رنگ آمیزی نقشه

مبینا نوری – ۹۶۱۵۰۹۱۰۶۳

ابتدا یک کلاس به نام Graph ایجاد کردیم.
و با استفاده از تابع add_ege یال ها را یکی یکی به آن اضافه کردیم :

```
43   graph = Graph(10)
44   graph.add_edge(1, 2)
45   graph.add_edge(1, 3)
46   graph.add_edge(1, 4)
47   graph.add_edge(1, 5)
48   graph.add_edge(1, 6)
49   graph.add_edge(2, 9)
50   graph.add_edge(2, 10)
51   graph.add_edge(2, 6)
52   graph.add_edge(3, 4)
53   graph.add_edge(4, 5)
54   graph.add_edge(5, 6)
55   graph.add_edge(6, 7)
56   graph.add_edge(6, 10)
57   graph.add_edge(7, 8)
58   graph.add_edge(7, 10)
59   graph.add_edge(8, 9)
60   graph.add_edge(8, 10)
61   graph.add_edge(9, 10)
```

برای بکترک زدن روی حالت رنگ آمیزی گراف ابتدا رنگ هر راس را برابر با -۱ مقدار دهی میکنیم.

سپس تابع بازگشتی backtrack_colors را صدا میزنیم.

```
37      def color(self):
38          colors = dict()
39          for i in range(1, self.n + 1):
40              colors[i] = -1
41          return (self.backtrack_colors(1, colors), colors)
```

این تابع دو ورودی میگیرد.در خروجی اگر بتواند کل راس ها را رنگ آمیزی معتبر کند True وگرنه False برمیگرداند.
اول شماره راسی که میخواهیم رنگ آمیزی کنیم و دوم وضعیت فعلی رنگ ها. اگر شماره راس از n بیشتر باشد یعنی کل گراف رنگ شده و True برگردانده میشود.

در ادامه با ترتیبی رندوم برای ۴ رنگ موجود (برای اینکه هربار یک نتیجه را ندهد) شروع به چک کردن میکنیم.
شرط این که بتوان یک رنگ را برای راس جدید گذاشت این است که هیچ کدام از همسایه هایش به آن رنگ نباشد. بعد از اینکه رنگ این راس را ست کردیم همین تابع را برای راس های بعدی صدا میکنیم. اگر True برگردانده شد True برمیگردانیم وگرنه رنگ های بعدی را برای این راس تست میکنیم.
اگر به ازای هر رنگی برای این راس، رنگ آمیزی معتبر پیدا نشد False برگردانده میشود.

```python
    def backtrack_colors(self, v, colors):
        if v > self.n:
            return True

        color_candidates = list(range(4))
        random.shuffle(color_candidates)

        for i in color_candidates:
            ok = True
            for adj in self.edges[v]:
                if colors[adj] == i:
                    ok = False
                    break
            if ok:
                colors[v] = i
                if self.backtrack_colors(v + 1, colors):
                    return True

        colors[v] = -1
        return False

```

نمونه اجرای برنامه :

```
Coloring:
1 : 3
2 : 1
3 : 2
4 : 1
5 : 2
6 : 4
7 : 2
8 : 4
9 : 2
10 : 3
```