به نام خدا

تمرین سری ششم

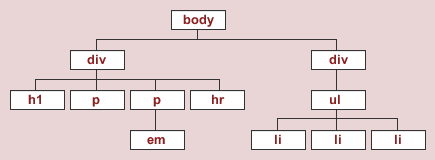
درس مبانی هوش محاسباتی

سینا علی‌نژاد

99521469

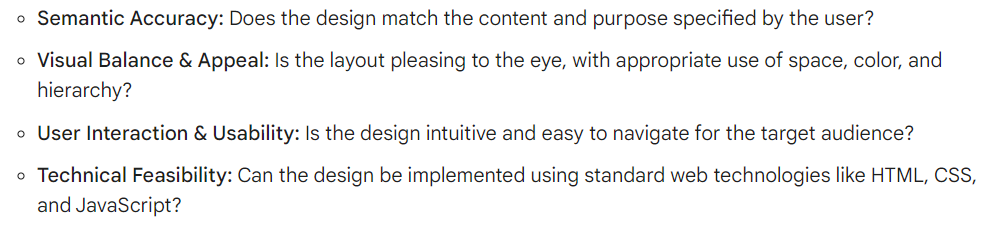
سوال 1-

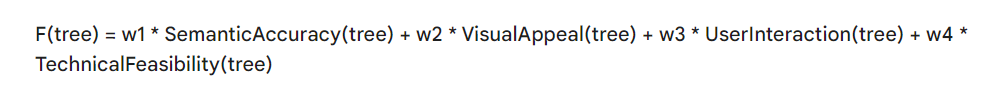
الف) ساختار ژنوم: برای ساختار ژنوم میتوان از همان parse tree مربوط به html استفاده کرد. Node ها همان تگها مثل html, body, head, p, a, ul خواهند بود و یالهای بین اینها رابطه والد و فرزندی را مشخص میکند. یعنی اگر یک گره فرزند گره دیگری بود، یعنی یک تگ درون یک تگ دیگر قرار گرفته. شکل زیر یک درخت با این ویژگیها را نشان میدهد.



جمعیت اولیه را با استفاده از روش ramped half-half میسازیم و ماکزیمم عمق را برابر با 10 میگیرم.

ب) تابع برازندگی(fitness function): برای این کار میتوان از فاکتورهای زیر استفاده کرد:





از طرفی میتوان با استفاده تکنیکهای text mining یک سری کلیدواژه از متن درخواستی کاربر استخراج کرد و بررسی کرد که ژنوم ساخته شده چقدر با این کلیدواژه ها overlap دارد و سپس بر اساس آن تابع برازندگی را تعریف کرد.

ج) با استفاده از توضیحات گفته شده توسط کاربر میتوان کلیدواژه های زیر را با استفاده از روشهای مختلف text mining استخراج کرد.

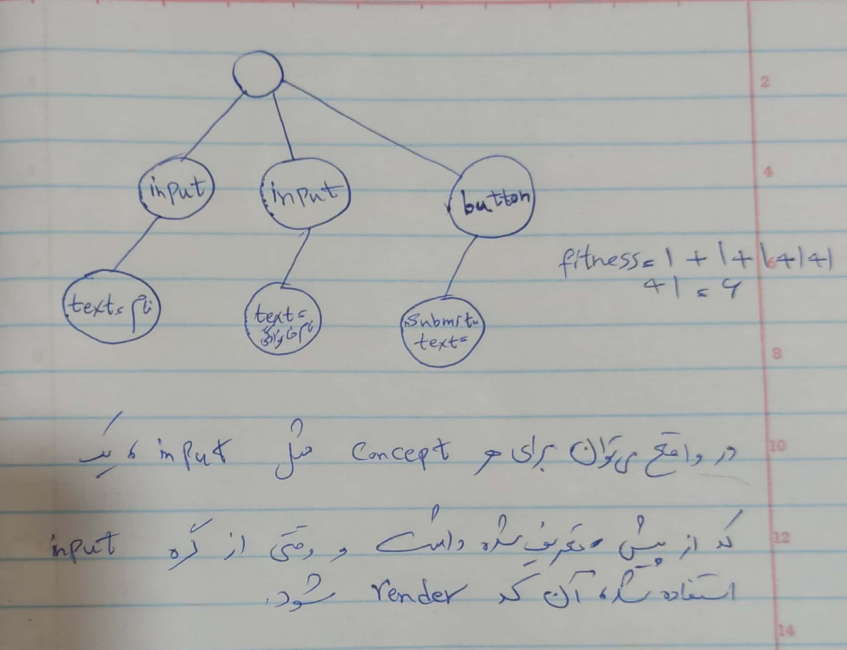
- input for name

- input for lastname

- button in the end

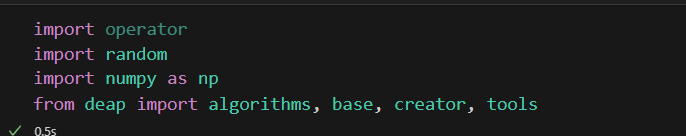
در عکس زیر، فرض کردم پس از چند مرحله به چنین چیزی میرسیم. مراحل میانی را در نظر نگرفتم.

در واقع باید mutation و crossover روی جمعیت اعمال شود و پس از آن به وسیله تابع fitness فرایند انتخاب صورت بگیرد تا پس از چندین نسل به درختهایی با fitness بالا برسیم.

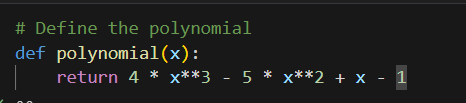


سوال 2- برای این سوال از کتابخانه deap برای اجرای الگوریتم ژنتیک استفاده کردم.

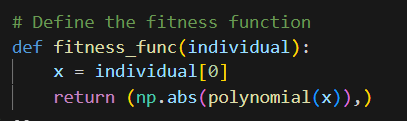
کتابخانه های مورد نیاز:



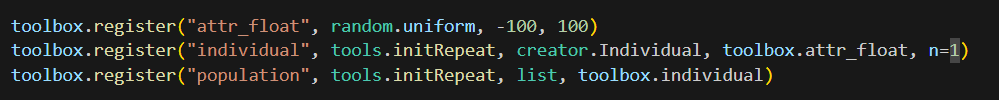
تابعی که میخواهیم ریشه آن را بدست بیاوریم:



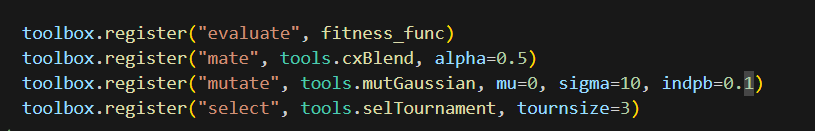
تابع fitness نیز باید تعریف شود تا مناسب بودن هر عضو از جمعیت را با استفاده از آن تعیین کنیم.



حال باید به وسیله ماژول base یک toolbox تعریف کنیم تا چگونگی ساخت جمعیت اولیه، mutaion، crossover و selection را مشخص کنیم.



در سه خط بالا، مشخص میکنیم که جمعیت اولیه باید به صورت یکنواخت بین -100 و 100 پخش شده باشند. پس فضای جستجوی ما بین این دو عدد است.

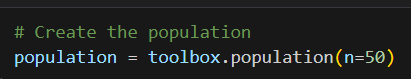
در خطوط بالا به ترتیب، تابع ارزیابی یا برازندگی، روش crossover، mutation و selection تعیین میشود.

عملگرهای ژنتیکی مورد استفاده در این کد عبارتند از:

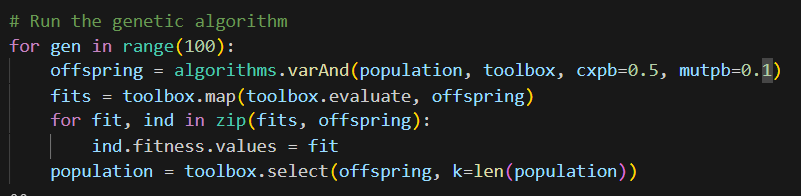
selection: انتخاب tournament با سایز 3.

crossover: blend crossover با مقدار آلفا 0.5

mutation: gaussian mutation با میانگین 0، انحراف معیار 10 و احتمال 0.1

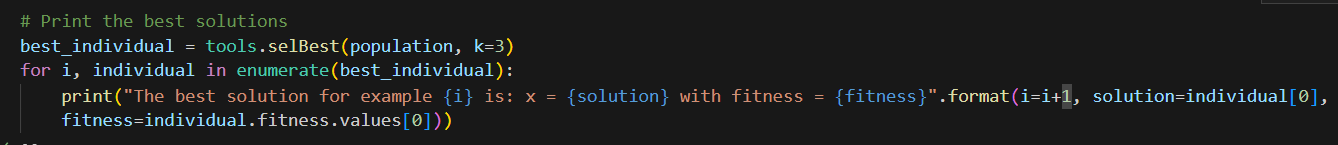


ساخت جمعیت اولیه با تعداد 50 تا.

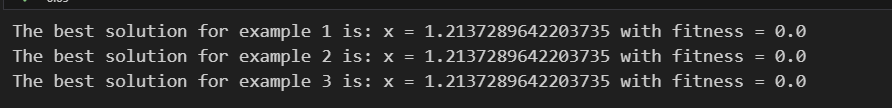


اجرای الگوریتم برای 100 نسل و سپس توقف. در هر بار اجرا، با استفاده از mutation و crossover به جمعیت افزوده میشود و در آخر از بین کل جمعیت selection انجام میشود. احتمال mutation همان 0.1 و احتمال crossover برابر با 0.5 گرفتیم.

در آخر سه تا از بهترین جوابها را چاپ کردیم:



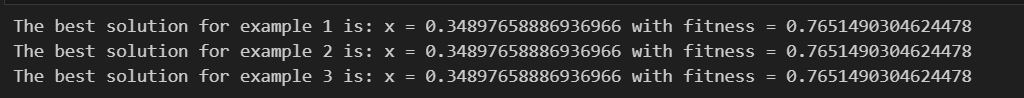
جواب برای تابع تعیین شده در اول گزارش:



جواب برای تابع زیر



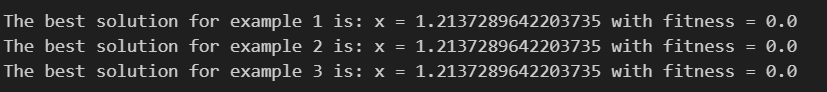
به صورت:



جواب برای تابع زیر:



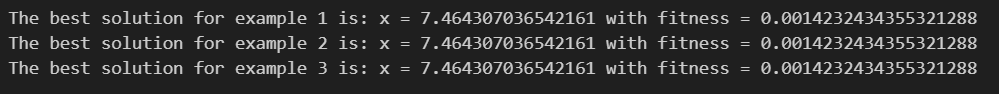
به صورت:



جواب برای تابع زیر:



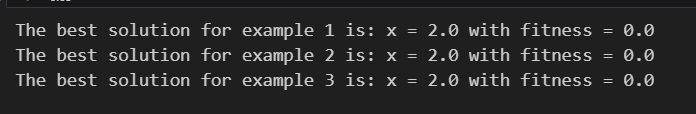
به صورت:



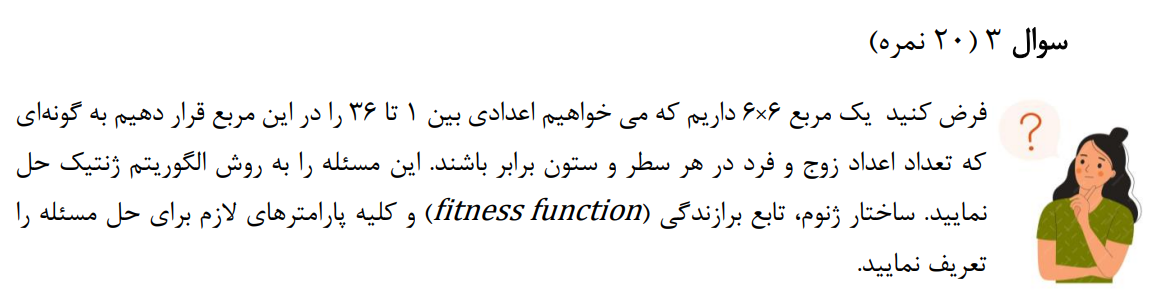
جواب برای تابع زیر:



به صورت:



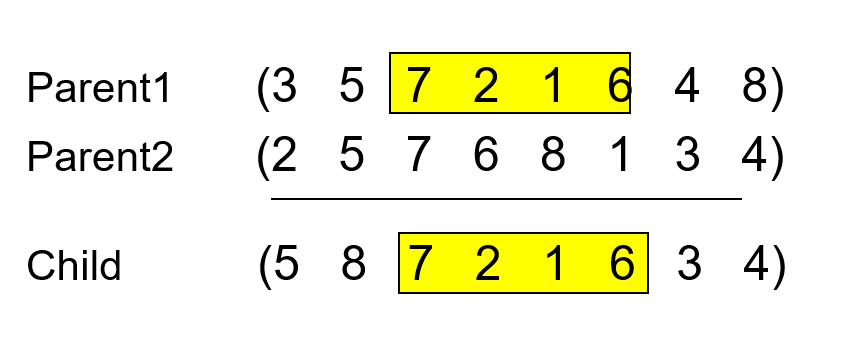
سوال 3-



ساختار ژنوم: ژنوم ما در اینجا به صورت یک ماتریس 6\*6 میباشد و در هر خانه یک عدد بین 1 تا 36 حضور دارد.

First population: تعداد جمعیت را 50 میگیرم و به این تعداد ماتریس 6\*6 با اعداد رندوم بین 1 تا 36 میسازم. فقط باید توجه شود که این ماتریس عدد تکراری نمیتواند داشته باشد و همه اعداد 1 تا 36 باید حضور داشته باشند.

Mutation and Crossover: ماتریس را میتوان flat کرد و در حالت flat شده Crossover را انجام داد. برای این کار از مشابهش در مسئله TSP استفاده میکنیم. یعنی یک قسمت از یکی را مستقیما در فرزند کپی کرده و بقیه را از والد دیگر به گونه ای انتخاب میکنیم که اعداد تکراری نداشته باشیم. مشابه شکل زیر:



تابع برازندگی(Fitness function):

هرچه عدد بالا به صفر نزدیک تر شود، تابع برازندگی بهتری است. اگر به صفر برسد یعنی به بهترین جواب رسیده‌ایم.

Selection: برای انتخاب، از روش baker method که از Roulette wheel بهره میبرد، استفاده میکنیم.

Termination: بعد از تعداد مشخصی نسل، الگوریتم را متوقف میکنیم. برای مثال پس از 50 نسل.

سوال 4-

9 % 3 + 1 = 1

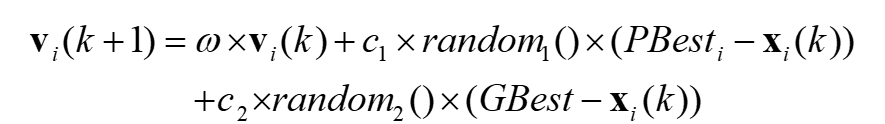
پس حالت اول باید شناسایی شود.

با استفاده از روش PSO این مسئله را حل میکنم.

برای این مسئله تعداد particle ها را برابر با n=100 میگیرم. برای هر کدام از 5 عکس ورودی، یک نقطه در فضای یک بعدی و به صورت افقی در کنار هم در نظر میگیرم و particle ها را در این نقاط که تعدادشان 5 تاست پخش میکنیم. از آنجا که فضا یک‌بعدی است، برای هر particle یک x و یک v در نظر میگیریم.

تابع برازندگی یا fitness: هر عکس را متناظر یک نقطه گرفتیم. هر مورچه که در نقطه متناظر با یکی از این عکسها قرار گرفت، با استفاده از correlation بین عکس اصلی (حالت اول) با عکس فعلی، میتواند fitness را برای خودش حساب کند. برای مثال اگر دو خانه متناظر از این دو عکس، یکی پیکسل روشن و دیگری پیکسل خاموش داشت، از مقدار fitness یکی کم و اگر یکسان بودند، به تابع fitness یکی اضافه میکنیم. پس بیشترین مقدار fitness عدد 16 و کمترین آن عدد -16 خواهد بود که برای زمانی است که دو عکس کاملا منفی یکدیگرند.

سرعت هر particle را با فرمول زیر بروز میکنیم.



مقادیر ثابت را به صورت زیر انتخاب میکنیم.

که طبق اسلایدها مقادیر مناسبی برای این الگوریتم هستند.

برای بروزرسانی جایگاه هر particle از فرمول زیر استفاده میکنیم:



مقادیر Gbest و Pbest نیز باید هربار بروز شوند. هر particle یک Pbest مخصوص خودش دارد و از بین این Pbest ها یکی که بهترین است به عنوان Gbest نگه داشته میشود.

Termination: مسئله را تا 200 iteration پیش می‌بریم و در نهایت با توجه به تابع برازندگی تعریف شده انتظار می‌رود مورچه ها یا همان particleها روی عکسهایی که همانند حالت اول هستند، جمع شده باشند. هرچه بیشتر شبیه حالت اول باشند، مورچه های بیشتری روی آنها خواهد بود.