تمرین درس شبیه سازی رایانه ای در فیزیک

حل مساله ی فروشنده ی دوره گرد با الگوریتم ژنتیکی

علیرضا رضایی

97100762

استاد اجتهادی

حل مساله ی فروشنده ی دوره گرد با الگوریتم ژنتیکی

مقدمه

در این تمرین میخواهیم مساله ی فروشنده ی دوره گرد را با الگوریتم ژنتیکی شبیه سازی کنیم به این صورت که تعدادی شهر داریم و میخواهیم ببینیم اگر فرشنده ای دوره گرد بخواهد از شهری شروع کند و بعد از گذر از همه ی شهر ها به آخرین شهر برسد ، به ترتیب از چه شهر هایی عبور کند که کمترین مسیر را در نهایت طی کرده باشد.

در این حل شهر ها را در راس های یک شبکه ی مربعی دو بعدی درنظر میگیریم و اقدام به حل سوال می کنیم.

در ابتدای کد کتابخانه های مورد نیاز را وارد کرده و سپس ثوابتی که در شبیه سازی با آنها کار خواهیم کرد را معرفی می کنیم.

number_of_total_ways = 100

تعداد کل راه هایی که میخواهیم به عنوان ژن های اولیه انتخاب کنیم.

sp = 1.3

P = [(1/n)*(sp-2*(sp-1)*((i-1)/(n-1))) for i in range(1, n+1)]

احتمال انتخاب هر راه بر حسب مکانش در دنباله ی مرتب شده ی راه ها

mutation_probability = 0.08

احتمال جهش

در قسمت بعدی توابع کمکی را تعریف می کنیم که عبارتند از:

1- تابع فاصله ی بین دو شهر که مختصات دو شهر را می گیرد و فاصله ی بینشان را حساب می کنید. (|B-A|)
2- تابع مجموع فاصله های بین شهر ها در یک مسیر مشخص که دنباله ای از شهر ها را می گیرد و با حساب فاصله ی بین هر دو شهر پی در پی و جمع کردن کل این فاصله ها طول مسیر پیموده شده از اولین تا اخرین شهر را حساب می کند.
3- تابع مرتب (سورت) کردن راه ها بر حسب مجموع طول راه که مجموعه ای از راه ها را گرفته و بر حسب کوتاه ترین به بلند ترین آنها را در دنباله مرتب می کند.
4- انتخاب تصادفی دو راه از بین کل راه ها با تابع توزیع احتمال مشخص شده که دو راه را بر حسب مکانشان در دنباله ی راه ها (که بر حسب کوتاهی مسیر مرتب شده است) و احتمالی که به این مکان ها می دهیم انتخاب می کند.
5- تابع انجام جهش که با یک احتمالی جای دو شهر را به صورت تصادفی در دنباله ی راه مشخص شده عوض می کند.
6- تابع ساخت فرزند از روی والد که از روی یک والد یک فرزند برایش به این صورت می سازد که :

```
for i in range(len(way)):
    w[i] = way[len(way)//2-i]
```

7- تابع رسم نتایج هم که وظیفه ی رسم نتایج و نمودار ها را بر عهده دارد.

در قسمت بعدی به حلقه ی اصلی حل با الگوریتم ژنتیکی می رسیم که تعداد شهر ها را به صورت اجباری و تعداد تکرار حلقه و جواب دقیق را به صورت اختیاری در ورودی می گیرد و اگر تعداد تکرار حلقه وارد شده بود به آن میزان الگوریتم ژنتیکی را تکرار کرده و جواب ها را نمایش می دهد و اگر جواب دقیق وارد شده بود آنقدر الگوریتم را تکرار می کند که به جواب دقیق برسد.

این تابع به این صورت کار می کند که ابتدا مجموعه ای به طول number_of_total_ways از راه های مختلف می سازد

سپس در حلقه ی اصلی هر بار این مجموعه را بر حسب کوتاه ترین راه مرتب کرده سپس دو تا از کوتاه ترین راه ها را نگه داشته و از بقیه ی راه ها به صورت تصادفی فرزند تولید می کند و انها را به صورت تصادفی جهش می دهد و سپس دوباره مجموعه ی همه ی راه ها را مرتب می کند و این روند را هی تکرار می کند تا به نتایج مطلوب گفته شده در ابتدای این توضیح برسیم.

در قسمت حل دقیق هم مجموعه ی همه ی راه های ممکن را تولید کرده و طول همه ی راه ها را حساب می کنیم و کوتاه ترین راه را به عنوان جواب دقیق خروجی می دهیم.

توضیحات جزیی تر هم به صورت کامنت گذاری در کد مشخص شده است.

نتایج

main(n=100, max_iterations=100)

shortest path has length: 433.16394619052915

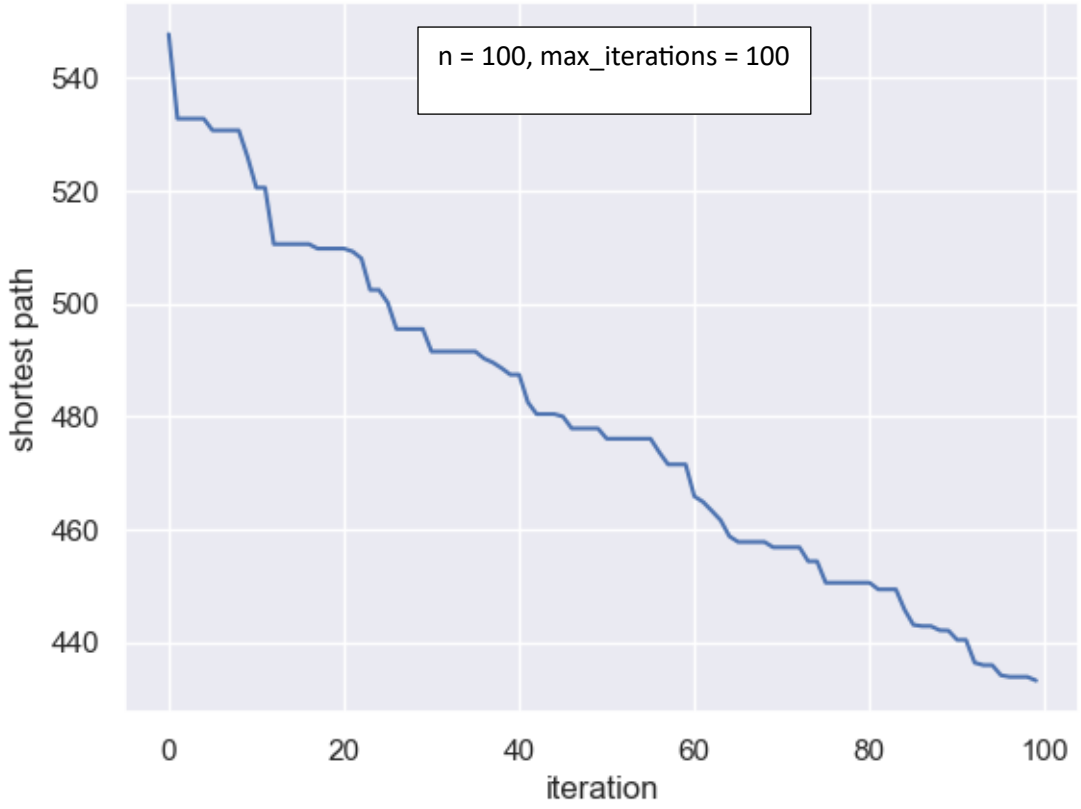

fig 1: shortest path Vs iteration step

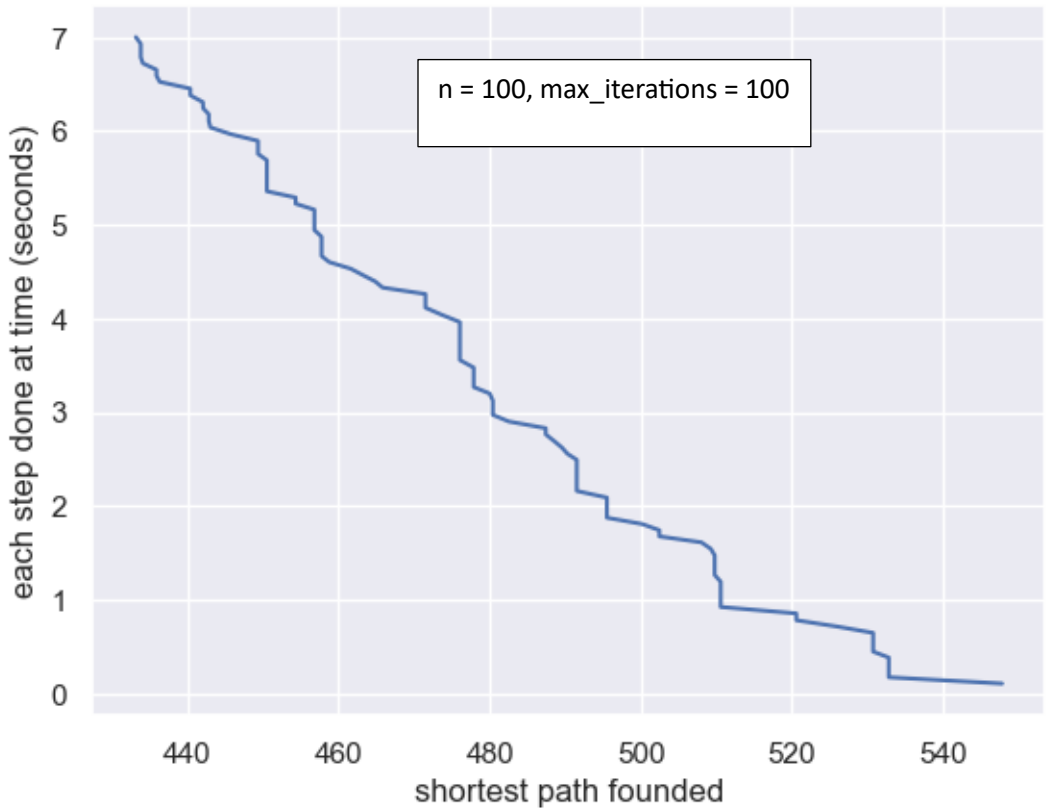

fig 2: each step done at time Vs shortest path founded

exact_main(9)

calculated in 21.616816997528076 seconds.
جواب دقيق: 8.0

main(n=9, exact_answer=8)

shortest path has length: 8.0

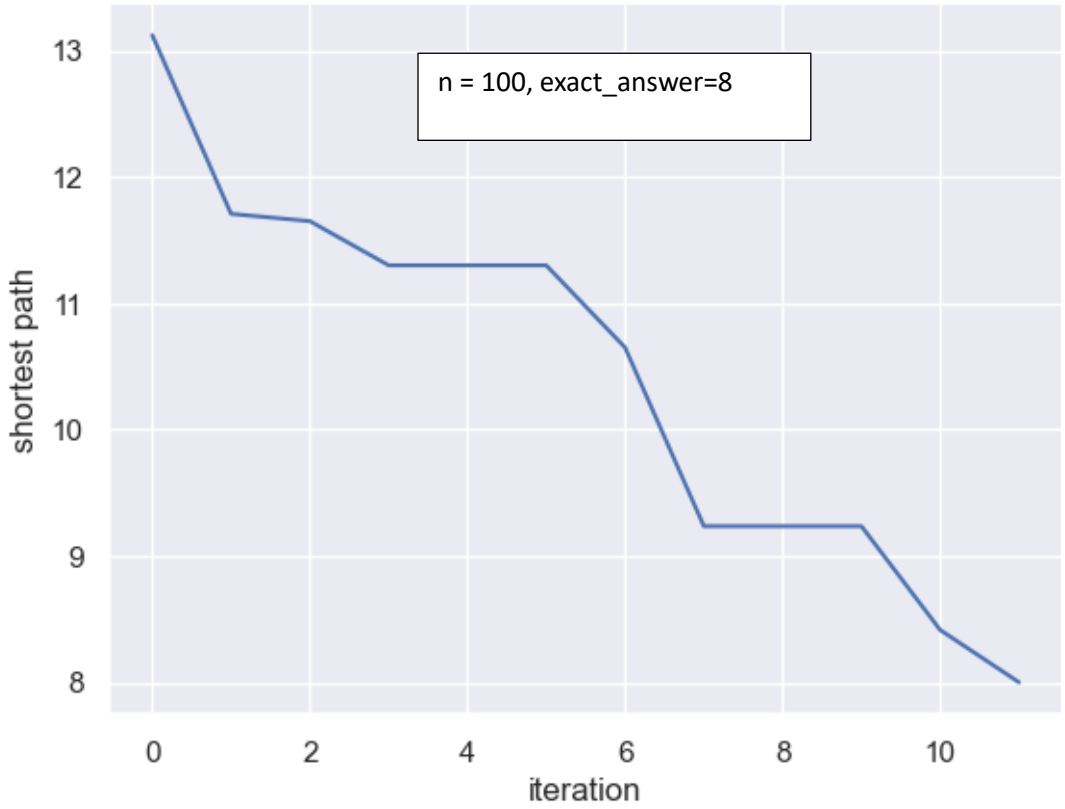

fig 1: shortest path Vs iteration step

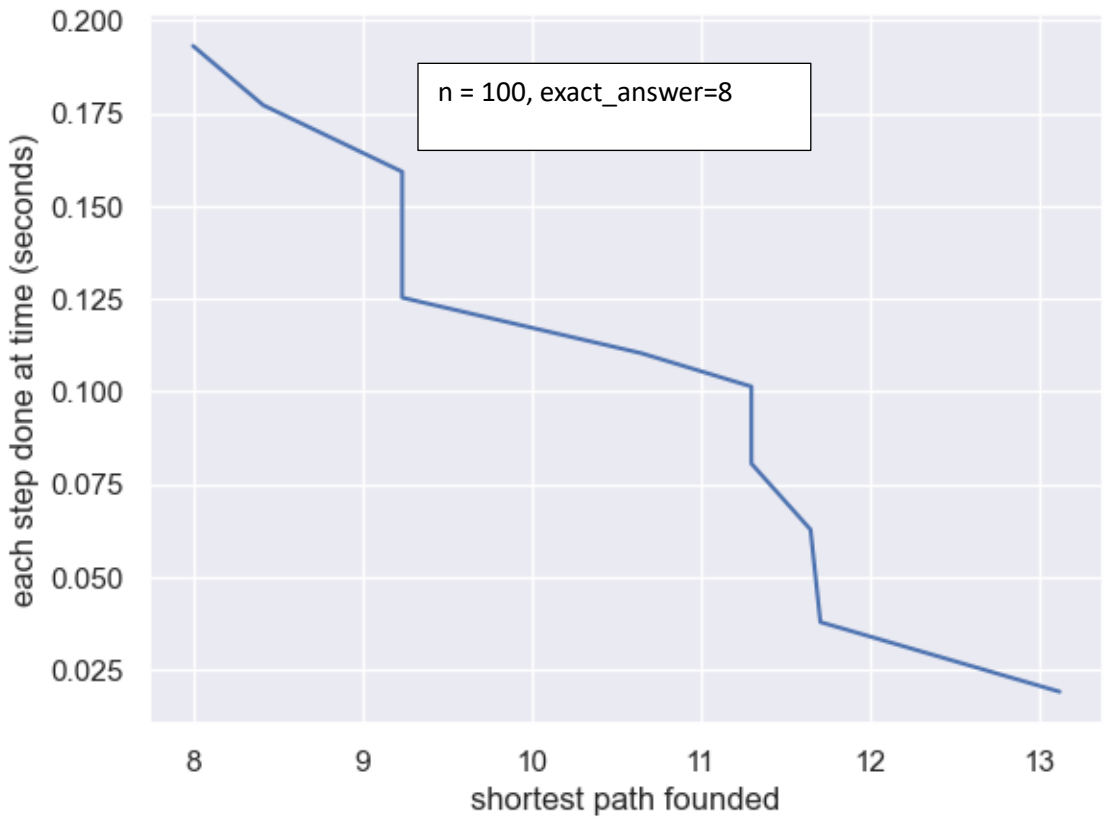

fig 2: each step done at time Vs shortest path founded