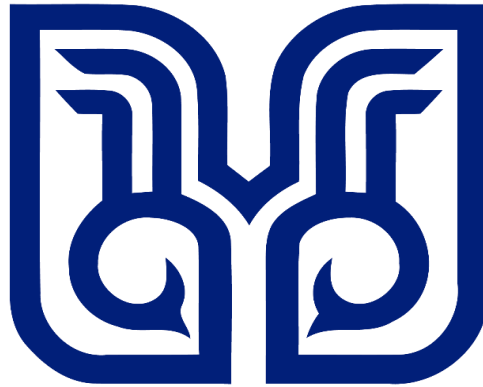


بِناَمِ خُداوندِ بخشنده مهربان



دانشگاه شهید باهنر کرمان

دانشکده فنی و مهندسی
بخش مهندسی کامپیوتر

مقایسه الگوریتم genetic و min-conflicts در حل مسئله N-Queen

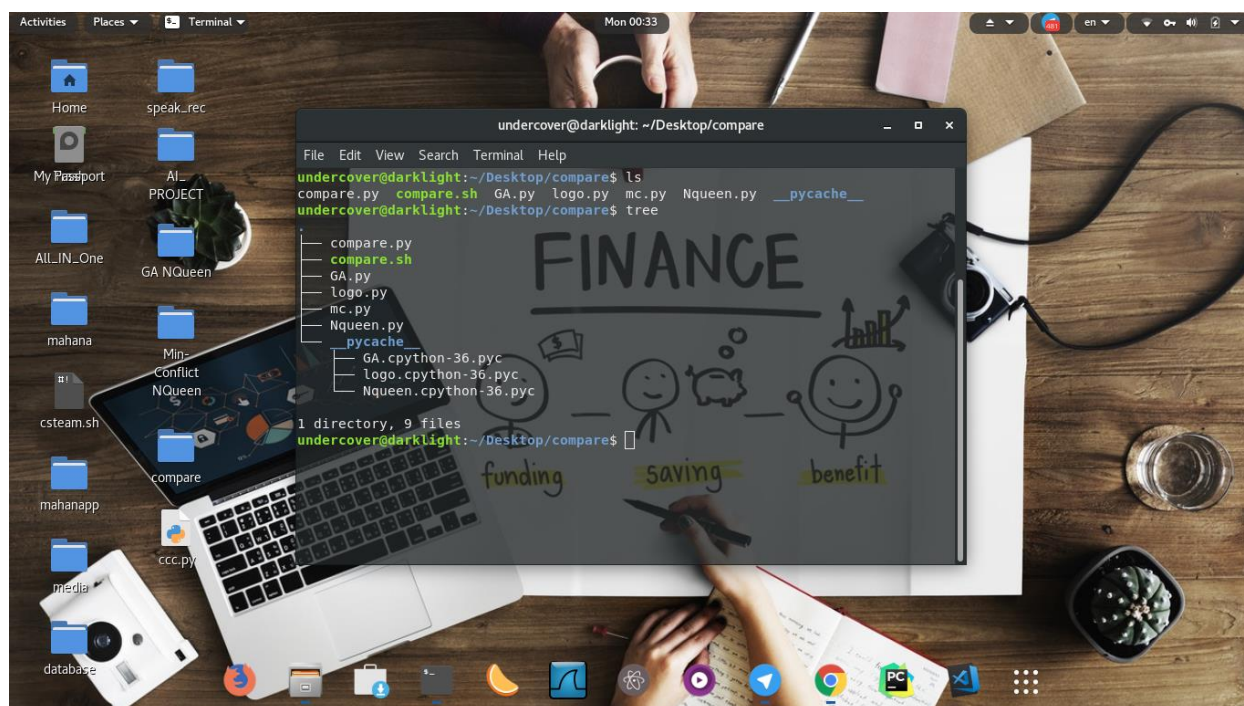
درس: هوش مصنوعی

استاد مربوطه: دکتر مهدی افتخاری

نویسنده: سعید قاسم شیرازی

94403036

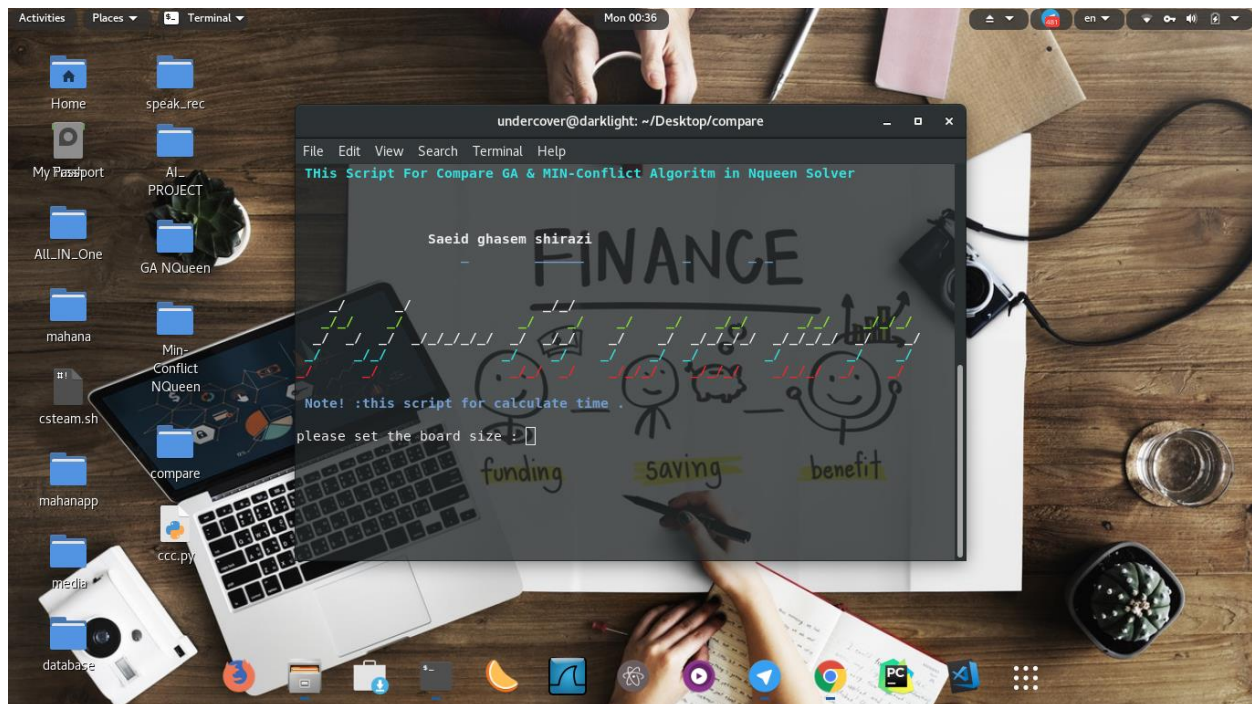
ساختار کلی پروژه به صورت زیر است:



که شامل یک فایل بش اسکریپت برای مقایسه 2 الگوریتم است و اسکریپت های دو الگوریتم به زبان پایتون نوشته شده اند.



الگوریتم اولی که اجرا میشود الگوریتم ژنتیک است. این الگوریتم به این صورت عمل میکند که در ابتدا سایز صفحه را از کاربر میگیرد:



و مقدار $population\ size$ به صورت پیش فرض 8 قرار دادیم و $generation\ size$ را برابر منفی 1 قرار داده ایم به این دلیل که الگوریتم تا جایی که جواب را پیدا کند ادامه یابد.

الگوریتم ما دارای توابع `switch`, `mutation`, `compute fitness`, `goal find`, `random selection`, `first` & `next generation` می باشد.

که توابع فوق کامنت گذاری شده و میتوان نحوه کار تابع را به صورت کامل فهمید.
به عنوان مثال:

```
def goal_find(self):
    for population in self.population:
        if population.fitness == self.goal:
            return True
    return False
def random_selection(self):
    #entekhab chand item az population feli baraye next generation
    #k in selection ha bishtarin meghtar fitness ra daranad!
    population_list = []
    for i in range(len(self.population)):
        population_list.append((i, self.population[i].fitness))
    population_list.sort(key=lambda pop_item: pop_item[1], reverse=True)
    return population_list[:int(len(population_list) / 3)]
```

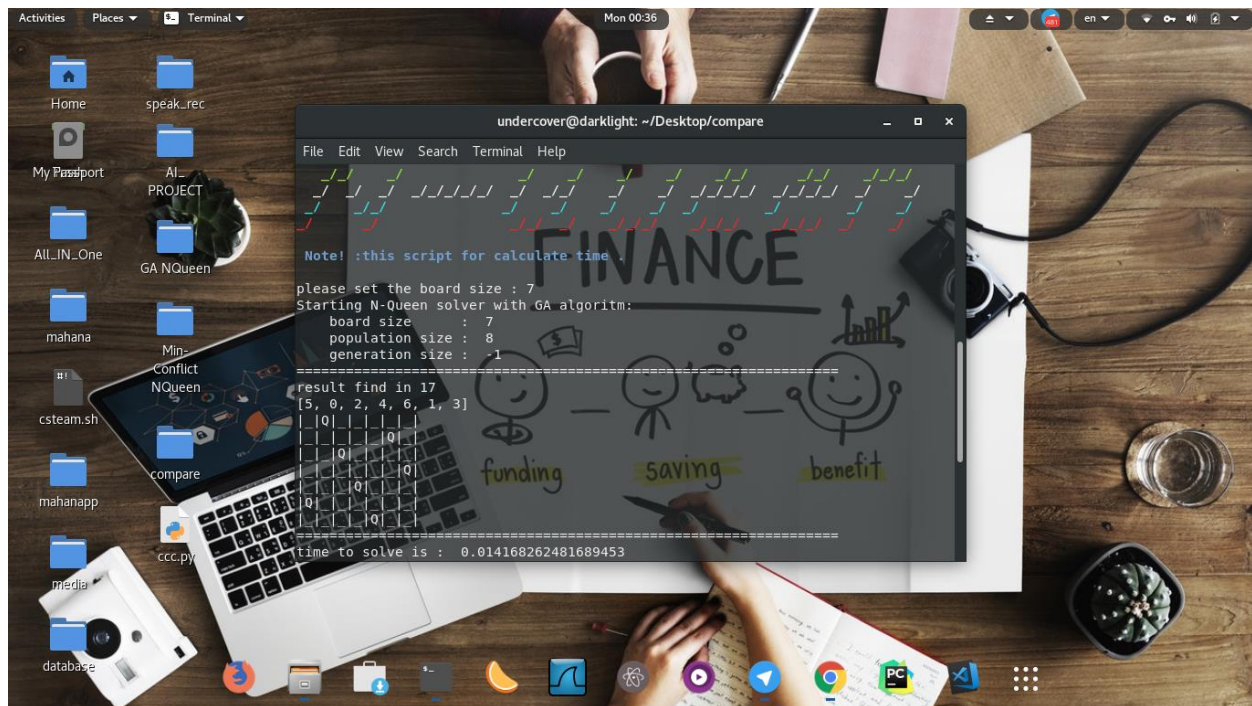

احتمال mutation در کد ما 0.25 در نظر گرفته شده است.

برای فهم کلی الگوریتم از سایت زیر استفاده شده است:

https://kushalvyas.github.io/gen_8Q.html

که مفهوم توابع ... , fitness, mutation به صورت کامل توضیح داده شده است که برای جلوگیری از حجم مطلب از بیان مجدد آن صرفه نظر کرده و کد کامنت گذاری شده پروژه پیوست گردیده است.

بعد از الگوریتم ژنتیک که به صورت کامل پیاده سازی شد و نتیجه خروجی آن به شکل زیر می باشد :



```
undercover@darklight: ~/Desktop/compare
File Edit View Search Terminal Help

Note! :this script for calculate time .

please set the board size : 7
Starting N-Queen solver with GA algorithm:
board size      : 7
population size : 8
generation size : 1

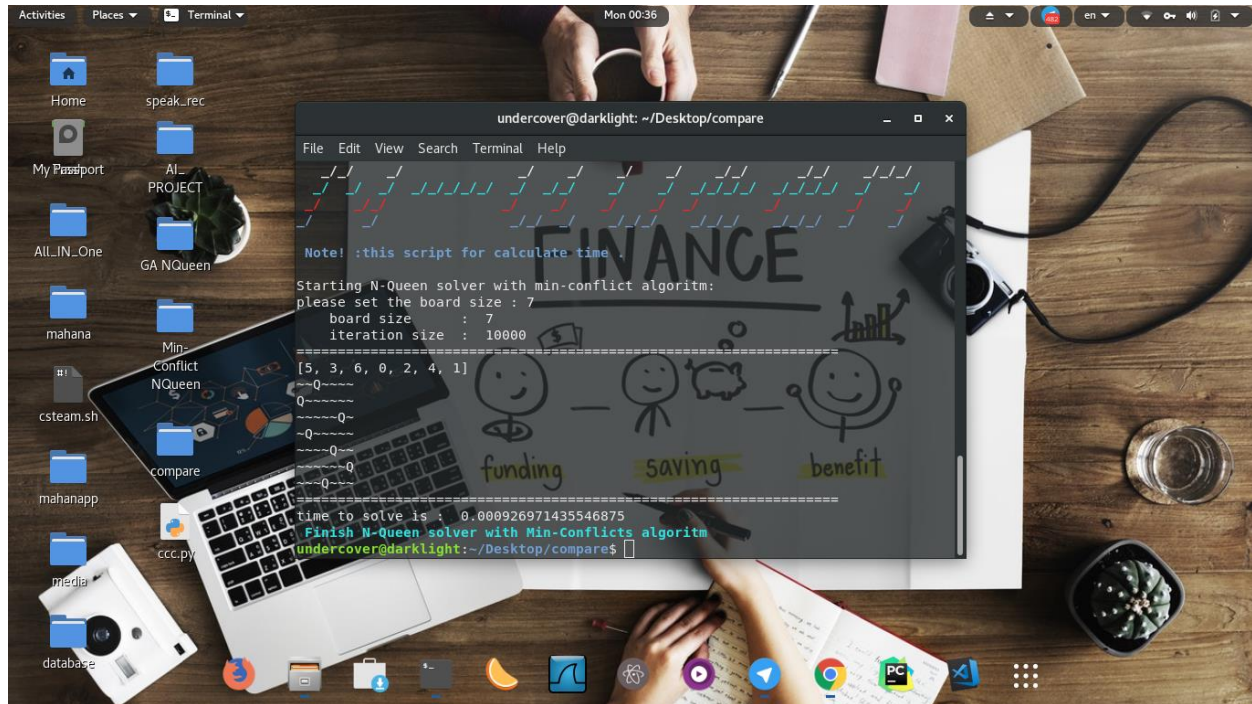
=====
result find in 17
[5, 0, 2, 4, 6, 1, 3]
|0| | | | | |
|0| | | | | |
|0| | | | | |
|0| | | | | |
|0| | | | | |
|0| | | | | |
|0| | | | | |
=====
time to solve is : 0.014168262481689453
```

نوبت به الگوریتم بعدی که min-conflicts است میرسیم.

الگوریتم ما شامل توابع hits, find conflict, min-conflicts, show می باشد.

در ابتدا این الگوریتم سائز بورد را از کاربر میگیرد، در این الگوریتم تعداد iteration که ما در نظر گرفته ایم 10000 می باشد یعنی تا 10000 بار میاد و چک میکنه که آیا جمع conflict های تمامی وزیر ها برابر 0 شده است یا نه. که اگر در این تعداد خطا 0 نشد پیغام خطا چاپ میکند. که این الگوریتم به صورت کلی میتوان گفت اینگونه عمل میکند که وزیر ها را رو قطر میچیند و به صورت دندوم یکی را انتخاب میکند و انقدر انرا تغییر میدهد تا تعداد conflict های آن به 0 برسد و سپس یک وزیر دیگر را به صورت رندوم انتخاب کرده و همین عملیات را تکرار میکند تا جایی که جمع تمامی برخوردها 0 شود. به صورت خلاصه میتوان گفت که این الگوریتم عملکردی شبیه به انسان را برای حل این مسئله انجام میدهد و طبق

اسکرپت مقایسه ای که ما نوشته ایم که زمان اجرای هر دو الگوریتم را به صورت دقیق محاسبه میکند میتوان فهمید که در این مثال الگوریتم min-conflicts خیلی سریع تر به جواب میرسد و عملکرد بهتری دارد.



تمامی کد های نوشته شده 2 الگوریتم به صورت کامنت گذاری شده پیوست گردیده است. و برای مقایسه 2 الگوریتم کافیت اسکرپت بش را اجرا کنیم.