در بخش اول این پروژه (فولدر CSP) میخواهیم همان مسئله مثال درس، یعنی رنگ‌آمیزی نقشه را حل کنیم. در این بخش سعی داریم با مدل کردن این مسئله به شکل یک CSP و استفاده از الگوریتم‌هایی که برای حل و بهبود عملکرد حل یاد گرفتیم، هر نقشه‌ای را بتوانیم بدون این که زمان زیادی ببرد، رنگ‌آمیزی کنیم. دقت کنید قبل از شروع کار پکیج‌های مورد نیاز پروژه که در فایل requirements.txt قرار دارند نصب کرده باشید.[1] میدانیم طبق قضیه چهار رنگ[2] فقط با استفاده از ۴ رنگ میتوانیم هر نقشه را طوری رنگ‌آمیزی کنیم که هیچ ۲ ناحیه مجاوری رنگ تکراری نداشته باشند.

با دو روش میخواهیم این CSP را حل کنیم: ۱- جستجوی عقبگرد (بک‌ترکینگ سرچ) ۲- بهبود تکرار شونده

به یاد بیاورید که دو روش عمومی برای بهبود بک‌ترکینگ یاد گرفتیم: فیلترینگ و اوردرینگ. در فیلترینگ با بررسی دامنه‌های متغیرها قبل از پیش بردن رنگ‌آمیزی، مقادیری که میتوانستیم متوجه بشویم قرار نیست در یک جواب برای مسئله باشند (در صورت مقداردهی شدن مطمئناً منجر به بک‌ترک خواهند شد) از دامنه حذف میشوند. در ابتدا بدون هیچ فیلترینگی اقدام به رنگ‌آمیزی نقشه‌ها خواهید کرد. در این روش تا ناسازگاری رخ نداده، بک‌ترکی رخ نخواهد داد؛ به عبارتی هرگاه دو ناحیه مجاور همرنگ مشاهده شد باید بعد از آن شاهد بک‌ترک باشید. ساده‌ترین نوع فیلترینگ forward checking، و بعد از آن arc consistency بود که هر دو را پیاده‌سازی خواهید کرد، با استفاده از فیلترینگ باید کاهش بسیار واضحی در تعداد بک‌ترکینگ‌ها مشاهده کنید. روش دوم برای بهبود الگوریتم ordering بود. در این روش با کمک هیوریستیک‌هایی که یاد گرفتید در در انتخاب بین متغیرهای باقیمانده و رنگ‌های فیلتر نشده در دامنه متغیرها هوشمندانه عمل خواهید کرد تا مجدداً سعی بر کاهش مقدار بک‌ترک داشته باشیم و در نتیجه مسئله سریعتر حل شود.

سه قطعه map.py، solver.py، و utils.py را مشاهده میکنید. کد map.py مربوط به پیش‌پردازش و پردازش‌های تصویری لازم روی تصاویر نقشه‌هاست و نیازی به تغییر دادن آن نیست. کد solver را با آرگومان نام نقشه‌ای که میخواهیم رنگ شود برای اجرای الگوریتم اجرا میکنیم (به همراه فلگ‌های مربوط به مد اجرا که در ادامه توضیح داده شده است) و تنها دو متود solve به دو روش اشاره شده، که توسط کامنت‌ها نیز مشخص شده‌اند را در آن کامل میکنید، ولی با خواندن میتوانید درک بهتری از متغیرهای ورودی متودهایی که قرار است بنویسید پیدا کنید. در نهایت تمامی الگوریتم‌های مورد استفاده در متودهای solve در utils.py مشخص شده‌اند. کامنت‌ها را به دقت بخوانید تا بتوانید بهتر و مطابق آنچه انتظار میرود به خوبی متودها را پیاده کنید. قسمت‌هایی که باید پر کنید در کد به وضوح مشخص شده‌اند. در نهایت با سه نقشه‌ی آماده شده ایران قدیم، استان تهران و کشور آمریکا که در فولدر قرار دارند کد خود را آزمایش کنید. اجرای کد به این شکل میباشد:

فلگ اول: بدون فیلترینگ n، با فوروارد چکینگ fc، و با آرک کانسیستنسی ac

فلگ دوم: استفاده از اوردرینگ برای انتخاب متغیر t و عدم استفاده f

فلگ سوم: استفاده از اوردرینگ برای انتخاب مقدار t و عدم استفاده f

برای مثال در ابتدایی ترین مد دستور به این شکل:

```
python solver.py iran.jpg -n -f -f
```

و در پیشرفته‌ترین مد دستور به این شکل در میاید:

```
python solver.py usa.png -ac -t -t
```

[1] pip install -r requirements.txt
[2] https://en.wikipedia.org/wiki/Four_color_theorem

برای دیباگ کردن میتوانید مقدار SLEEP_TIME_IN_MILLISECONDS در خط هشتم solver.py را زیادتر کنید تا مراحل رنگ شدن و بک‌ترکینگ را بهتر مشاهده کنید. (و اگر پرینتی در کنسول اضافه کردید بتوانید شمرده‌تر آن را با خروجی گرافیکی تطبیق دهید.) دقت کنید با Exc میتوانید اجرا را متوقف کنید.

بسته به میزان استفاده از ابزارهای بهبود الگوریتم تعداد بک‌ترک نیز عوض میشود. انتظار میرود با استفاده از هر دو ابزار بتوانید با تعداد بک‌ترک صفر نقشه آمریکا را رنگ کنید.

راهنمایی) تعداد بک‌ترک‌ها بر اساس مد اجرا مطابق زیر خواهند بود:

ابتدایی ترین مد: ایران ۱۱، استان تهران ۳۹، و آمریکا ۶۵

با forward-checking ولی بدون اوردرینگ: ایران ۰، استان تهران ۲، آمریکا ۱

با هر دو اوردرینگ ولی بدون فیلترینگ: ایران ۱۴، استان تهران ۲۴، آمریکا ...

با هر نوع فیلترینگ و هر دو اوردرینگ: ایران ۰، استان تهران ۰، و آمریکا ۰

میتوانید از راهنمایی نیز حدس بزنید که بعضاً تعداد بک‌ترک‌ها مطابق انتظار نخواهند بود. سعی کنید توجیحی برای این اتفاق بیابید.

امتیازی: سعی کنید راه حلی بیابید که در آن با ایجاد تغییری در نحوه اوردرینگ، حتی در صورت عدم وجود فیلترینگ، اوردرینگ عملکرد را بهتر کند. (سعی کنید با اوردرینگ و بدون فیلتریگ آمریکا را رنگ‌آمیزی کنید...)

روش دوم حل CSP با کمک بهبود تکرار شونده است. در این روش مقداردهی اولیه رندومی به همه متغیرها انجام میدهیم و به صورت رندوم مقادیری که منجر به نقض محدودیت شده‌اند با مقدار دیگری جایگزین میکنیم. از هیورسیتیکی که در درس برای انتخاب مقدار جدید یاد گرفتید استفاده کنید. این روش ضمانتی به رسیدن به جواب ندارد و وابسته به مقداردهی اولیه ممکن است بسیار سریع به جواب برسد یا هیچوقت به جواب نرسید و در یک حلقه بی‌انتها گیر کند، ولی اکثر اوقات برای نقشه‌هایی که در این پروژه رنگ میکنیم با سرعت خوبی به جواب میرسد. برای حل مسئله به این روش دستور را به این شکل اجرا کنید:

```
python solver.py usa.png -ii
```