

به نام خدا

نوید نصیری ۹۸۲۹۵۴۳

گزارش پروژه نهایی درس هوش مصنوعی

استاد : دکتر پالهنک

موضوع پروژه : رنگ آمیزی گراف

شرح پروژه : مسئله رنگ آمیزی گراف به این صورت است که سعی کنیم با کمترین رنگ، تمام گره‌های گراف را رنگ آمیزی کنیم به طوری که هیچ دو گره‌ای در کنار هم، هم‌رنگ نباشند.

برای حل این سوال، یک برنامه پایتون نوشتیم که به کمک الگوریتم جست‌وجوی محلی و همچنین قابلیت سرد شدن تدریجی، بتواند به روش‌های مختلفی مسئله را حل کند که در پایین به بخش‌های مختلف آن اشاره خواهیم کرد.

ورودی برنامه یک فایل تکست است که هنگام اجرای برنامه پایتون، آدرس آن را به آن می‌دهیم و فرمت آن به شکل زیر است:

```
GraphColoring > input_coloring.txt
1 11 20
2 e 1 2
3 e 1 4
4 e 1 7
5 e 1 9
6 e 2 3
7 e 2 6
8 e 2 8
9 e 3 5
10 e 3 7
11 e 3 10
12 e 4 5
13 e 4 6
14 e 4 10
15 e 5 8
16 e 5 9
17 e 6 11
18 e 7 11
19 e 8 11
20 e 9 11
21 e 10 11
22
```

در خط اول، به ترتیب اول تعداد گره‌ها و سپس تعداد یال‌های گراف را وارد می‌کنیم و سپس در خط‌های بعدی، پس از نوشتن حرف e که اول کلمه edge است، شماره دو گره‌ای که یالی بین آن دو وجود دارد را می‌نویسیم.

خروجی برنامه هم یک فایل تکست است که آدرس تشکیل شدن آن را در اجرای برنامه به آن می‌دهیم و به فرمت زیر می‌باشد:

```
graphColoring > output_color
```

```
1 4
2 1 2
3 2 4
4 3 2
5 4 1
6 5 3
7 6 3
8 7 1
9 8 1
10 9 1
11 10 3
12 11 2
13
```

که در خط اول، حداکثر رنگ مورد استفاده در این مسئله را بیان می‌کند و در خط های بعدی، عدد اول نشان‌دهنده شماره گره و عدد دوم نشان‌دهنده شماره رنگ آن گره است به طوری که هیچ دو گره همسایه‌ای هم‌رنگ نشوند.

سپس برای اجرای برنامه می‌توانیم از h - استفاده کنیم تا قابلیت‌های آن را مشاهده

کنیم:

```

(env_graph) /media/navid/New Volume/Term_7/AI/ai_project/GraphColoring — navid
ix:pts/1 → python3 main.py -h
(21:33:00) → python3 main.py -h
usage: GraphColoring [-h] [-i INPUTFILE] [-o OUTPUTFILE] [-d] [-s]

in this program, you can color the graph with graph-coloring algorithm

optional arguments:
  -h, --help            show this help message and exit
  -i INPUTFILE, --inputfile INPUTFILE
                        write path of input file
  -o OUTPUTFILE, --outputfile OUTPUTFILE
                        write path of output file
  -d, --debug            if you want to show level by level coloring, set this flag
  -s, --simulated_annealing
                        if you want to run algorithm with simulated annealing, set this flag

Enjoy coloring...:)
(env_graph) /media/navid/New Volume/Term_7/AI/ai_project/GraphColoring — navid
ix:pts/1 →
(21:41:17) →

```

همانطور که مشاهده می‌کنیم:

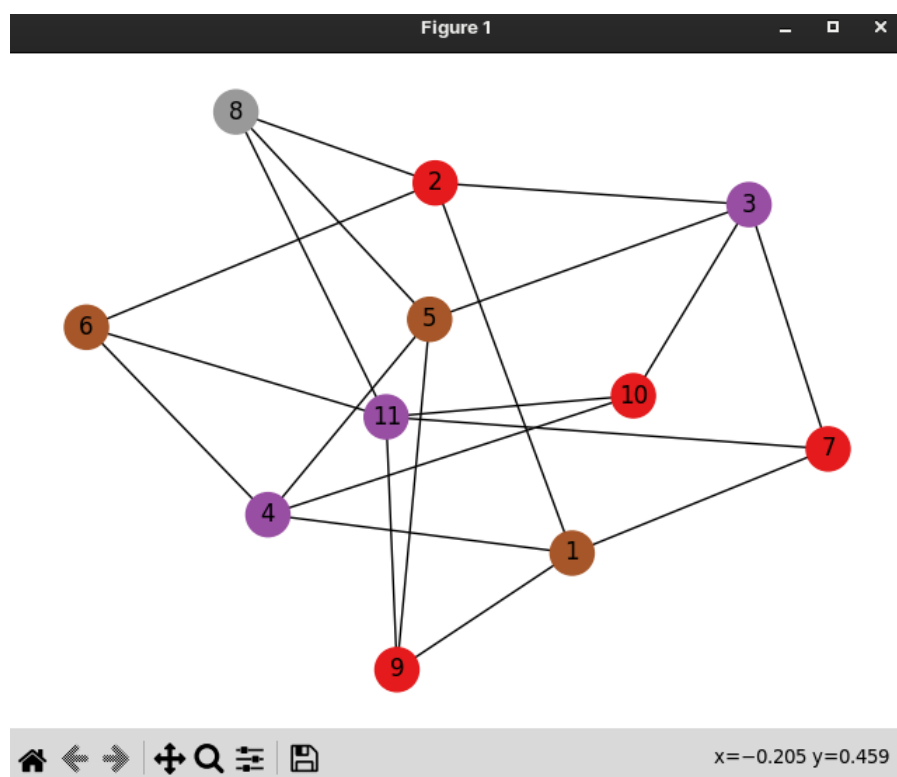
- با استفاده از -i می‌توانیم آدرس فایل تست ورودی را بدهیم.
- با استفاده از -o می‌توانیم آدرس فایل خروجی را بدهیم.
- با استفاده از -d می‌توانیم برنامه را در حالت دیباگ، اجرا کنیم. به این معنی که رنگامیزی گراف را مرحله به مرحله به ما نشان می‌دهد.
- با استفاده از -s هم می‌توانیم از روش بهینه‌ساز سرد شدن تدریجی استفاده کنیم.

توضیح الگوریتم پیاده شده :

الگوریتم پیاده شده‌ی جست‌وجوی محلی به این صورت است که در ابتدا به همه گره‌ها یک رنگ اختصاص می‌دهد. سپس تمام همسایه‌های آن حالت از رنگامیزی را

محاسبه می‌کند و آن همسایه‌ای را برای حالت بعدی انتخاب می‌کند که مقدار evaluation کمتری را داشته باشد.

مقدار evaluation برای هر حالت به این صورت محاسبه می‌شود که تعداد جفت هم‌رنگ‌هایی که کنار هم هستند را می‌شمارد. پس هر بار آن همسایه‌ای که بیشتر از بقیه، جفت هم‌رنگ‌های کنار هم را کاهش داد، انتخاب می‌شود.



در حالتی هم که از سرد شدن تدریجی استفاده می‌کنیم، به این شکل است که دیگر همواره بهترین همسایه را انتخاب نمی‌کنیم و با یک احتمالی، اجازه می‌دهیم یک انتخاب شانسی از بین همسایه‌ها داشته باشیم و ما با این قابلیت می‌توانیم پاسخ‌های متنوع‌ای برای مسئله پیدا کنیم که چند نمونه از آنها را می‌توانیم در پایین مشاهده کنیم:

