IOParser: این کلاس متدهایی ایستا (static) برای اعمال ورودی گرفتن از کاربر فراهم می‌کند.



شکل (۱) – نمودار UML کلاس‌های برنامه

مدل‌سازی مسئله:

۱. متغیرها: برای هر خانه‌ی جدول دو متغیر در نظر می‌گیریم. یک متغیر برای عدد خانه و یک متغیر برای رنگ خانه استفاده می‌شود. در نتیجه در مجموع $2 \times n^2$ متغیر برای مسئله داریم.
۲. دامنه‌ها: در ابتدا دامنه‌ی متغیرهای عددی اعداد 1 تا n و دامنه‌ی متغیرهای رنگی همه‌ی رنگ‌های ورودی برنامه است.
۳. محدودیت‌ها: بین هر دو متغیر عددی که در یک سطر یا ستون هستند یک محدودیت ۲تایی نامساوی بودن وجود دارد. هم‌چنین بین هر دو متغیر رنگی که مجاور هم هستند نیز محدودیت ۲تایی عدم تساوی وجود دارد. یک محدودیت ۴تایی نیز بین رنگ و عدد ۲ خانه‌ی مجاور وجود دارد. به این صورت که خانه‌ای که عدد بزرگ‌تری دارد، رنگ با اولویت بالاتری دارد و برعکس.

هیوریستیک‌ها:

در این مسئله برای انتخاب هر متغیر از هیوریستیک‌های MRV و Degree استفاده شده است. هیوریستیک MRV با شمارش تعداد اعضای دامنه‌ی هر متغیر به دست می‌آید. برای هیوریستیک Degree در ابتدای کار مقدار درجه هر متغیر (بیشینه مقدار ممکن درجه‌ی هر متغیر) را حساب می‌کنیم سپس در هر بار مقداردهی به یک متغیر، در صورت لزوم از مقدار درجه متغیرهایی که با آن یک محدودیت دارند کم می‌کنیم. بیشینه درجه‌ی هر متغیر از روابط زیر به دست می‌آید:

$$v \text{ is a number variable};\ Max_{Degree}(v) = 2 \times (n-1) + Adj.Count(v)$$
$$v \text{ is a color variable}\ \ ;\ Max_{Degree}(v) = 2 \times Adj.Count(v)$$

که در آن $Adj.Count(v)$ برابر تعداد همسایه‌های هر متغیر است.

اگر مقدار هیوریستیک MRV (تعداد اعضای دامنه هر متغیر) را با h_{MRV} و مقدار هیوریستیک درجه را با h_{Degree} نشان دهیم، برای انتخاب متغیر طبق هیوریستیک‌ها داریم:

$$\underset{(v \in V)}{arg\,max}.\, h(v) \;=\; \underset{(v \in V)}{arg\,max}.\, -h_{MRV}(v) + \frac{h_{Degree}(v)}{1 + MaxDegree}$$

که در آن $MaxDegree = 2(n-1) + 4 = 2n + 2$ بیشینه مقدار درجه‌ی ممکن برای همه‌ی متغیرها است.

از آنجا مقدار $h_{MRV}(v)$ همواره صحیح است و هم‌چنین همواره $0 \le \frac{h_{Degree}(v)}{1+MaxDegree} < 1$ است، در نتیجه اگر $-h_{MRV}(v) > -h_{MRV}(v')$ باشد، آن‌گاه صرف نظر از مقدار h_{Degree} برای هر دو متغیر خواهیم داشت:

$$h(v) > h(v')$$

و در صورت تساوی مقادیر دو متغیر برای هیوریستیک MRV ($-h_{MRV}(v) = -h_{MRV}(v')$) آنگاه مقدار h_{Degree} تعیین‌کننده خواهد بود.

نمونه اجرای برنامه:

ورودی نمونه ۱

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

خروجی نمونه ۱

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

ورودی نمونه ۲

```
4 5
a b c d
*# *# *# *# *#
*# 4c *# *# *#
*# *# *# *# *#
*# *# *# *# *#
*# *# *# *# *#
```

خروجی نمونه ۲

```
5c 1d 4c 2d 3c
2d 4c 3d 5c 1d
3c 2d 5c 1d 4c
4b 5a 1d 3c 2d
1d 3b 2c 4b 5a
```